Forschungszentrum Karlsruhe
Technik und Umwelt
Wissenschaftliche Berichte
FZKA 5698

Modellbasierte Objekterkennung mittels Neuronaler Netze unter Verwendung textureller Merkmale

M. Goik, H. Guth
Institut für Angewandte Informatik

Mai 1996

Forschungszentrum Karlsruhe

Technik und Umwelt

Wissenschaftliche Berichte

FZKA 5698

Modellbasierte Objekterkennung mittels Neuronaler Netze unter Verwendung textureller Merkmale

M. Goik*), H. Guth

Institut für Angewandte Informatik

*)von der Fakultät für Maschinenbau der Universität Karlsruhe genehmigte Dissertation

Forschungszentrum Karlsruhe GmbH, Karlsruhe

1996

Forschungszentrum Karlsruhe GmbH
Postfach 3640, 76021 Karlsruhe

ISSN 0947-8620

Modellbasierte Objekterkennung mittels Neuronaler Netze unter Verwendung textureller Merkmale

Zusammenfassung

Die vorliegende Dissertationsarbeit entstammt dem Themenkreis „Qualitätssicherung in der industriellen Fertigung“ im Kontext der optischen Inspektion von Strukturen. Es werden Probleme behandelt, die im Zusammenhang der Objekterkennung und Merkmalsextraktion in Grauwertbildern auftreten. Es wird ein spezielles modellbasiertes Verfahren beschrieben, um unter weitgehender Integration von CAD-Vorwissen der zu untersuchenden Strukturen eine maximal fehlerfreie Erkennung zu ermöglichen. Zu diesem Zweck ist die Verwendung textureller Merkmale notwendig, zu deren Klassifikation neuronale Netze eingesetzt werden.

Model-based Object Recognition by Neural Network based Textural Feature Discrimination

Abstract

The current Phd-thesis covers problems arising in optical quality measurements within industrial production of mechanical structures. These problems stem from general difficulties in visual object recognition and feature generation. A dedicated model based approach is described. This approach combines maximal integration of CAD knowledge of the underlying structures with use of textural features to achieve better results in the recognition process. A neural network based classifier is used for analysis of these textural features.

Inhaltsverzeichnis

Abbildungsverzeichnis

Glossar

ANN	Artificial Neural Network	16
FK	Forschungszentrum Karlsruhe - Technik und Umwelt	
GDSII	Datenformat des Elektronenstrahlschreibers, siehe auch Abb. 1.3	5
IAI	Institut für angewandte Informatik am → **FK**	
IGES	Datenformat des verwendeten CAD Systemes	4
IMT	Institut für Mikrostrukturtechnik. Ein Institut des → **FK**	
LIGA	RöntgentiefenLIthographie, Galvanoformung und KunststoffAbformung	
REM	Rasterelektronenmikroskop	

Kapitel 1

Allgemeine Einführung

1.1 Hintergrund

Die vorliegende Arbeit beschäftigt sich mit Verfahren zur Verbesserung der bestehenden bildanalytischen Qualitätssicherung von Mikrostrukturen. Der Hintergrund ist die am Forschungszentrum Karlsruhe laufende Erforschung von Verfahren zur Herstellung von Mikrostrukturen.

Die bekannteren Techniken in diesem Forschungszweig werden durch die Halbleiterfertigung, speziell in der Silizium-Technologie, repräsentiert, welche bislang überwiegend zur Herstellung integrierter Schaltkreise verwandt wird. Während es sich dabei in der industriellen Großfertigung überwiegend um rein elektronische Elemente handelt, liegt das Augenmerk am Forschungszentrum Karlsruhe auf der Herstellung mechanischer Mikrostrukturen. Bekanntere Beispiele, bei denen Strukturen dieser Art Verwendung finden, sind die Mikropumpe und der Beschleunigungssensor [BM91].

Die Grundlage zur Herstellung solcher Strukturen bildet der *LIGA*-Prozeß [MB93], es handelt dabei um ein röntgenlithographisches Verfahren.

Der Demonstrator eines Systems zur Messung von Beschleunigungen [SFK+93] zeigt exemplarisch den Übergang von einer Mikrostruktur zu einem Mikrosystem. Die mechanische Komponente wird dabei gekoppelt mit einem Mikrokontroller und bildet ein Gesamtsystem zur einfachen Integration etwa im Bereich der Kraftfahrzeugindustrie.

Die Anwendungsmöglichkeiten dieser Mikrosysteme sind in sehr verschiedenen Bereichen zu finden, beispielsweise in der Medizintechnik. Als ambitioniertes Ziel findet sich in diesem Feld die Konstruktion eines künstlichen Implantates zur Steuerung des Blutzuckerspiegels insulinpflichtiger Patienten. Für ein sol-

ches Mikrosystem muß eine mechanische Insulinpumpe und ein Sensor für den Blutzuckerspiegel mit einer intelligenten Auswerte- und Steuerungselektronik gekoppelt werden, um eine korrekte Dosierung zu erreichen. Hierbei ist die Zuverlässigkeit des Gesamtsystems eine unverzichtbare Grundvoraussetzung für den tatsächlichen Einsatz, daher ist eine massive Qualitätskontrolle aller beteiligten Komponenten unabdingbar.

1.2 Qualitätskontrolle

Ein langfristiges Ziel der am Forschungszentrum Karlsruhe laufenden Arbeiten ist der Nachweis, daß Mikrosysteme zu akzeptablen Kosten gefertigt werden können. Dieser Nachweis ist die Voraussetzung zu einer späteren industriellen Serienfertigung. Dazu wird für die einzelnen Produktionsschritte eine automatisierte Qualitätssicherung notwendig. Beispielsweise müssen fehlerhafte Bauelemente möglichst frühzeitig ausgesondert werden, um nicht unnötig die Kapazitäten der folgenden Produktionsschritte zu belasten.

Ein weiterer wesentlicher Grund besteht in der Optimierung von Prozeßparametern hinsichtlich einer optimalen Ausbeute bei möglichst minimalem Einsatz von Ressourcen.

Schließlich ist als dritter Punkt für die Notwendigkeit einer Qualitätskontrolle die DIN/ISO 9000 [DIN90] Norm zu nennen, welche neben einer lückenlosen Dokumentation der Fertigung auch einen Qualitätsnachweis der einzelnen Fertigungsschritte festschreibt.

Die Realisierung der Qualitätssicherung muß dabei grundsätzlich anders erfolgen als in der Halbleiterfertigung. Die Funktionskontrolle kann dort überwiegend elektronisch erfolgen. Der zu untersuchende Schaltkreis wird dabei an einen Testautomaten angekoppelt, welcher ein Prüfprogramm ausführt.

In der Mikrosystemtechnik besteht die Notwendigkeit zum Nachweis des mechanischen Funktionierens eines Bauteiles. Eine notwendige Vorbedingung ist das Einhalten gewisser geometrischer Toleranzgrenzen der gefertigten Strukturen. Dieser Nachweis erfolgt durch die Analyse von Aufnahmen, welche entweder mit einem Licht- oder einem Rasterelektronenmikroskop (REM) gewonnen werden. Diese Aufnahmen werden in frühen Phasen der Entwicklung eines Mikrosystems direkt untersucht und ausgewertet. Bei der Herstellung von Prototypen kann die Auswertung noch visuell durch einen Menschen erfolgen. In einer industriellen Massenfertigung hingegen ist eine automatisierte Erfassung und Auswertung unumgänglich. Aus diesem Grund ist die Qualitätskontrolle notwendige Voraussetzung für die industrielle Fertigung.

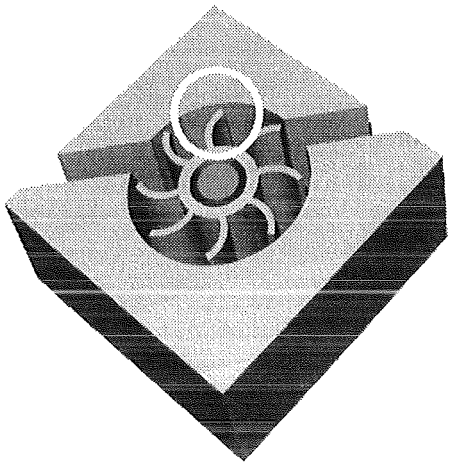


Abbildung 1.1: Dieses Bild zeigt eine vereinfachte, synthetisch erzeugte Darstellung für eine Mikroturbine. Die Funktion der Turbine ist nicht garantiert, kann aber aufgrund der aus der gewählten Blickrichtung stammenden Bildinformation nicht ausgeschlossen werden.

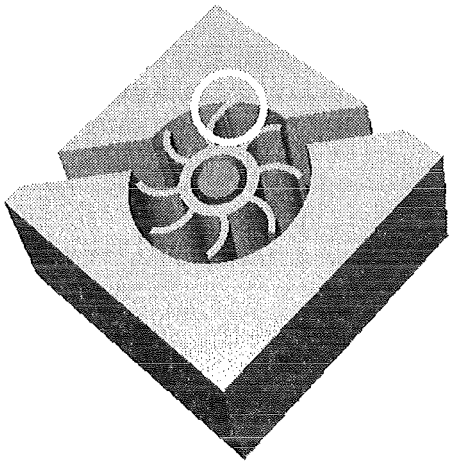


Abbildung 1.2: Die hier dargestellte Turbine würde von einer weiteren Fertigung ausgeschlossen werden. Der weiße Kreis zeigt auf einen Verbindungssteg, durch welchen der Rotor mit dem feststehenden Rahmen verbunden ist.

Das Ergebnis der Qualitätskontrolle ist eine Bewertung, ob eine bestimmte Komponente eines späteren Mikrosystemes weiterverwendet wird, oder verworfen werden muß.

Abb. 1.1 zeigt als Beispiel ein synthetisch generiertes Bild einer Mikroturbine, wie sie in ähnlicher Form am IMT gefertigt wurde. Ein solches Bauteil kann dazu benützt werden, die Strömungsgeschwindigkeit eines fließenden Mediums zu bestimmen. In Abb. 1.2 ist hingegen ein nicht funktionsfähiges Bauteil dargestellt. Ein während der Herstellung nicht entfernter Steg verhindert die Rotation des Turbinenrades. Die Tatsache steht bereits in diesem Stadium des Produktionsprozesses fest, würde aber in einer Serienfertigung erst bei einer später erfolgenden Funktionskontrolle des Gesamtsystemes erkannt. Die Konsequenz wäre das unnötige Durchlaufen nachfolgender Fertigungsschritte dieses fehlerhaften Bauteiles.

1.3 Stand der Technik

Bei der Diskussion über Systeme zur optischen Qualitätssicherung wird prinzipiell zwischen zwei Arten von Systemen unterschieden:

- Inspektionssysteme
- Vermessungssysteme

Inspektionssysteme dienen der schnellen Auswahl nicht verwendungsfähiger Elemente. Dabei werden aus den zu untersuchenden Bildern bestimmte Parameter generiert. Diese Parameter werden dann statistisch klassifiziert. Das inspizierte Objekt wird als gut bewertet, solange die Parameter innerhalb bestimmter Grenzen liegen. Ein guter Überblick über die grundsätzlich verwendeten Verfahren zur visuellen Inspektion findet sich in [ST92].

Vermessungssysteme sind dagegen gekennzeichnet durch die Fähigkeit, bestimmte geometrische Parameter innerhalb einer geforderten Genauigkeit zu ermitteln. Dieser gegenüber einem Inspektionssystem größere Aufwand bedingt einen erhöhten Zeitaufwand.

Drei Beispiele prototypischer Vermessungssysteme auf lichtoptischer Basis sind [RM90], [Lüb] und [Bür91a].

Das in [RM90] vorgestellte System dient der Qualitätskontrolle elektronischer Schaltkreise unter Verwendung der CAD-Entwurfsdaten auf *symbolischer* Ebene. Dies bedeutet, daß kein einfaches Pixel-Matching Verfahren, wie in [Ull89] beschrieben, sondern eine Erkennung höherwertiger Merkmale durchgeführt wird. Dies erlaubt dann Aussagen zu treffen wie „Verbindung ist nicht unterbrochen".

Der Fokus des in [Lüb] vorgestellten Systems liegt in der exakten Vermessung von Werkstücken, wie sie etwa im Maschinenbau anfallen. Dazu zählt beispielsweise die korrekte Positionierung und Form von Bohrlöchern innerhalb geforderter Toleranzen.

Der Gegenstand dieser Arbeit schließlich ist die Weiterentwicklung der Bildverarbeitungskomponente des dritten oben genannten Vermessungssystems. Es handelt sich dabei um das *COSMOS-2D* System [Bür91a], welches am IAI entwickelt wurde.

Der wesentliche Ansatz bei *COSMOS-2D* ist analog zu [RM90] die Verwendung des Geometriewissens über eine zu untersuchende Struktur. Der Konstrukteur einer solchen Struktur erzeugt seinen Entwurf auf einem mechanischen CAD System, das Ergebnis ist eine Beschreibung im *IGES* Format. Dadurch ist das *Soll*wissen über die Geometrie festgelegt.

Die zur Röntgenlithographie notwendige Maske zur „Belichtung“ wird an einem Elektronenstrahlschreiber erzeugt. Dazu werden die mechanischen CAD Daten in ein entsprechendes Datenformat, das *GDSII* Format, konvertiert. Danach entstehen bei jedem Prozeßschritt des *LIGA* Verfahrens Strukturen, welche Gegenstand von Vermessungsaufträgen an *COSMOS-2D* sein können. Diese Vermessungsaufträge werden ebenfalls vom Konstrukteur generiert. Die Ist -Daten der erzeugten Struktur bestimmt das *COSMOS-2D* System auf folgende Weise: Aus dem Grauwertbild der Aufnahme wird eine Pixelliste erzeugt, Details siehe [Mas88]. Diese enthält die als Kanten detektierten Pixel, sowie Richtungsinformationen über die zugehörige Kante. Aus dieser Pixelliste wird danach eine symbolische Beschreibung erkannter Strukturelemente mit zugehörigen Startparametern gewonnen. Diese werden dann verwendet, um durch ein least-square-Verfahren die genauen Parameter zu bestimmen.

Das *COSMOS-2D* System benutzt zum Bestimmen der Ist -Daten die CAD - Entwurfsdaten für den Erkennungsalgorithmus. Die damit verbundenen Probleme werden auf Seite 29 näher beschrieben. Abb. 1.3 zeigt die Integration von *COSMOS-2D* in den Fertigungsprozeß.

Darstellungen von *COSMOS-2D* finden sich in [BGH89], [BGH91c], [Bür91b], [BGH91a], [BGH91b] und [Bür91a].

1.4 Motivation der vorliegenden Arbeit

Der grundlegende Idee dieser Arbeit besteht in der *Integration des Modellwissens der Struktur* zur Verbesserung der Kanten- und Eckenextraktion.

Die allgemeinen Verfahren der Bildverarbeitung zur Kanten- und Eckenextraktion werden seit über 30 Jahren untersucht und können nicht verbessert werden. Sie benutzen allerdings auch kein Vorwissen über die betrachtete Szene. Ein Beispiel für ein solches am IAI entwickeltes System zur 3D Objekterkennung ohne Szenenvorwissen wird in [KHHK94] beschrieben.

Da im Fall von *COSMOS-2D* durch den CAD Entwurf ein sehr weitgehendes Vorwissen über die erzeugten Strukturen vorhanden ist, kann diese Information zur *Verbesserung* der Erkennung auf der untersten Ebene der Bildverarbeitung beitragen.

Ein weiterer gewichtiger Aspekt dieser Arbeit betrifft den Fall texturdefinierter Übergänge. Diese treten immer dann auf, wenn eine Kante nicht durch einen Sprung im mittleren Grauwert $\bar{I}$, sondern durch die Veränderung textureller Merkmale definiert wird, dies ist in Abb. 1.4 dargestellt.

In einem solchen Fall versagen die in *COSMOS-2D* verwendeten klassischen Al-

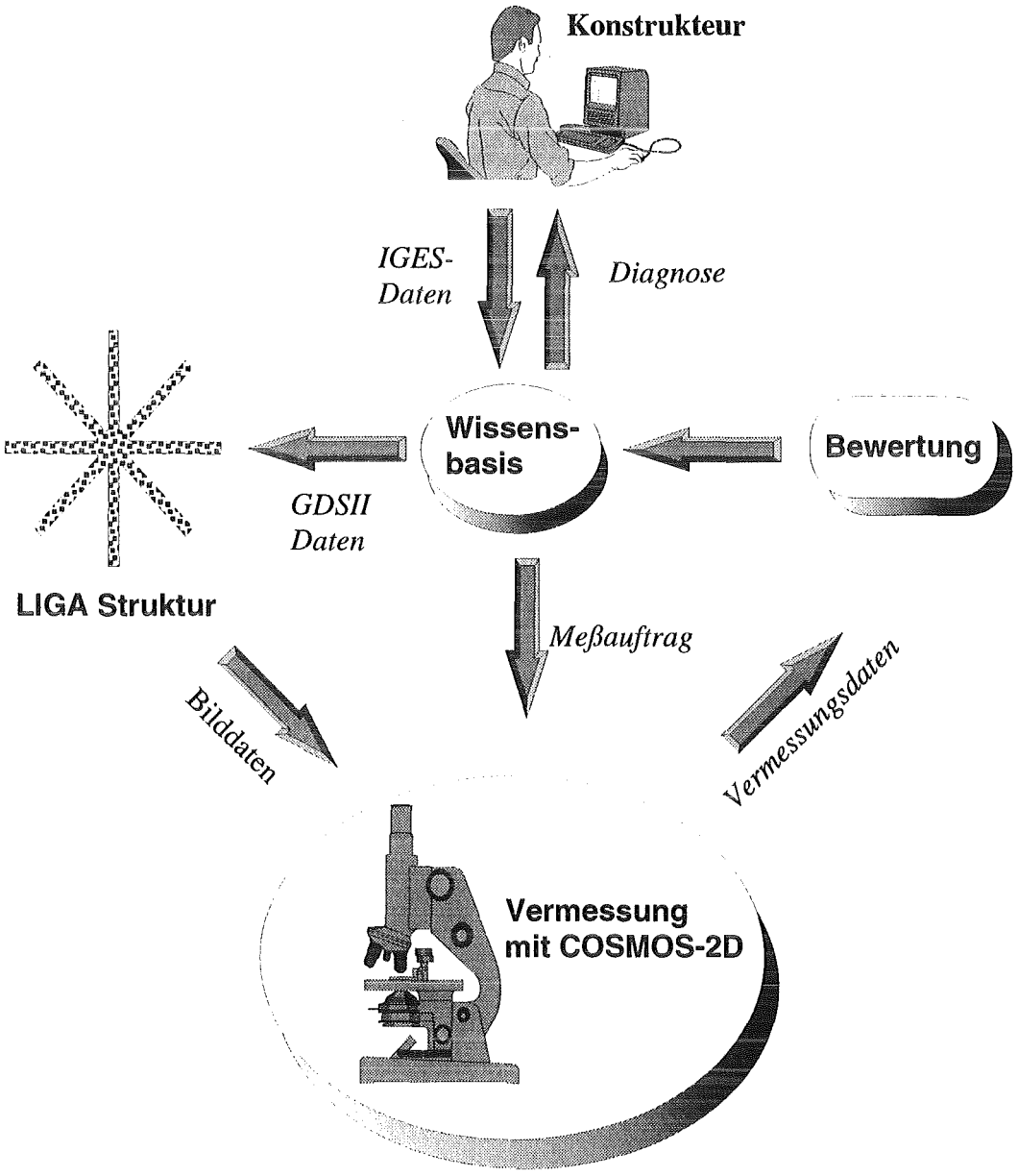


Abbildung 1.3: Schema des Gesamtsystemes

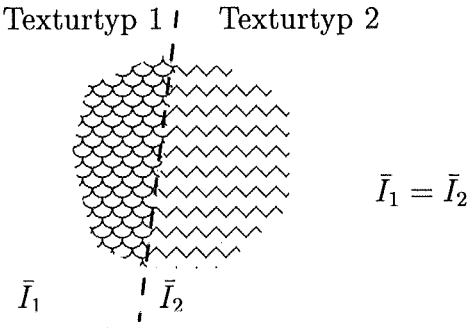


Abbildung 1.4: Texturdefinierter Übergang

gorithmen zur Kantenextraktion, da diese *nur* auf Grauwertunterschiede, aber nicht auf Texturunterschiede ansprechen. Dies wird in §3.1.2 genauer diskutiert.

Das *COSMOS-2D* System zur Vermessung von Strukturen weist Probleme aus zwei unterschiedlichen Quellen auf:

- Probleme auf der Bildverarbeitungsebene
  - Strukturkantenextraktion
    - Vorhandene Kanten werden nicht gefunden
    - Nicht vorhandene Kanten werden als Kanten erkannt
  - Prozeßbedingte Strukturveränderungen
- Erweiterung zu einem dreidimensionalen Vermessungssystem.

Der erste Unterpunkt allgemeiner Bildverarbeitungsprobleme betrifft die Verarbeitung von Kantenbildern im *COSMOS-2D* System. Hier sind Defizite vorhanden im Fall schwierig zu detektierender Kanten und Ecken.

Die tieferliegende Ursache für die Probleme bei schwierigen Bildern in *COSMOS-2D* ist die Verwendung eines General Purpose Kantenoperators zur Erzeugung des Binärbildes. Dieser Kantenoperator verwendet, im Gegensatz zu *COSMOS-2D* selbst, *keine* Vorinformation über das zu extrahierende Grauwertbild, sondern lediglich Intensitätsinformationen einer Umgebung des jeweils betrachteten Pixels. Aus diesem Grund reagiert das Verfahren sehr stark auf Störungen, wie sie bei problematischen Strukturen oder bei einem schlechten Signal/Rauschverhältnis auftreten. Die Folgen sind einerseits fehlende Erkennung vorhandener Elemente einer Struktur und andererseits Fehlklassifikationen.

Ein komplexeres Verfahren benötigt mehr Zeit für diese Aufgabe, allerdings kann es beschränkt bleiben auf die dem Meßauftrag zugrundeliegenden Bildbereiche.

In den Übersichtsartikekn [CD86] und [SFH92] wird dargestellt, in welcher Weise Vorwissen über die zu erkennende Szene die Rekonstruktion erleichtert.

Um auf diese Weise Effizienz zu erreichen, müssen die besonderen Randbedingungen eines Vermessungssystems berücksichtigt werden. Die Ausgangslage wird dadurch günstiger, als es bei dem allgemeineren Paradigma der Bildverarbeitung, nämlich der Klassifikation einer unbekannten Szene bei bekannten Bildprimitiven der Fall ist. Das weitgehende Vorwissen ermöglicht, wie in Kapitel 3 gezeigt wird, eine effiziente Gewinnung der durch einen Meßauftrag spezifizierten Informationen. Dies ist wichtig im Hinblick auf die *Echtzeitanforderungen* im industriellen Einsatz.

Der Unterpunkt der prozeßbedingten Strukturveränderungen betrifft das Herstellungsverfahren der zu vermessenden Proben. Das *LIGA* Verfahren kennt etwa 250 verschiedene Prozeßschritte. Viele dieser Einzelschritte verursachen Abweichungen von der Sollstruktur. Als Beispiel für ein solches Verhalten soll hier der *Proximity Effekt* dienen, welcher in Abb. 1.5 illustriert wird.

Aufgrund der Verwendung von kurzwelliger Röntgenstrahlung treten *vernachlässigbare* Interferenzerscheinungen auf, allerdings ist die Wechselwirkung der energiereichen Röntgenstrahlung mit dem Substrat zu berücksichtigen. Aus diesem Grund werden auch Bereiche „belichtet", die in geometrischen „Schattenzonen" liegen. Um den Einfluß dieses schädlichen Effekts möglichst gering zu halten, kann man die Dicke p der strahlungsempfindlichen Schicht gering halten. Die Energie der verwendeten Röntgenstrahlung spielt ebenfalls eine Rolle.

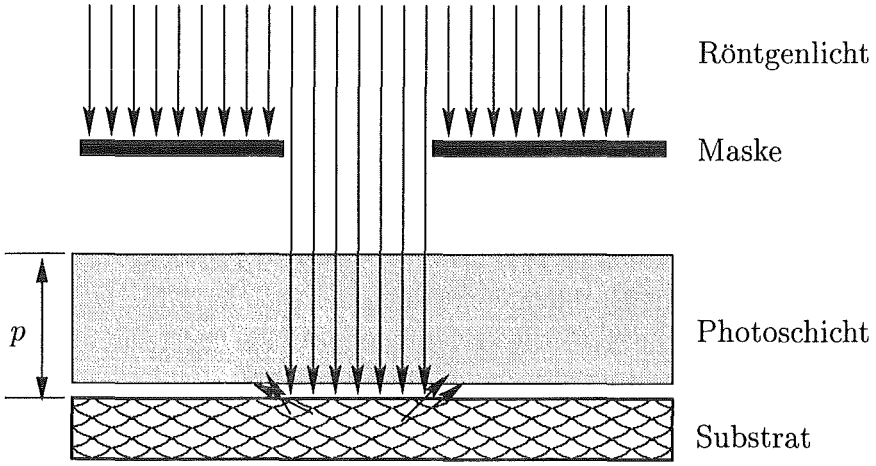


Abbildung 1.5: Der Proximityeffekt tritt auf infolge der Wechselwirkung der energiereichen Röntgenstrahlung mit dem Substrat. Dadurch werden Elektronen aus dem Substrat herausgeschlagen, welche die Photoschicht schädigen. Es erfolgt eine Zerstörung der Photoschicht in die geometrische „Schattenzone" hinein.

Der Proximityeffekt bewirkt eine Verrundung der Ecken einer Struktur, Abb. 1.6

zeigt ein Beispiel. Die Sollstruktur weist zwei sehr spitzwinklige Ecken auf. Diese erleiden im Laufe des Fertigungsprozesses eine unterschiedlich starke Verrundung.

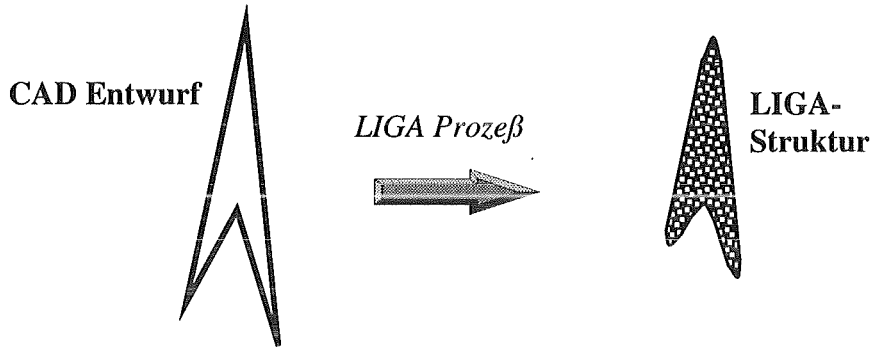


Abbildung 1.6: Es wird ein Beispiel gezeigt, wie sich die Ausprägung eines CAD-Entwurfes durch Einfluß des LIGA *Verfahrens verändert. In diesem Fall treten Verrundungen durch Ätzprozesse ein*

COSMOS-2D berücksichtigt nur das CAD Vorwissen, nicht jedoch ein weitergehendes Prozeßwissen. In Abb. 1.7 wird das Beispiel des Meßauftrages „Distanz zweier Ecken“ dargestellt.

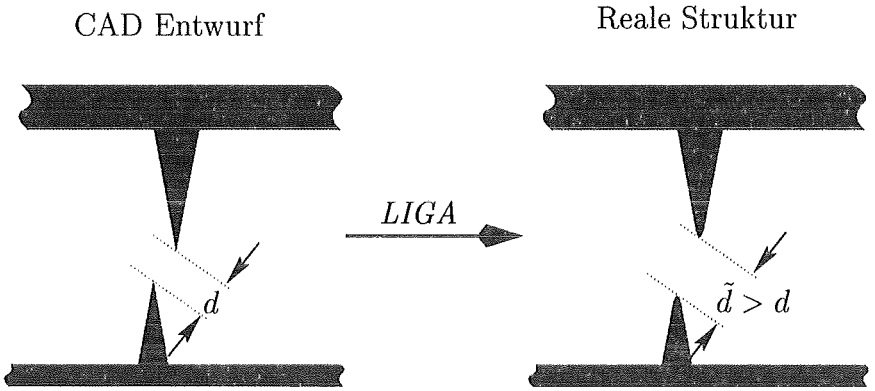


Abbildung 1.7: Die Darstellung zeigt den Einfluß des Herstellungsprozesses auf das Ergebnis eines Meßauftrages. Der CAD Entwurf ergibt den Wert d für den Abstand der beiden Spitzen. In der realen Struktur ist eine Verrundung der Ecken eingetreten, daher ist der gemessene Abstand $\tilde{d}$ größer als d.

Der linke Teil zeigt einen CAD-Entwurf mit zwei spitzwinkligen Ecken. Da diese Struktur im *LIGA* Verfahren entsteht, erfahren diese Ecken des CAD- Entwurfs eine Verrundung. Dies führt bereits zu Problemen in der Vermessung, da das *COSMOS-2D* System versucht, zwei ideale Ecken zu erkennen. Das Ergebnis der Vermessung dieser Idealstruktur ist der Abstand d. Der tatsächliche Abstand $\tilde{d}$ wird gegenüber diesem Wert i. allg. größer sein.

Dem zweiten Ziel einer Erweiterung auf ein dreidimensionales Vermessungssystem liegt eine Eigenschaft der *LIGA* Technologie zugrunde. Im Verhältnis zur Siliziumtechnologie ist ein höheres Aspektverhältnis erzielbar. *LIGA-* Strukturen können im Verhältnis zu ihrer lateralen Ausdehnung eine sehr große räumliche Tiefe besitzen. Daher bedeutet die Beschränkung von *COSMOS-2D* auf eine „quasi zweidimensionale" Szenenanalyse eine wesentliche Einschränkung für den Anwender. Die Tiefe einer Struktur ist gegenüber der lateralen Ausdehnung nicht vernachlässigbar. Eine 3-D Vermessung stellt erhebliche Mehranforderungen an die zugrundeliegende Bildverarbeitung. Hierzu müssen Paare von Bildern im Stereoverfahren analysiert werden, das Prinzip ist in Abb. 1.9 skizziert.

Auch hier wird in der ersten Stufe eine Extraktion der Kanten vorgenommen. Die Differenzen zwischen den Kantenbildern zweier korrespondierender Stereobilder ermöglichen dann eine zumindest teilweise Rekonstruktion der zugrundeliegenden dreidimensionalen Szene.

Das besondere Problem dieser Rekonstruktion besteht in der Tatsache, daß diese Differenzen der Stereobildpaare gering sind in Relation zur räumlichen Ausdehnung der im Bildpaar enthaltenen Gegenstände. Geringe Lokalisierungsfehler der aus dem Kantenbild gewonnenen Bildprimitive haben zudem *massive* Fehler in den gesuchten Maßen der dreidimensionalen Szenenbeschreibung zur Folge. Aus diesem Grund ist für eine gute Rekonstruktion der Originalszene, wie sie von einem *Vermessungssystem* erwartet wird, eine sehr exakte Lokalisierung der Bildprimitive notwendig. Die Genauigkeit von Kantenoperatoren genügt diesem Ziel nur bei relativ störungsfreien Bildern. In Kapitel 3 wird gezeigt, wie sich Subpixelgenauigkeit bei Verwendung eines Modelles der zu erkennenden Bildprimitive erreichen läßt.

Abb. 1.8 demonstriert ein weiteres zusätzliches Problem einer 3-D Vermessung. Für die hier gezeigten Säulen mit trihedraler Struktur erhält man im Gegensatz zur zweidimensionalen Vermessung Ecken mit maximal drei auslaufenden Kanten.

Details für die Realisierung eines dreidimensionalen Vermessungssystems finden sich in [Stu94] und in [GDG+93].

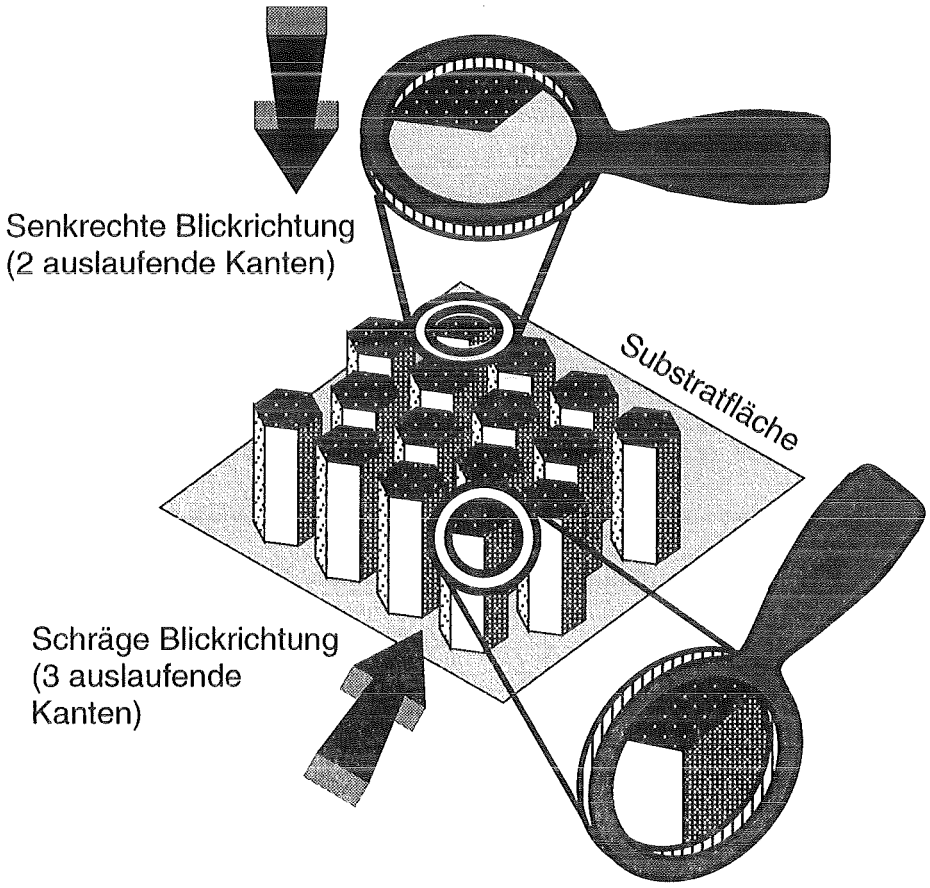


Abbildung 1.8: Dies ist ein Beispiel für die 3-D Ansichten einer Struktur. Bei den Säulen handelt es sich um trihedrale Elemente, d.h. jede Ecke hat genau drei einlaufende Linien. Die Ansicht von oben liefert eine Ecke mit zwei einlaufenden Linien. Dies ist ein Artefakt der gewählten Projektionsrichtung und des Merkmals, daß die auftretenden Kanten entweder parallel oder senkrecht zur Grundfläche verlaufen. Die schräge Ansicht zeigt hingegen den trihedralen Charakter der Struktur.

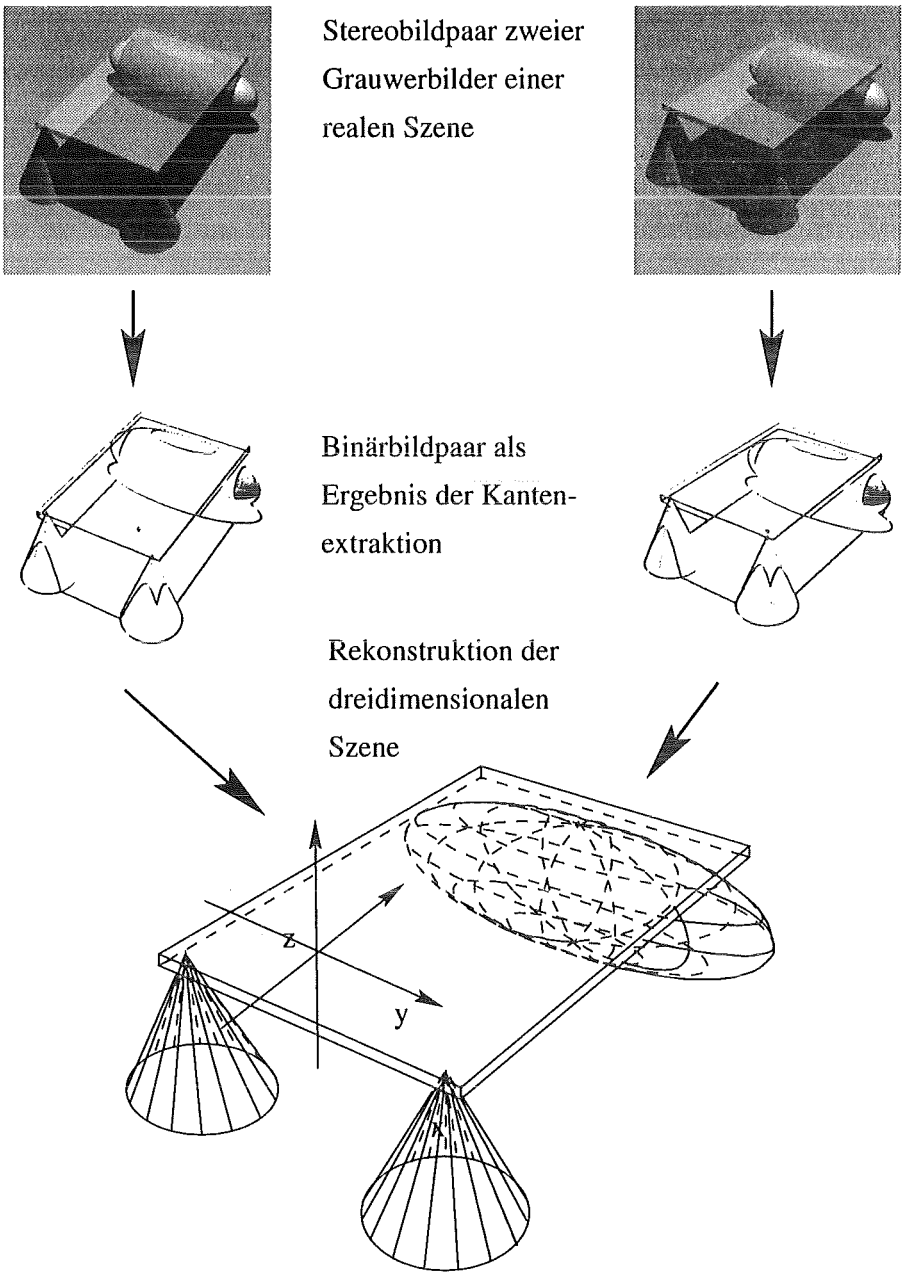


Abbildung 1.9: Dies ist das Prinzip der Rekonstruktion einer dreidimensionalen Szene. Aus den beiden Grauwertbildern werden Kantenbilder erzeugt. Aus diesen werden dann über Projektionsrechnungen die Parameter für das 3-D Modell bestimmt.

1.5 Gliederung der Arbeit

In §2 wird der Versuch beschrieben, durch Einsatz eines neuronalen Filters implizit die Erkennung gewünschter Kantentypen zu kodieren und so eine direkte Kantenerkennung zu ermöglichen. Dies entspricht einem klassischen Kantenfilter, dessen Kern auf ein bestimmtes Intensitätsmodell zugeschnitten ist. Es wird ein Vergleich der Ergebnisse mit klassischen Operatoren gegeben.

§3 diskutiert zunächst die bisher vorhandene Erkennung und deckt die Schwächen von „general purpose" Kantenoperatoren auf. Es wird eine Gegenüberstellung modellbasierter Verfahren mit Diskussion der Vor- und Nachteile gegeben. Anschließend wird gezeigt, wie die Integration des CAD-Vorwissens in die Modellbildung eingeht und auf welche Weise die fehlenden nichtgeometrischen Parameter bestimmt werden können.

§3.5 beschreibt eine Kombination von Verfahren zur Verbesserung der i. allg. fehlerbehafteten CAD-Geometriedaten. Quelle solcher Fehler sind durch den Herstellungsprozeß bedingten Abweichungen und Störungen des Bildaufnahmesystems. Anschließend wird in §3.5.4 die Zusammenführung der Teilkomponenten beschrieben.

§4 liefert zunächst eine Begründung, weshalb die vorgestellte modellbasierte Erkennung unter bestimmten Umständen versagt, obwohl eine visuelle Erkennung offenkundig möglich ist. Dies führt auf die Notwendigkeit der Integration texturelle r Merkmale in den Erkennungsprozeß. Es wird ein Überblick über die verschiedenen „Definitionen" des Texturbegriffs gegeben. Anschließend wird ein neuronal basiertes Verfahren beschrieben, um eine Einteilung in beliebig vorgegebene Texturklassen zu ermöglichen.

§5 liefert eine Zusammenfassung der experimentellen Ergebnisse sowohl für den Fall eines reinen Intensitätsmodells, als auch für die Verwendung textureller Merkmale. Dabei werden sowohl synthetisch generierte als auch reale Grauwertbilder betrachtet.

§A gibt einen mathematischen Anhang über die Realisierung von Restriktionen neuronaler Netze auf die *transformationsinvariante* Erkennung von geometrischen Mustern.

§B gibt eine allgemeine Einführung über die an mehreren Stellen in dieser Arbeit verwendeten Methoden der nichtlinearen Ausgleichsrechnung.

§C liefert eine Darstellung der mathematischen Verfahren der für diese Arbeit relevanten Methode der statistischen Texturklassifikation. Es werden die sogenannten Co-occurrence Matrizen dargestellt, welche es erlauben, statistische Merkmale zweiter Ordnung eines Grauwertbildes zu erzeugen.

§D stellt die Verwendung der verwendeten neuronalen Netze vom Backpropagation Typ vor.

§E erläutert Details der verwendeten Entwurfstechniken. Es wird gezeigt, wie objektorientierte Techniken verwendet werden können, um eine benutzertransparente Schnittstelle zu erzeugen, welche dennoch den Mehraufwand für funktionelle Erweiterungen gering hält.

Kapitel 2

Direkte transformationsinvariante Kantenerkennung

2.1 Grundlagen

Die Grundlage des in diesem Kapitel vorgestellten Verfahrens ist die Erweiterung des Prinzips eines allgemeinen Kantenfilters. Betrachtet man die in Abb. 3.12 vorgestellten Kantenübergänge, so stellt sich heraus, daß zu jedem besonderen Typ von Kantenübergang ein eigener speziell zu wählender Operatorkern gehört, der eine optimale Erkennung gewährleistet. In [Can86] wird ein allgemeiner Formalismus beschrieben, wie ein solcher Operatorkern gewonnen wird. Die Voraussetzung dafür ist eine analytische Beschreibung des Funktionsverlaufs.

Es gibt nun bei realen Bildern mindestens zwei Probleme:

- Die auftretenden Kantenübergänge sind i. allg. nicht vom selben Typ, da die Ursachen ihrer Entstehung verschieden sind. Der Kantenoperator muß daher eine Gruppe endlich vieler Kantentypen erkennen, was im Widerspruch zur optimalen Erkennung steht.

- Die Übergänge eines bestimmten Typs sind analytisch nicht bekannt, da die bestimmenden physikalischen Grundlagen entweder zu komplex, bzw. die darin eingehenden Parameter nur teilweise bekannt sind. Ein Kantenverlauf wie in Abb. 3.12, Teil 4, entsteht beispielsweise durch Interferenz. Die Form wird daher durch die Wellenlänge des einfallenden Lichts bestimmt, welche i. allg. nicht genau bekannt ist.

Die Idee besteht nun darin, das im ersten Punkt genannte Dilemma möglichst gut durch Einsatz eines neuronalen Verfahrens aufzulösen.

Ein solches Verfahren wird an Bildern jenes Typs trainiert, auf die der resultierende Operator angewandt werden soll. Auf diese Weise wird das Problem umgangen, sowohl analytische Form als auch die zugehörigen Parameter des Kantenüberganges zu kennen. Diese treten dann implizit kodiert auf in Form der Gewichte des durch Training generierten Neuronalen Netzes.

Der zweite Punkt des oben genannten Dilemmas ist nur zu umgehen, wenn die Qualität der Erkennung für bestimmte Kantentypen auf Kosten der Erkennung anderer Typen steigt. Dies kann z.B. der Fall sein, wenn durch Verdeckung entstehende Kanten eines Bildes nicht interessieren.

Gegenüber einem herkömmlichen Operator kann eine Verbesserung der Erkennung eintreten, wenn Besonderheiten des Bildtyps eine Rolle spielen.

2.2 Einsatz eines Neuronalen Netzes

2.2.1 Grundlegende Eigenschaften

Es würde den Rahmen dieser Darstellung sprengen, eine Einführung in die verschiedenen Typen neuronaler Netze zu geben.

Etwas genauer geht es um den Einsatz *künstlicher* neuronaler Netze. Die Leistunsfähigkeit biologischer Nervensysteme in der Mustererkennung (Visuelle und akustische Erkennung) ließ die Idee aufkommen, das Verhalten solcher Netze nachzuahmen. Das kleinste Element biologischer Systeme ist das einzelne Neuron. Dieses hat eine bestimmte Anzahl von Eingangssignalen, die es nach bis heute im Detail verborgenen Gesetzmäßigkeiten zu einem Ausgangssignal weiterverarbeitet. Künstliche neuronale Netze (ANN) enthalten ein stark vereinfachtes Modell biologischer Neuronen. Im einfachsten Fall werden die i verschiedenen Eingangssignale mit Faktoren ω_i gewichtet und aufsummiert. Sobald diese Summe einen gewissen Schwellwert überschreitet, steigt der Ausgangspegel an, das Neuron „feuert„. Dies ist in Abb. 2.3 skizziert.

Ein Ensemble solcher „Neuronen“ wird dann, wie sein biologisches Vorbild, zu einem Netz verknüpft, indem die Ausgänge wieder als Eingangssignale für andere Neuronen dienen, wie in Abb. 2.6 dargestellt.

Gute einführende Darstellungen solcher ANN's finden sich z.B. in [MR90], [Car89] und in [HKP91].

Neuronale Netze werden zur Klassifizierung von Mustern verwendet. Die Qualität

einer Realisierung hängt sowohl von der gestellten Aufgabe, als auch von der verwendeten Repräsentation der Eingabemuster bzw. -Daten ab.

Dabei stellt sich häufig die Frage nach der Interpretation des Ergebnisses. Neuronale Netze können die Lösung bestimmter Problemklassen erlernen. Das Ergebnis dieses Lernvorganges ist eine Menge sogenannter Gewichte, es handelt sich dabei um reelle Zahlen. Diese Gewichte enthalten im Erfolgsfall die interne Repräsentation einer Problemlösung. Es ist allerdings i. allg. nicht möglich, das „wie" einer solchen Lösung zu verstehen, in vielen Fällen bleibt das Netz eine „Black Box", d. h. die Lösung ist nicht im klassischen Sinn als Algorithmus übersetzbar. Eine sehr gute Darstellung dieser Fragestellung findet sich in [TP89].

Ein sehr häufiger Fall ist die Lösung einer Aufgabe, deren analytischer Hintergrund entweder unbekannt, oder aber zu komplex für eine praktikable algorithmische Lösung ist. Ein einfaches Beispiel für den ersten Fall stellt die Arbeit [HS93] dar.

Grundsätzlich werden zwei Grundklassen von Neuronalen Netzen unterschieden, nämlich:

- **Supervised Learning:** Für jedes angebotene Lernmuster steht ein zu lernendes Ausgabemuster zur Verfügung. Es wird gewissermaßen „trainiert", daher der Begriff „Supervised Learning". Ein entscheidendes Problem ist dabei die Auswahl geeigneter Repräsentanten [HP92] für die Trainingsmuster. Diese bestimmen die Robustheit der Erkennung. Dies ist wesentlich, da bei realen Problemstellungen die zu klassifizierenden Merkmalsräume durch Anwesenheit von Störungen i. allg. überlappen, der beste Ausweg ist dann die Auswahl von nichtüberlappenden Repräsentanten [GWCK92].
- **Unsupervised Learning:** Netzwerke, die nach diesem Prinzip lernen, benutzen Redundanzen in den angebotenen Eingabemustern, um eine Klasseneinteilung der Eingangsmuster zu erzielen. Ein wichtiges Beispiel ist die Erkennung orthogonaler Merkmale durch Merkmalsdetektoren, vgl. auch [RS90], [EOS92b] und [EOS92a]. Die genannten Artikel beschreiben Kohonen-Netze, auch bekannt als „self-organizing feature maps".

Für die auftretende Problemstellung soll mit „Supervised Learning" gearbeitet werden. Somit sind grundsätzlich mehrere Netzwerktypen möglich, dazu zählen:

1. **Netz als assoziativer Speicher (Hopfield-Modell)** Dient zum Speichern von Eingabemustern. Die gelernten Muster sind darstellbar als Minima eines Energiegebirges. Für ein Eingabemuster liefert das Netz das nächstgelegene Minimum als Ergebnis.
2. **Feed - Forward Netz**

Das „Arbeitspferd“ für praktische Anwendungen, eine Darstellung im Kontext der Mustererkennung findet sich in [KK93]. Die Eingangssignale werden durch Schichten von Neuronen weitergeleitet. Das Verhalten der einzelnen Neuronen wird bestimmt durch eine endliche Anzahl reeller Parameter, den *Gewichten* des Neurons. Die Antwort des Gesamtsystems auf ein Eingabemuster wird bestimmt durch das Verhalten der individuellen Neuronen. Wie oben erwähnt, findet sich in [TP89] ein Modell zur Frage der Interpretation der Gewichte.

3. **Boltzmann-Maschine**

   Der Hauptunterschied der Netze vom Typ *Boltzmann-Maschine* liegt im Verhalten der Neuronen, aus denen das Netz besteht. Das Schaltverhalten dieser Neuronen ist nichtdeterministisch. Für jedes Neuron gibt es eine Wahrscheinlichkeit, einen von zwei möglichen Zuständen anzunehmen. Diese Wahrscheinlichkeiten hängen ab von einem Parameter „Temperatur„ , wobei für den Wert Null der Übergang jedes einzelnen Neurons des Systems exakt definiert ist. Ein physikalisches Analogon für solche Systeme sind *Spingläser*. Die Netze dieses Typs können als Erweiterung von Hopfield Netzen auf Netze mit „hidden units“ betrachtet werden.

Als zusätzliche Randbedingung der gestellten Aufgabe kommt hinzu, daß die Eingabemuster unabhängig von Rotationen, Translationen und Skalierungen erkannt werden sollen. Dies ist bei natürlichen Vorbildern hervorragend realisiert. Ein Buchstabe beispielsweise wird von einem Menschen unabhängig von Lage und Größe erkannt. Biologische Untersuchungen zeigen, daß wesentliche Teile der dazu notwendigen Verarbeitung bereits in der Retina des menschlichen Auges stattfinden. Für eine praktische Implementierung gibt es mehrere Möglichkeiten, die später genauer beschrieben werden.

Zum einen kann ein geeigneter Präprozessor verwendet werden, der die zu analysierenden Grauwertbilder in transformationsinvariante Größen umsetzt, wobei diese Invarianten dann als Eingaben für ein neuronales Netz benützt werden. Dies ist grundsätzlich immer möglich, wobei wesentliche Teile der Problemlösung auf die Wahl eines geeigneten Präprozessors verlagert werden.

Als zweite Möglichkeit bietet es sich an, die Topologie des Netzes so zu wählen, daß von vorneherein eine invariante Erkennung gewährleistet ist. Dies ist der in dieser Arbeit vorgeschlagene Weg.

Schließlich kann das Problem auch dem Netz selber überlassen bleiben, wodurch allerdings die Klassifikationsaufgabe schwieriger wird. Ein Beispiel einer solchen Realisierung stellt [PBC92] dar.

2.2.2 Kantenextraktion

Das Problem der Kantenextraktion soll wie allgemein üblich durch ein Filter realisiert werden. Das Prinzip zeigt Abb. 2.1. Dabei wird jeweils ein $n \times n$- Ausschnitt an allen Koordinaten (x, y) des zu bearbeitenden Grauwertbildes betrachtet. Der eigentliche Filter bestimmt nun eine Wahrscheinlichkeit $\mathbf{P}(x, y)$, ob sich an dieser Stelle ein Punkt einer Kante befindet. Dies ist als Schritt (1) in Bild 2.1 dargestellt. Beispiele solcher klassischen Filter sind Median, Gradient und Sobel - Filter. Das neuronale Netz soll eingesetzt werden, um einen solchen Filter zu realisieren.

Im einfachsten Fall wird dann im zweiten Schritt durch Schwellwertbildung mit Schwelle P_0 , $0 < P_0 < 1$, ein Kantenbild erzeugt:

$$\mathbf{K}(x, y) = \begin{cases} 0 & ,\mathbf{P}(x, y) < P_0 \\ 1 & ,\mathbf{P}(x, y) \geq P_0 \end{cases} \tag{2.1}$$

Eine verbesserte Methode berücksichtigt bei der Schwellwertbildung die Kantenwahrscheinlichkeit der Nachbarpixel. Dabei wird ein bestimmtes Kantenmodell zugrunde gelegt, um iterativ die Kantenwahrscheinlichkeiten anhand der Ausgangswahrscheinlichkeiten einer Nachbarschaft des betreffenden Pixels zu berechnen.

2.3 Aufbau des Neuronalen Filters

2.3.1 Prinzipieller Aufbau

Bild 2.2 zeigt eine Realisierung für den ersten Schritt in Abb. 2.1 mittels eines neuronalen Netzes.

Diese Realisierung enthält keinerlei Elemente einer invarianten Erkennung, das dargestellte Netz müßte die Invarianzen durch Training erlernen. Allgemein wird ein $n \times n$ Ausschnitt des Grauwertbildes betrachtet, im Beispiel $n = 3$. Für die Grauwertmatrix S wird folgende Konvention verwendet:

$$S_{min} \leq S_{ij} \leq S_{max}; 1 \leq i, j \leq n; S_{ij}, S_{min}, S_{max} \in \mathbb{Z} \tag{2.2}$$

$[S_{min}, S_{max}]$ ist dabei das Grauwertintervall, typischerweise ist $S_{min} = 0, S_{max} = 255$. Wir betrachten nun im Folgenden ein einzelnes Neuron mit n afferenten (einlaufenden) Signalen $\xi_i, 1 \leq i \leq n$:

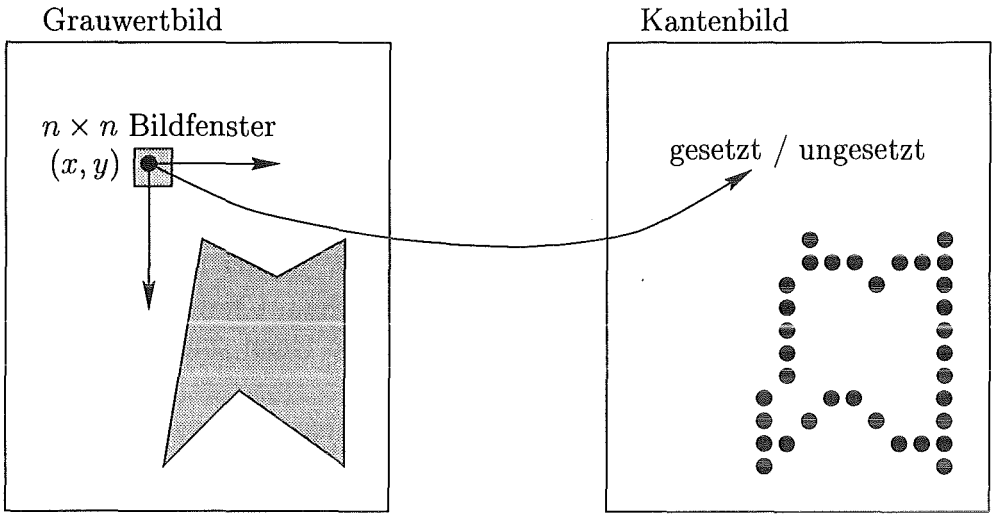


Abbildung 2.1: Das rechte Pixelbild wird durch Abscannen des gesamten Grauwertbildes gewonnen. Das Wertepaar (x, y) kennzeichnet ein Pixel des originalen Grauwertbildes. Es soll ermittelt werden, ob an dieser Stelle eine Kante vorhanden ist, oder nicht. Dazu wird eine Umgebung von (x, y) definiert, die hier als grau schattiertes Quadrat eingezeichnet ist. Aus den Grauwerten dieser Umgebung wird eine Wahrscheinlichkeit für die Zugehörigkeit zu einer Kante ermittelt. Dies kann im einfachsten Fall ein Operator sein, welcher die Differenz zwischen dem obersten und dem niedrigsten in der Umgebung auftretenden Grauwert ermittelt. Ein solcher Operator liefert für jeden Bildpunkt (x, y) des Grauwertbildes eine Wahrscheinlichkeit $\mathbf{P}(x, y)$ für das Auftreten einer Kante. Durch Schwellwertbildung erhält man eine Funktion $\mathbf{K}(x, y)$, die 0 ist für Werte unterhalb und 1 für Werte oberhalb einer gewählten Schwelle p_0. Die Problematik der Wahl einer geeigneten Schwelle wird in Abb. 3.6 deutlich.

Der typische Ansatz für das Verhalten eines solchen Neurons mit Gewichten ω ist zunächst:

$$O_k = \mathrm{f}\left(\sum_{i=1}^{n} \xi_i w_{ki}\right) \tag{2.3}$$

f ist dabei eine sigmoidale Funktion, ðf(x) ist monoton wachsend und strebt für $|x| \to \infty$ den Werten -1, bzw.1 zu. Beispiele für eine solche Funktion sind $\arctan(x)$ und $\tanh(x)$, wie in Abb. 3.9 dargestellt. Die ξ_i sind die Eingangssignale und die w_{ki} die Gewichte des Neurons mit Index k. Durch die w_{ki} wird das Verhalten des Neurons parametrisiert. Man kann auch sagen, daß die w_i die Fähigkeit des Netzes zur Erkennung repräsentieren.

Gleichung 2.3 definiert die Signalverarbeitung eines in Abb. 2.3 skizzierten Neurons. Es ist prinzipiell möglich, mit der in Abb. 2.2 dargestellten Architektur einen Filter zu realisieren. Dieser ist aber nicht a priori invariant bezüglich bestimmter Transformationen, wie Rotation, Translation, Skalierung und Spiegelung. Das

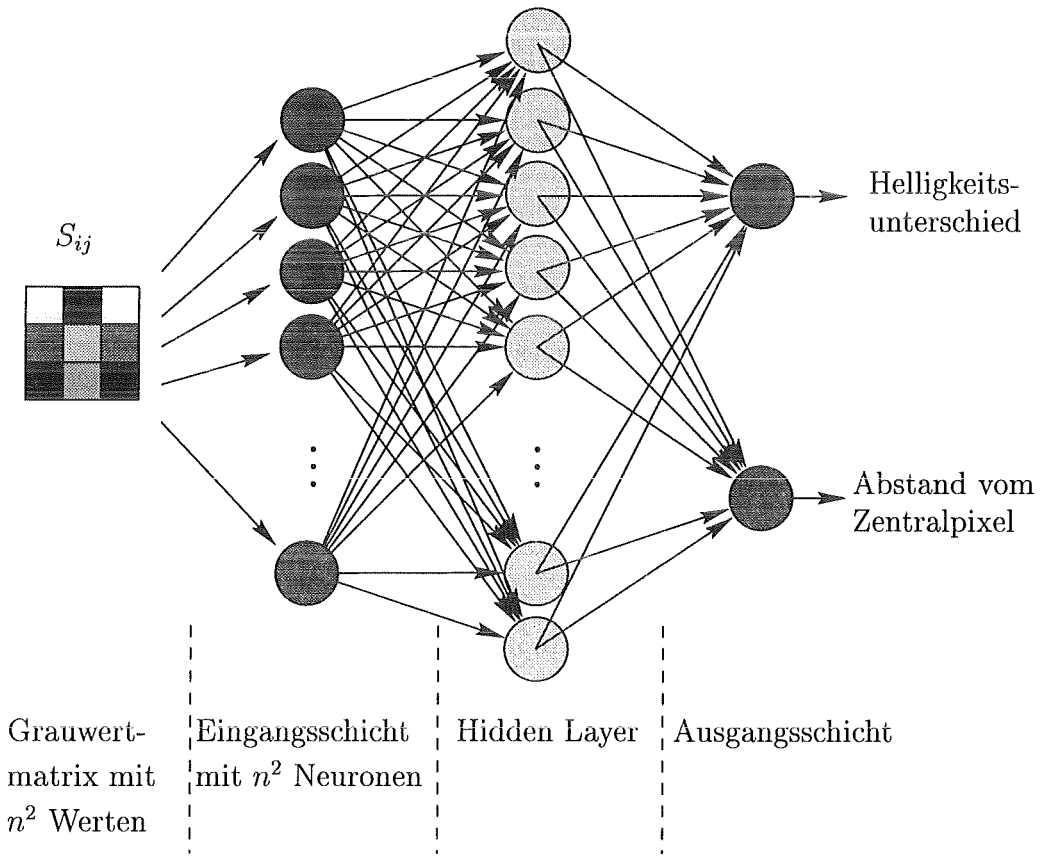


Abbildung 2.2: Dies ist ein prinzipiell möglicher Aufbau eines neuronalen Kantenfilters. Die Matrix S_{ij} der Grauwerte eines quadratischen Bildausschnittes der Breite n mit $1 \leq i, j \leq n$ wie in Abb. 2.1 dienen als Eingangswerte eines neuronalen Netzes. Für den Ausschnitt der Breite n sind dies n^2 Grauwerte. Als Ausgangssignal liefert das Netz ein Maß für den Helligkeitsunterschied, sowie den Abstand vom Zentralpixel des betrachteten Bildausschnittes.

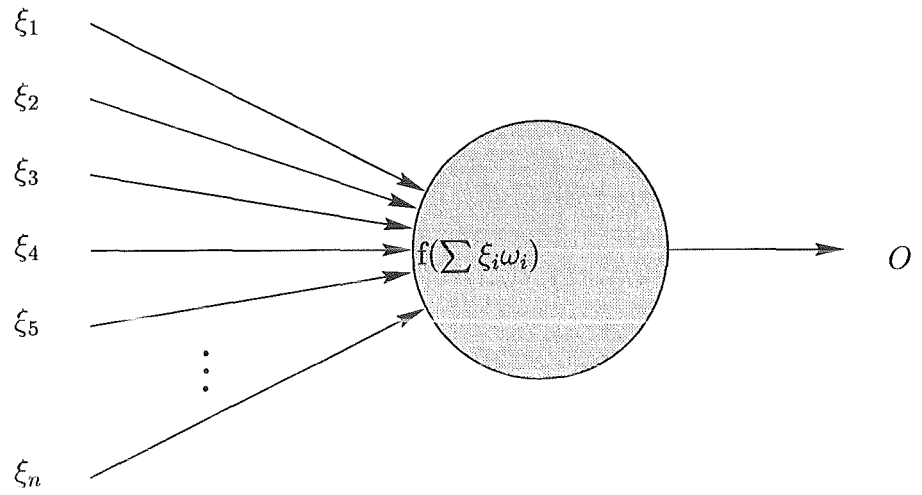


Abbildung 2.3: Ein einzelnes Neuron mit n Eingangssignalen. Die Eingangssignale ξ_i werden linear gewichtet. Das Ausgangssignal ist eine Funktion dieser gewichteten Summe.

verwendete Netz könnte anhand geeigneter Probemuster lernen, Kanten invariant bezüglich dieser Transformationen zu erkennen.

Eine zweite Kritik betrifft das einzelne Neuron. Es zeigt sich, daß mit dem einfachen Ansatz aus Gleichung 2.3 zumindest das einzelne Neuron nicht dazu verwendet werden kann, Korrelationen der afferenten Signale zu bilden. Dies ist der Grund dafür, daß man in diesem Fall Hidden Layer braucht, um eine invariante Erkennung zu ermöglichen. Ein Ausweg stellt die Modifikation von 2.3 dar:

$$\mathbf{O}_k = \mathrm{f}\left(\sum_i \overset{(1)}{w}_{k;i}\, \xi_i + \sum_{i_1<i_2} \overset{(2)}{w}_{k;i_1 i_2}\, \xi_{i_1}\xi_{i_2} + \ldots + \sum_{i_1<\ldots<i_n} \overset{(N)}{w}_{k;i_1\ldots i_n}\, \xi_{i_1} \cdot \ldots \cdot \xi_{i_n} \right) \tag{2.4}$$

Hier treten nun Faktoren $\overset{(N)}{w}_{k;i_1\ldots i_n}$ auf, welche jetzt als Gewichte für Produktterme von N Eingangssignalen ξ_i fungieren. Die Indizes in Klammern über den Gewichten w geben die Anzahl der Faktoren an, aus denen die Produkte bestehen.

Durch geeignete Wahl der $\overset{(\alpha)}{w}_{i_1\ldots i_k}$ können ganz bestimmte Korrelationen unter den Eingangsmustern ausgewählt werden. Betrachtet man beispielsweise einen Term der Form $\overset{(2)}{w}\, \xi_1\xi_2$, so kann das Signal ξ_1 in der zweiten Summe von Gleichung 2.4 nur dann einen Beitrag liefern, wenn ξ_2 von Null verschieden ist.

Neuronen des Typs Gleichung 2.4 wurden zuerst in [RM88] beschrieben, und dort als Sigma-Pi units bezeichnet[†].

[†]Die Bezeichnung hat folgenden Grund: Gleichung 2.4 kann man mit Multiindizes in der

2.3.2 Übergang zu invarianter Erkennung

Eine Möglichkeit zur rotations-, translations- und skalierungsinvarianten Kantenerkennung wird in [RS89] beschrieben. Auf eine Lösung mittels eines neuronalen Netzes bezogen, würde dies die Verwendung eines Präprozessors implizieren. Das Netz würde dann ein gefiltertes Bild analysieren, welches die geforderten Invarianzen bereits besitzt. Die Problematik der Anwendung von Präprozessoren ist, daß sie einerseits rechenintensiv ist, und andererseits das Erkennungsverfahren für überlagertes Rauschen anfällig wird.

Ein zweiter Weg überläßt die gesamte Klassifikation dem neuronalen Netz. Das Netz muß in diesem Fall sowohl das Signal als auch die geforderten Invarianzen lernen, was in der Praxis zu Schwierigkeiten führt, und zudem wieder eine längere Trainingsphase bedeutet.

Ein weiterer Weg führt auf Einschränkungen für die Gewichtsfaktoren $\overset{(k)}{w}_{i_1 \ldots i_k}$. Um eine einfache Darstellung zu erhalten, betrachten wir den Fall eines kontinuierlichen Bildes $\mathbf{S}(x, y)$. Als Kurzform für die Koordinaten verwenden wir $\mathbf{r} = (x, y)$.Im kontinuierlichen Fall können sowohl Randprobleme, als auch Diskretisierungsprobleme unbeachtet bleiben. Die festen Indizes k, i in Gleichung 2.4 gehen dann über in kontinuierliche Variablen $u, \mathbf{r}$, die Summen werden durch Integrale ersetzt:

$$\begin{aligned} \mathbf{O}(u) &= \mathrm{f}\left(\int d^2\mathbf{r}\, \overset{(1)}{w}(u;\mathbf{r})\mathbf{S}(\mathbf{r}) \right. \\ &+ \left. \int d^2\mathbf{r}_1 \int_{|\mathbf{r}_1|<|\mathbf{r}_2|} d^2\mathbf{r}_2\, \overset{(2)}{w}(u;\mathbf{r}_1,\mathbf{r}_2)\mathbf{S}(\mathbf{r}_1)\mathbf{S}(\mathbf{r}_2) + \ldots \right) \end{aligned} \tag{2.5}$$

Wir betrachten nun eine Transformation T der Koordinaten $\mathbf{r}$, und bezeichnen diese mit $\mathbf{r}' = \mathbf{T}(\mathbf{r})$. Dann transformiert sich Gleichung 2.5 zu:

$$\begin{aligned} \mathbf{O}(u) &= \mathrm{f}\left(\int d^2\mathbf{r}'\, w(u;\mathbf{r}')\mathbf{S}'(\mathbf{r}') + \ldots \right) \\ &= \mathrm{f}\left(\int d^2\mathbf{r}\, w(u;\mathbf{T}(\mathbf{r}))\mathbf{J}(\mathbf{r})\mathbf{S}(\mathbf{r}) + \ldots \right) \\ &\overset{!}{=} \mathrm{f}\left(\int d^2\mathbf{r}\, w(u;\mathbf{r})\mathbf{S}(\mathbf{r}) + \ldots \right) \end{aligned} \tag{2.6}$$

Form
$\mathbf{O}_k = \mathrm{f}\left(\sum_{|\alpha|=k} \overset{(k)}{w}_{k;\alpha} \prod_{p=1}^{k} \xi_p \right)$ schreiben, eben in der Folge Sigma-Pi

Die Forderung der letzten Gleichheit in Gleichung 2.6 ergibt sich gerade aus der gewünschten Transformationsinvarianz. $J(\mathbf{r})$ bedeutet dabei den Betrag der Funktionaldeterminante der Transformation T. Da diese Gleichung für eine beliebige Funktion $S(\mathbf{r})$ gelten muß, erhalten wir Constraints für die Gewichte $w(u; \mathbf{r}_1, \ldots, \mathbf{r}_n)$:

$$\mathbf{J}(\mathbf{r}_1) \cdot \ldots \cdot \mathbf{J}(\mathbf{r}_n) w(u; T(\mathbf{r}_1), \ldots, T(\mathbf{r}_n)) \stackrel{!}{=} w(u; \mathbf{r}_1, \ldots, \mathbf{r}_n) \tag{2.7}$$

Für Translationen und Rotationen ist $\mathbf{J}(\mathbf{r}) \equiv 1$, für Skalierungen $\mathbf{r}' = \lambda \mathbf{r}$ gilt $\mathbf{J}(\mathbf{r}) = \lambda^2$. Im mathematischen Anhang findet sich eine detailliertere Betrachtung über die Realisierbarkeit dieser Constraints. An dieser Stelle sei nur ein Beispiel angefügt, nämlich eine Translation der räumlichen Koordinaten um den Vektor $\mathbf{a}$, also $\mathbf{r}' = \mathbf{r} + \mathbf{a}$. $\mathbf{J}$ ist in diesem Fall identisch eins, daher:

$$\begin{aligned} w(u; \mathbf{r}_1, \mathbf{r}_2) &= w(u; \mathbf{r}_1 + \mathbf{a}, \mathbf{r}_2 + \mathbf{a}) \\ \Longrightarrow w(u; \mathbf{r}_1, \mathbf{r}_2) &= \Phi_u(\mathbf{r}_2 - \mathbf{r}_1). \end{aligned} \tag{2.8}$$

Φ_u ist dabei eine beliebige Funktion auf dem $\mathbb{R}^2$. Gleichung 2.8 stellt die allgemeinste Form einer translationsinvarianten Wichtungsfunktion dar. Eine Ableitung für andere wesentliche Transformationsarten findet sich im Anhang.

2.3.3 Diskrete Realisierung

Es soll nun eine konkrete Realisierung eines neuronalen Filters mit den beschriebenen Invarianzen dargestellt werden. Die Notation bezieht sich auf Gleichung 2.2. Als generelle Konvention in dieser Darstellung gilt, daß die Ausgangssignale bzw. die Gewichte des i- ten Layers mit

$$\overset{(i)}{\xi}_k, \qquad \overset{(i)}{w}_k \text{ für } 1 \leq k \leq N_i \tag{2.9}$$

bezeichnet werden, wobei N_i die Zahl der im i-ten Layer vorhandenen Neuronen darstellt.

Die Eingangsneuronen haben lediglich die Aufgabe der Signalweiterleitung, die Ausgangssignale des ersten Layers werden gemäß Gleichung 2.9 mit

$$\overset{(1)}{\xi}_i, \qquad 1 \leq i \leq n^2 \tag{2.10}$$

indiziert, d.h. jedes Eingangsneuron ist mit genau einem Pixel des quadratischen Bildfensters verbunden. Für die Gewichte des nun folgenden Hidden Layers gelten Constraints entsprechend den geforderten Invarianzeigenschaften. Im Falle einer

gleichzeitigen Invarianz bezüglich Translationen, Rotationen und Skalierungen findet man im Anhang in Gleichung A.11 die Einschränkungen für die Gewichte. Daraus ergibt sich unmittelbar, daß in Gleichung 2.4 nur Produkte mit mehr als zwei Faktoren beitragen, da alle anderen Gewichtsfaktoren konstant sind. Wir erhalten somit als einfachsten nichttrivialen Ansatz für die Ausgangssignale des ersten Hidden Layers:

$$\overset{(2)}{\xi}_p = \mathrm{f}\left(c(\overset{(1)}{\xi}) + \sum_{1\le i<j<k\le n^2} \overset{(2)}{w}_{p;ijk}\, \overset{(1)}{\xi}_i\, \overset{(1)}{\xi}_j\, \overset{(1)}{\xi}_k \right) \tag{2.11}$$

$c(\overset{(1)}{\xi})$ steht hier für die ersten beiden Summenterme in Gleichung 2.4 mit einem bzw. zwei Faktoren ξ_i. Da diese jeweils konstantes Gewicht w haben, enthält c genau zwei frei wählbare Gewichte w.

Es ist sinnvoll, für die weitere Darstellung eine bequemere Notation einzuführen. Die Laufindizes i, j und k liegen in $\{1,\ldots,n^2\}$ und erfüllen die Bedingung $i<j<k$. $V_3 \subset \{1,\ldots,n^2\}^3$ sei die Menge dieser Indextripel. Dann können wir Gleichung 2.11 schreiben als:

$$\overset{(2)}{\xi}_p = \mathrm{f}\left(c(\overset{(1)}{\xi}) + \sum_{\alpha\in V_3} \overset{(2)}{w}_{p;\alpha}\, \overset{(1)}{\xi}_{\alpha_1}\, \overset{(1)}{\xi}_{\alpha_2}\, \overset{(1)}{\xi}_{\alpha_3} \right) \tag{2.12}$$

Aufgrund Gleichung A.11 sind die Gewichte w nicht mehr unabhängig voneinander. Sie zerfallen vielmehr in Gruppen mit identischen Elementen. Wir führen daher eine Äquivalenzrelation in V_3 ein, um eine Klassenbildung zu erhalten. Für zwei Indizes $\alpha,\ \beta \in V_3$ gelte $\alpha \sim \beta :\Longleftrightarrow w_{p;\alpha} = w_{p;\beta}$. Diese Definition ist unabhängig vom betrachteten Neuron p. Mit den dadurch definierten Klassen erhalten wir:

$$\begin{aligned} \overset{(2)}{\xi}_p &= \mathrm{f}\Big(c(\overset{(1)}{\xi}) + \sum_{\bar\alpha\subset V_3} \overset{(2)}{w}_{p;\alpha} \underbrace{\sum_{\beta\in\bar\alpha} \overset{(1)}{\xi}_{\beta_1}\, \overset{(1)}{\xi}_{\beta_2}\, \overset{(1)}{\xi}_{\beta_3}}_{:=\Lambda_\alpha} \Big) \\ &= \mathrm{f}\left(c(\overset{(1)}{\xi}) + \sum_{\bar\alpha\subset V_3} \overset{(2)}{w}_{p;\alpha}\, \Lambda_\alpha \right) \end{aligned} \tag{2.13}$$

Diese Form gestattet folgende Interpretation: Aus den Eingangssignalen ξ_i werden zunächst Invarianten Λ_α bezüglich der betrachteten Transformationsgruppe gebildet. Diese Invarianten dienen dann als Eingabesignale für ein Neuronales Netz mit Gewichten $w_{p;\alpha}$.

Dies ist ein wesentliches Ergebnis: Wir erhalten eine Signalvorverarbeitung mit anschließender Erkennung durch ein normales Feed- Forward Netz.

Es soll nun am Beispiel eines 3×3 Ausschnittes gezeigt werden, wie die Klassenbildung erfolgt. Es werden folgende Pixelkoordinaten verwendet:

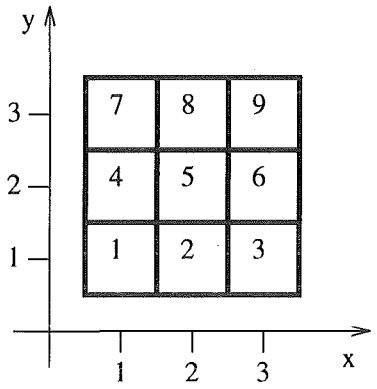


Abbildung 2.4: Wahl der Pixelkoordinaten

Die Nummerierung der einzelnen Pixel entspricht der Nummerierung der Eingangsneuronen. Die folgenden Muster entstehen durch Translation und Rotation und bilden eine der zuvor erwähnten Klassen:

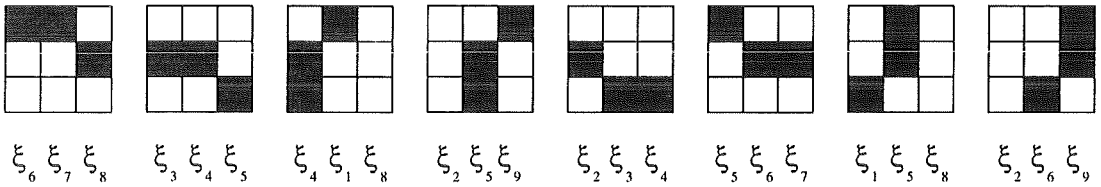


Abbildung 2.5: Beispiel einer Klasse invarianter Muster

Berücksichtigt man noch Spiegelungen, so verdoppelt sich in diesem Beispiel die Zahl der Muster. Insgesamt ergibt sich damit die in Abb. 2.6 dargestellte Realisierung für den neuronalen Filter im Fall eines 3×3 -Fensters.

Die Signalstärken Λ_i werden wie im Beispiel Gleichung 2.5 durch Summation $\xi_6\xi_7\xi_8+\xi_3\xi_4\xi_5+\ldots$ gebildet. Die Anzahl K der Klassen transformationsinvarianter Muster wie in Gleichung 2.5 ist a priori nicht bekannt. Es werden zuerst alle $\binom{n^2}{3}$ möglichen Muster mit drei besetzten Pixeln gebildet. Diese werden dann gemäß Gleichung A.11 in Klassen eingeteilt.

2.4 Ergebnisse

Um die Performance des resultierenden Operators zu bewerten, wurde die von Pratt [Pra78] vorgeschlagene *Figure of Merit* verwendet. Diese lautet:

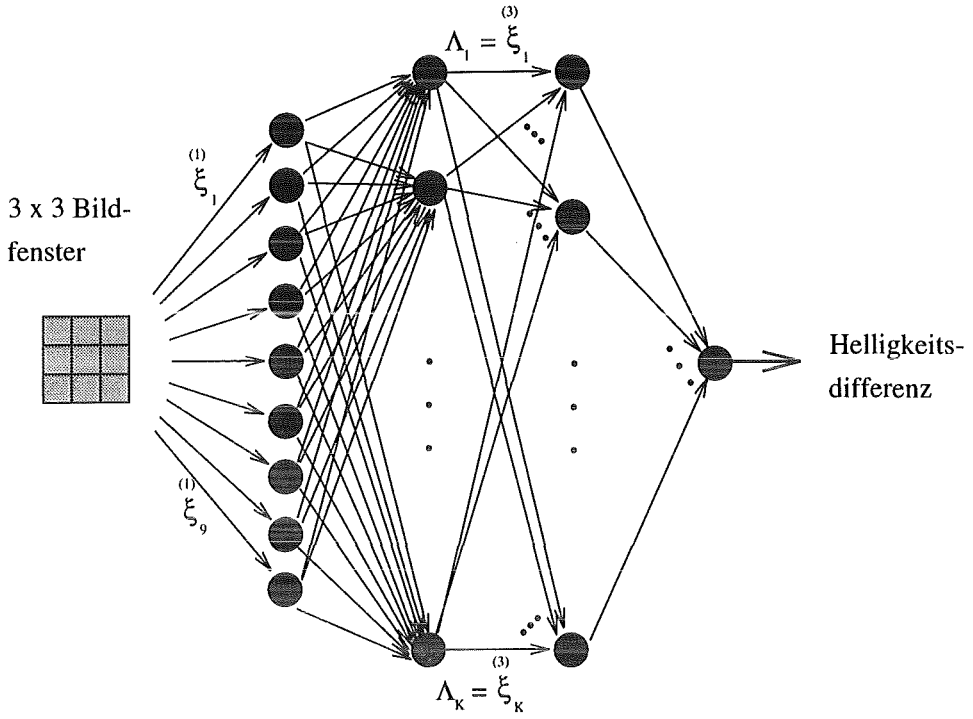


Abbildung 2.6: Gesamtdarstellung des Kantenfilters

$$R = \frac{1}{I_N} \sum_{i=1}^{I_G} \frac{1}{1 + \alpha d_i^2} \tag{2.14}$$

Dabei bedeutet I_G die Anzahl der gefundenen Ecken und d_i die Folge der Distanzen der gefundenen Kantenpositionen von der idealen Kante. I_N bezeichnet das Maximum zwischen I_G und der Anzahl der tatsächlich vorhandenen Kanten des betrachteten Bildes. Das Maß R „bestraft" fehlerhafte Kantenpositionen über die Lokalisierungsfehler d_i, welche typischerweise im Bereich von 1-3 Pixeln liegen. Die Stärke dieses Straftermes kann durch Variation des Skalierungsfaktors α bestimmt werden.

Für einen idealen Kantenoperator gilt $R = 1$. Eine erkannte Kante mit einer globalen Verschiebung von einem Pixel gegenüber der Idealposition ergibt für $\alpha = 1\backslash 9$ den Wert $R = 0.9$.

Entscheidend für einen Kantenoperator ist das Verhalten dieses Qualitätsmaßes in Abhängigkeit vom Signal/Rauschverhältnis SNR. Betrachtet man das Bild einer idealen Kante mit Höhe h und einem Rauschen der Varianz σ, so gilt:

$$SNR = \frac{h^2}{\sigma^2}$$

Abb. 2.7 zeigt den Verlauf von R für den in dieser Arbeit beschriebenen neuronal basierten Operator und drei gängige Kantenoperatoren. Dazu wurden genau dieselben simulierten Bilder betrachtet, mit denen auch des neuronale Netz zuvor trainiert wurde.

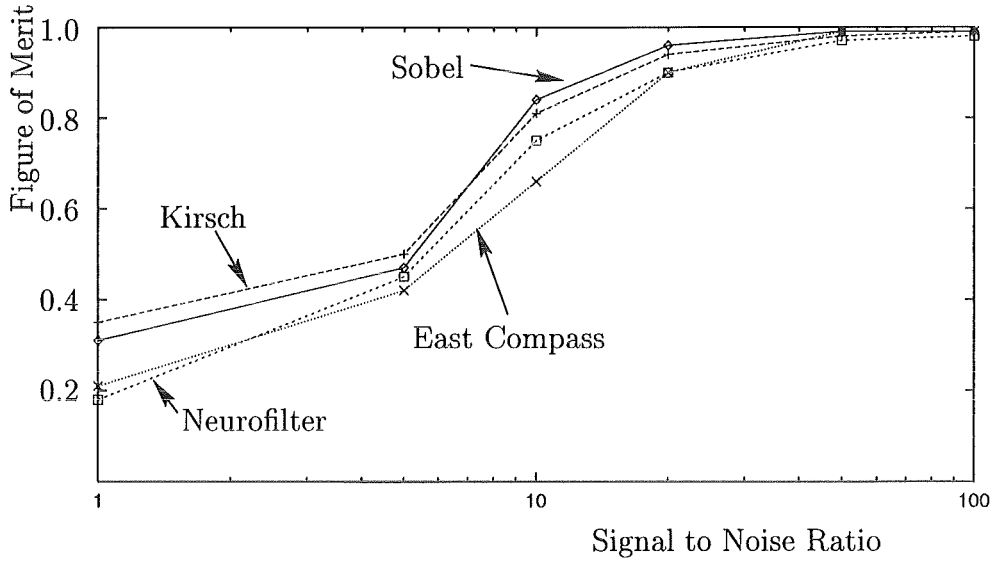


Abbildung 2.7: Das Diagramm zeigt den Vergleich des vorgestellten neuronalen Filters mit einigen klassischen Operatoren, wie sie in der Bildverarbeitung verwendet werden. Man erkennt deutlich, daß keine Verbesserungen auftreten.

Man erkennt die schlechtere Performance gegenüber den Sobel- und Kirschoperatoren.

Die Ursache liegt vermutlich in der begrenzten Größe des Bildausschnittes, den der Operator bearbeitet. Die Zahl der zu betrachtenden Muster mit k Pixeln bei der Fenstergröße n beträgt $\left(\frac{n^2}{k}\right)$. Bei einer Fenstergröße von $n = 5$ ergibt dies etwa $6.8 \cdot 10^7$ zu betrachtende Muster mit 16 besetzten Pixeln. Dies liegt mehrere Größenordnungen über einem durchführbaren Rechenaufwand und ist nicht einmal experimentell durchführbar.

Kapitel 3

Modellbasierte Erkennung

3.1 Generelles zur Kantendetektion

Die modellbasierte Erkennung von Bildobjekten ermöglicht neben einer verbesserten Erkennung gegenüber lokalen, punktbasierten Filterverfahren mit kleinen Umgebungen eine Lokalisierung bis hinein in den Subpixelbereich. Diese kann insbesondere auch bei Anwesenheit von Rauschen erzielt werden. Ermöglicht wird dies auf Kosten eines größeren Rechenzeitaufwandes. Dies kann im Kontext der vorliegenden Aufgabe in Kauf genommen werden, da ja lediglich Teilbereiche eines Grauwertbildes betrachtet werden müssen.

Subpixelauflösung kann grundsätzlich nur dann sicher erreicht werden, wenn eine Modellierung des Bildes existiert. Verschiedene dazu verwendete Verfahren werden in den Übersichtsartikeln [TH86] und [WC90] beschrieben. In [HC92] wird auf die Bedeutung von Subpixelverfahren für die industrielle Inspektion hingewiesen, und ein Verfahren zur Realisierung eines Subpixelkantenfilters gezeigt. Dieses Verfahren macht *keine* Annahme über den betrachteten Intensitätsverlauf, sondern wertet lediglich eine aus den Ableitungen der Intensitätsfunktion gewonnene Wahrscheinlichkeitsdichte aus. Die Autoren verwenden allerdings im experimentellen Teil der Arbeit symmetrische Kantenübergänge, weshalb trotz fehlender Modellbildung gute Ergebnisse erzielt werden. Die Arbeiten von [NB86], [ZH83], [RS92] und [Roh92] enthalten Beispiele parametrischer Intensitätsmodelle zur Beschreibung von Kantenübergängen.

Ein Beispiel für ein Verfahren mittels eines Intensitätsmodells entlang glatter Kurven findet sich in [FL90]. Für das ähnliche Problem der Detektion glatter Kurven mittels eines physikalisch motivierten Modells elastischer „Schläuche“ siehe auch [KWT88].

Wie bereits in der Einleitung erwähnt, spielt die Lokalisierung geometrischer Ob-

jekte im Subpixelbereich vor allem für die dreidimensionale Szenenrekonstruktion, vergl. auch [SB92], eine entscheidende Rolle.

Im Folgenden sollen die entscheidenden Aspekte des bei *COSMOS-2D* verwendeten Bildverarbeitungsverfahrens beschrieben werden, soweit sie für die hier vorgestellten Verfahren relevant sind. Es handelt sich im wesentlichen um Standardverfahren, wie sie generell in der Bildverarbeitung Verwendung finden.

3.1.1 Operatoren zur eindimensionalen Kantendetektion

Der erste Schritt im *COSMOS-2D* System ist die Extraktion der Kanten, d.h. die Erzeugung eines Kantenbildes aus einem Grauwertbild. In [MH80] findet sich eine ausführliche Darstellung der Theorie zur Kantenerkennung, welche zur Konstruktion geeigneter Filter führt. Analoge Verfahren für eine direkte Eckenerkennung werden im Übersichtsartikel [E.R88] beschrieben.

Das Prinzip wird zunächst für den Fall einer eindimensionalen diskreten Intensitätsverteilung in Abb. 3.1 dargestellt.

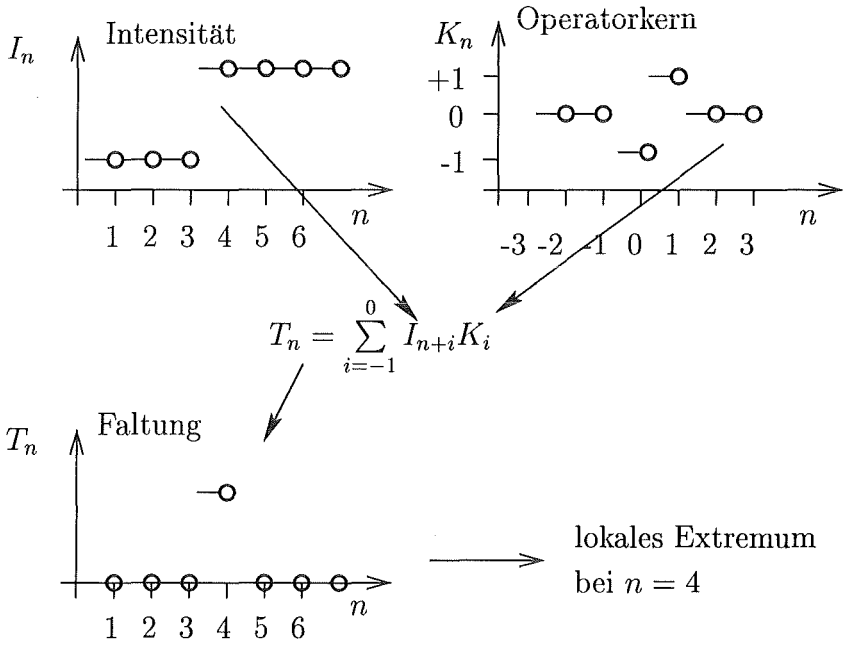


Abbildung 3.1: Beispiel einer eindimensionalen diskreten Kantenextraktion einer Intensitätsfolge I_n. Der Operatorkern K_n wird mit I_N gefaltet. Dadurch erhält man die Folge T_n. Diese weist an der Stelle des Überganges ein Maximum auf, daher kann durch Schwellwertbildung die Koordinate des Zentrums, im Beispiel der Wert 4, erkannt werden.

Der Intensitätsverlauf ist hier gegeben als diskrete Folge von Grauwerten. Der

erste Graph in Abb. 3.1 zeigt eine ideale Stufe. Die Suche nach Kanten erfolgt durch eine Faltung (Konvolution) mit einem Kantenoperator. Dieser lineare Kantenoperator wird definiert durch seinen Kern K_n, im eindimensionalen Fall einer endlichen Folge von reellen Zahlen. Das Ergebnis dieser Faltung hat in diesem Beispiel genau ein absolutes Maximum an der Stelle des Kantenüberganges. Der dargestellte Operator bildet also aus I_n die Folge $T_{n+1} = I_{n+1} - I_n$ der Differenzen aufeinanderfolgender Glieder. Die tatsächliche Berechnung der Ableitung erfolgt meist symmetrisch bezüglich des betrachteten Punktes. Zudem werden i. allg. die Funktionswerte mehrerer benachbarter Punkte dazu verwendet. Auf diese Weise wird eine bessere Näherung für die Ableitung erzielt.

Die so erhaltenen *Extrema* sind Kandidaten für Kantenübergänge. Man beachte, daß ein linearer Anstieg einer Folge I_n zwar eine konstante Folge T_n bedingt, diese aber keine lokalen Extrema aufweist und somit keine Kante detektiert wird, was in Übereinstimmung mit dem Ziel eines solchen Operators ist, da einem linearen Intensitätsanstieg kein Kantenübergang entspricht. Man bezeichnet einen solchen Operator als Differenzenoperator 1. Ordnung. Dies bedeutet im Grenzübergang zum nachfolgend beschriebenen kontinuierlichen Fall den Übergang vom Differenzenoperator 1. Ordnung zur ersten Ableitung. Anders ausgedrückt: Alle Differenzenoperatoren 1. Ordnung konvergieren im Limes gegen die erste Ableitung.

Die Darstellung eines Operators durch seinen Kern ist wesentlich für das Verständnis von Kantenoperatoren in der Bildverarbeitung. Für die dort auftretenden *endlich dimensionalen* Operatoren gilt eine 1:1 Beziehung zwischen einem Operator und seinem Kern. Aus diesem Grund wird in der Literatur zur Bildverarbeitung der Begriff des Operators synonym mit dessen zugehörigem Kern verwendet.

Im kontinuierlichen Fall geht die Variable n über in x, der Operatorkern ist nun eine kontinuierliche Funktion $K(x)$. Die Anwendung des Kantenoperators auf eine Intensitätsfunktion $I(x)$ ist wiederum definiert als deren Faltung mit dem Kern K, oft als $I*K$ geschrieben. Die Faltungsoperation selbst geht von der Summation im endlichen Fall über in eine Integration über die räumliche Variable x. Man erhält eine zu Abb. 3.1 analoge Darstellung:

Die lokalen Extrema von $T(x)$ sind wieder Kandidaten für Kantenübergänge. $T(x)$ entspricht einem Differenzenoperator 1. Ordnung. Die Suche nach lokalen Extrema bedeutet eine Nullstelle in der ersten Ableitung von T, daher entspricht dieses Verfahren insgesamt einer Differenzenbildung zweiter Ordnung und damit der Suche nach einer Nullstelle in der zweiten Ableitung der Intensitätsfunktion.

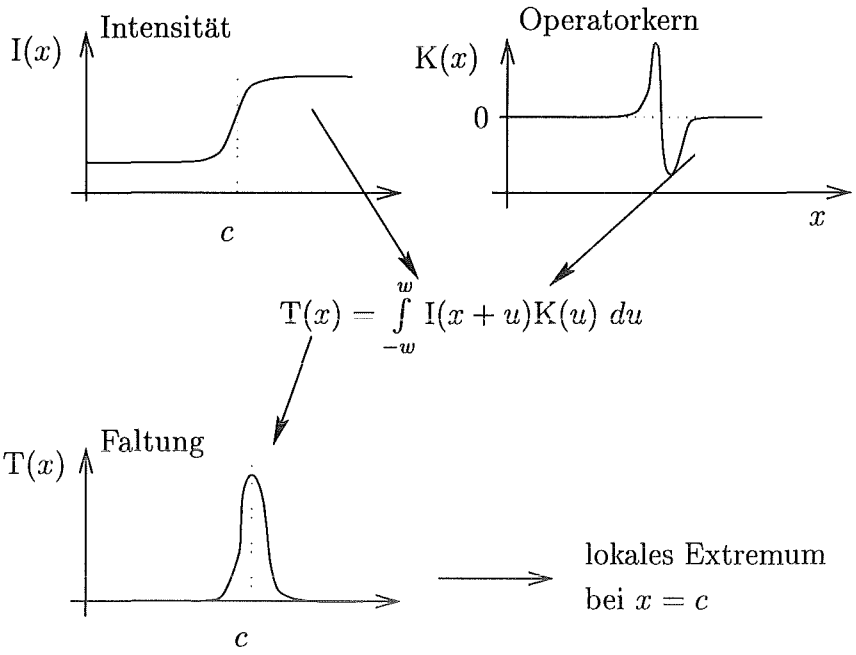


Abbildung 3.2: In Analogie zu Abb. 3.1 geht die Summation der Eingangsintensitäten über eine Folge über in eine Integration über ein Produkt aus einer kontinuierlichen Intensitätsfunktion mit einem ebenfalls kontinuierlichen Operatorkern $K(x)$.

3.1.2 Probleme der Kantendetektion

Das hier für den kontinuierlichen eindimensionalen Fall dargestellte Prinzip unterliegt in der Praxis vielfältigen Einschränkungen. Zunächst einmal treten verschiedene Grundtypen von Kantenübergängen auf, von denen einige in Abb. 3.3 skizziert sind.

Es lassen sich zwei Meßgrößen für den Erfolg des Verfahrens formulieren:

- Geringe Fehlerrate bei der Kantenklassifikation
- Gute Lokalisierung der Kantenposition

Gute Lokalisierung ist notwendig, da eine breite Kante wie in Abb. 3.14 zwar eventuell erkannt wird, aber möglicherweise, etwa bei unsymmetrischen Übergängen, nicht im Zentrum des Überganges.

Es zeigt sich, daß der in Abb. 3.2 auftretende Operatorkern abgestimmt sein muß auf den zu detektierenden Kantentyp, um die zuvor formulierten Bedingungen an den Erfolg eines Kantenoperators möglichst optimal zu erfüllen. In [Can86] wird ein Formalismus beschrieben, der jedem zu detektierenden Kantentyp einen

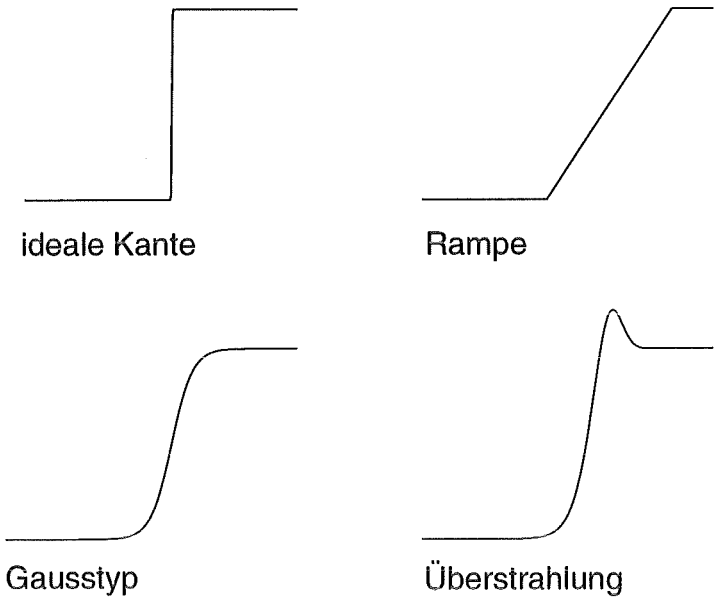


Abbildung 3.3: Hier sind einige typische Arten von Intensitätsübergängen bei Kanten dargestellt. Ein allgemeiner Kantenoperator soll unabhängig von einer bestimmten Klasse solcher Übergänge eine möglichst gute Erkennung erzielen. Bessere Ergebnisse werden erzielt, wenn eine Spezialisierung auf einen bestimmten Kantentyp erfolgt.

Operatorkern zuordnet, um dies zu erreichen. Der Kern K(x) ergibt sich durch Optimierung eines Funktionals unter Nebenbedingungen. Wie bereits in §2 erläutert, gestaltet sich dies in der Praxis aufgrund unzureichender Kenntnis des jeweiligen Typs als schwierig. Kanten werden daher in der Regel suboptimal erkannt. Die Ergebnisse der kontinuierlichen Betrachtungsweise liefern Operatoren für die digitale, also diskrete Bildverarbeitung. Das wichtige Resultat lautet:

Die optimale Detektion einer Kante erfordert einen speziell auf diesen Kantentyp abgestimmten Operator.

3.1.3 Darstellung eines Bildes

In diesem Abschnitt soll die Kantendetektion zweidimensionaler digitaler Bilder beschrieben werden. Die Darstellung eines digitalen Bildes erfolgt als zweidimensionales Feld, die Indizes beziehen sich auf Abb. 3.4.

$\mathbf{P} = \{(i,j),\ 1 \leq i \leq w,\ 1 \leq i \leq h\}$
Dabei bezeichnet w die Breite und h die Höhe des Bildes. Das einzelne Paar (i,j) wird als *Pixel* bezeichnet, $\mathbf{P}$ ist die *Pixelmenge*. Auf der Menge der Pixel sind ver-

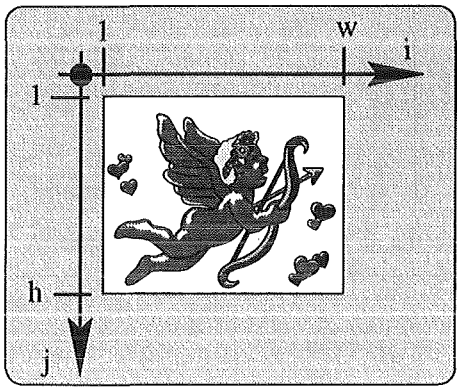


Abbildung 3.4: Pixelindizierung

schiedene Funktionen $f_1, f_2, \ldots, f_n$ definiert. Im Fall eines Farbbildes sind dies drei Funktionen für die Intensitäten der drei Grundfarben. Für die Darstellung eines Grauwertbildes wird genau eine Funktion G benötigt, die jedem Pixel (i, j) einen Grauwert $g_{ij} = G(i, j)$ zuweist. Ein Bild ist also beschrieben als die Pixelmenge zusammen mit den auf dieser Menge definierten Funktionen f_i.

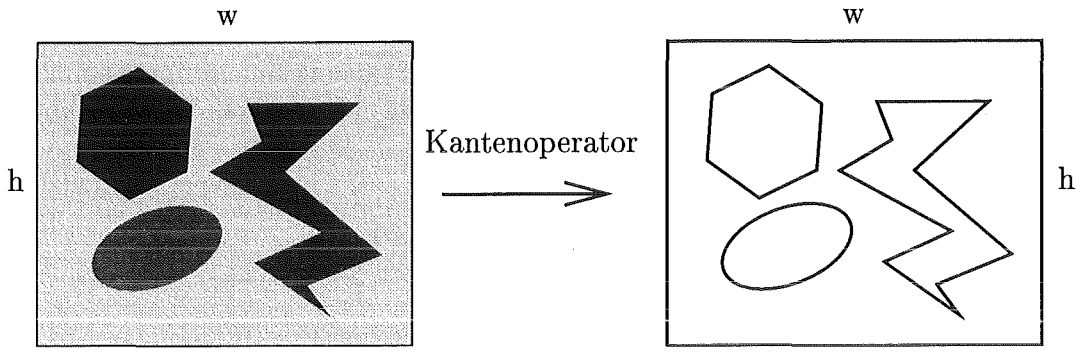
Für die Bilddarstellung wird die in der Bildverarbeitung übliche Konvention verwendet werden, wonach das Pixel $(1, 1)$ die linke obere Ecke und (w, h) die rechte untere Ecke des Bildes P bezeichnet.

Ein *Operator auf dem Bild* ist eine Abbildung, welche die Funktionen $f_1, \ldots, f_n$ als Argumente nimmt und auf eine auf **P** definierte Funktion abbildet. Ein Beispiel für einen solchen Operator ist der (lokale) Kontrast eines Grauwertbildes. In diesem Fall wird die Intensitätsfunktion abgebildet auf die Kontrastfunktion, d.h. diejenige Funktion, welche jedem Pixel den lokalen Kontrastwert zuordnet.

3.1.4 Kantenoperatoren

Im Folgenden werden Kantenoperatoren betrachtet. Als Verallgemeinerung des bereits erläuterten eindimensionalen Falles werden nun zweidimensionale Grauwertübergänge gesucht. Das Prinzip eines Kantenoperators besteht darin, an der Stelle eines solchen Grauwertüberganges einen Wert r und sonst den Wert s zu liefern, etwa $r = 1$, $s = 0$. Entsprechend diesen beiden Zuständen erhält man als Ergebnis ein *Binärbild*. Abb. 3.5 zeigt ein Beispiel für Stufenübergänge.

Um den Speicherbedarf eines Bildes zu kennen, benötigt man dessen *Tiefe*. Die Tiefe eines Bildes ist die Anzahl der Graustufen, die pro Pixel definiert sind. Eine häufig verwendete Tiefe bei Grauwertbildern ist 256. Man spricht dann allerdings von einer Tiefe von 8 Bit. Dies ist auf einem Rechner 1 Byte, damit



Grauwertbild
$g_{ij} \in \{1, \ldots, 2^n\}$
$b \times h \times n$ Bits

Kantenbild K
$k_{ij} \in \{0, 1\}$
maximal $b \times h$ Bits

Abbildung 3.5: Zweidimensionaler Kantenoperator

können dann die $2^8 = 256$ verschiedene Grauwerte dargestellt werden. Das Intervall $[0 \ldots 255]$ bildet dann eine Darstellung für den Grauwertübergang von „schwarz" nach „weiß".

Ein Binärbild kennt hingegen nur zwei Zustände pro Pixel, für das daher ein Bit zur Darstellung auf einem Rechner benötigt wird, daher sprechen wir von einer Farbtiefe 1. Dies bedeutet für ein Grauwertbild der Tiefe n eine Datenreduktion um den Faktor n. Dies gilt zumindest, solange keine Datenkompression von Ausgangs- und Binärbild durchgeführt wird.

Die Beschreibung eines Operators zur Kantenextraktion gliedert sich meist in zwei Stufen. Zunächst wird ein linearer Operator definiert, der ein Maß S für die Stärke eines Kantenüberganges liefert. Dieses Intensitätsmaß wird dann im zweiten Schritt durch einen Abschneideparameter α binarisiert. Liefert der Operator einen Wert größer als α, so wird als Ergebnis ein Kantenpunkt erhalten. Ein Beispiel für einen Kantenoperator definiert der folgende Kern [†]:

$$l_{ij} = \begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}$$

Dies definiert den sogenannten Laplace Operator, die diskrete Version des kontinuierlichen Differentialoperators $\triangle = \frac{\partial^2}{\partial_x^2} + \frac{\partial^2}{\partial_y^2}$. Der Operator selbst wird dann analog zum eindimensionalen Fall durch die Faltung definiert vermöge:

$$s_{qr} = \sum_{i=1}^{3} \sum_{j=1}^{3} g_{q+i,r+j} l_{ij}$$

[†]Fehlende Werte sind identisch 0.

Die Matrix g_{ij} kennzeichnet die Grauwerte, l_{ij} den Kern des zweidimensionalen Kantenoperators. Das Verfahren entspricht der Faltung im eindimensionalen, diskreten Fall wie in Abb. 3.1. Führt man anschließend eine Schwellwertbildung durch, können aus s_{qr} die Kantenpositionen ermittelt werden.

Ausführliche Diskussion verschiedener Operatoren und deren Vor- und Nachteile erfolgt in [RK82] sowie in [HS92].

3.1.5 Kantendetektion

Die bisher vorgestellten Operatoren zur Kantendetektion ermitteln die Kantenwahrscheinlichkeit als lineare Funktion der Intensitätsfunktion. In *COSMOS-2D* werden auch nichtlineare Kantenoperatoren verwendet. Diese sind als Teil einer Spezialhardware zur Kantenbilderzeugung in Videoechtzeit realisiert. Dabei wird *kein* einfaches Schwellwertverfahren zur Binarisierung verwendet, sondern ein auf Relationen von Nachbarschaften beruhender Algorithmus.

Abb. 3.6 zeigt als Beispiel eine 3-D Aufnahme einer sogenannten Wabenstruktur, welche nach dem LIGA - Verfahren hergestellt wurde. Dies ist ein problematischer Fall, da bei einer nicht senkrecht zur Struktur stehenden Aufnahmerichtung zusätzliche Probleme auftreten können. Eines dieser Probleme wird in der Aufnahme deutlich sichtbar. Die Helligkeitsunterschiede zwischen zwei verschiedenen Regionen sind gering, wenn etwa die zugehörigen Flächen der Struktur nur geringe Unterschiede in der Normalenrichtung aufweisen und aus dem selben Material bestehen. Dies kann im 2-D Fall nicht auftreten, da aufgrund der gewählten senkrechten Projektionsrichtung stets Übergänge zwischen dem Substrathintergrund und dem Strukturmaterial auftreten.

Für die Erzeugung des Kantenbildes wurde ein *Sobeloperator* verwendet. Dieser Operator wurde so skaliert, daß seine Ausgangswerte im Intervall $[0, \ldots, 255]$ liegen. Die zwei unterschiedlichen Binärbilder ergeben sich aus zwei verschiedenen Abschneideparametern, nämlich $\alpha = 100$ und $\alpha = 230$.

Dieses Bild verdeutlicht die in *COSMOS-2D* auftretenden Probleme. Die in Abb. 3.6 dargestellte Aufnahme hat eine relativ schlechte Qualität. Die Kontrastunterschiede einiger Kanten sind so gering, daß der Abschneideparameter α relativ klein gewählt werden muß. Dies bedingt dann eine große Zahl von Fehlklassifikationen, da Punkte als Kanten erkannt werden, wo lediglich Texturmerkmale der Wabenwände vorhanden sind. Eine höhere Wahl von α vermindert zwar die Zahl der Fehlklassifikationen, allerdings wird jetzt ein Teil der im Original vorhandenen Ecken nicht erkannt. Dies ist ein grundsätzlich unauflösbares Dilemma eines Kantenoperators bei schlechtem Signal/Rauschverhältnis. Einen analogen Vergleich entsprechender Bilder mit problematischer Wahl des Abschneidepara-

meters und deutlicher Verbesserung durch ein modellbasiertes Verfahren findet man in [FL90].

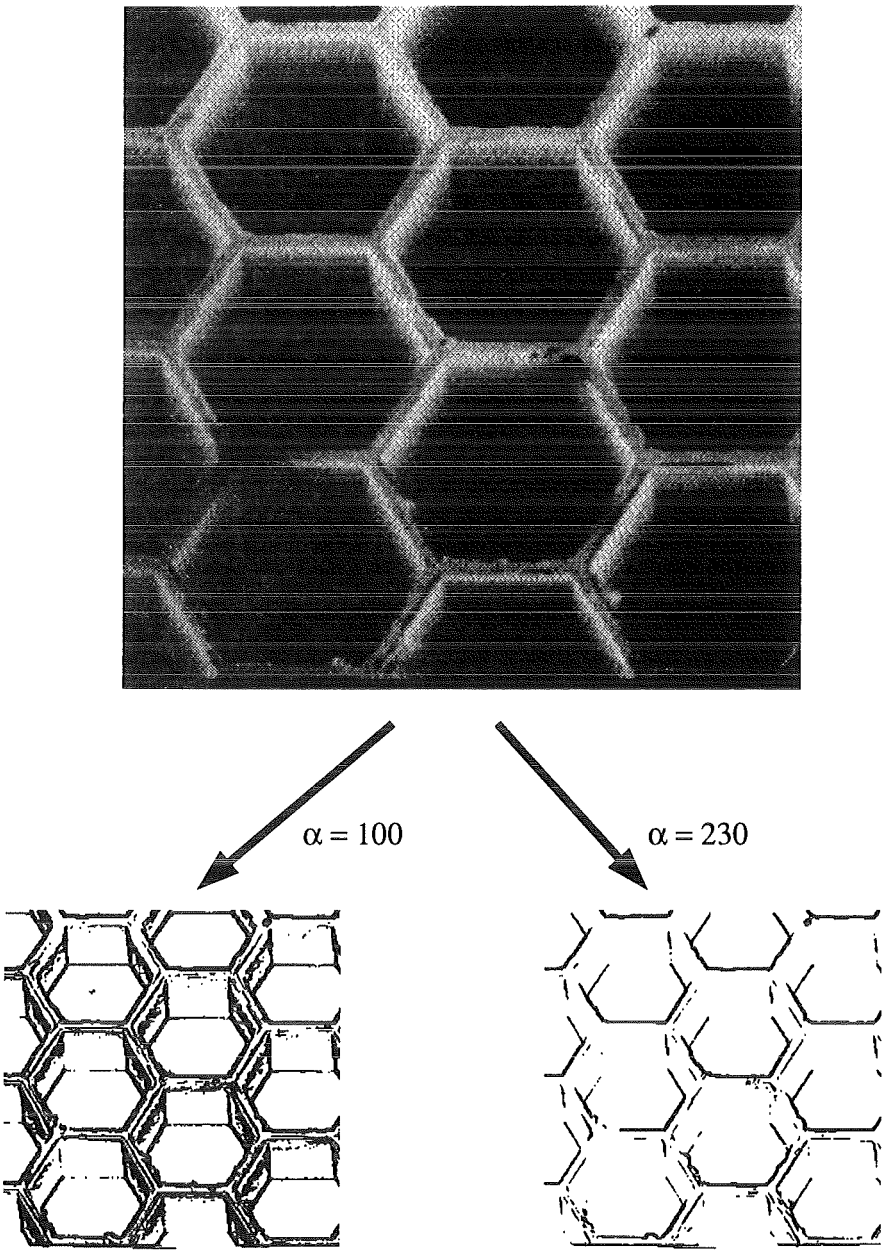
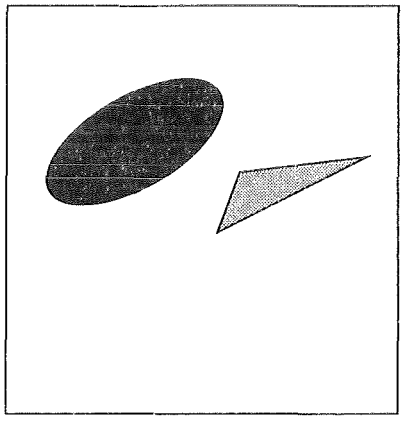


Abbildung 3.6: Binarisierung eines Grauwertbildes für zwei verschiedene Schwellwerte α. Für eine kleine Schwelle werden zwar alle Kantenpunkte erkannt, allerdings steigt die Rate derjenigen erkannten Punkte, welche keine Kantenpunkte sind. Diese verschwinden für eine höher gewählte Schwelle α, allerdings werden jetzt vorhandene *Kantenpunkte nicht erkannt.*

3.2 Modellbasierte Kanten und Eckenerkennung

3.2.1 Generierung einer symbolischen Beschreibung

Das Endergebnis einer symbolischen Szenenbeschreibung sind Objekte mit den ermittelten zugehörigen Attributen. Ein Beispiel zeigt Abb. 3.7. Es werden hier nur die rein geometrischen Parameter idealisierter Objekte dargestellt.



Objekt Dreieck

Attribute:
- Länge1
- Länge2
- Länge3
- Schwerpunkt
- Orientierung
- Grauwert_innen
- .....

Objekt Ellipse

.....

Abbildung 3.7: Das skizzierte Bild zeigt ein Grauwertbild. Dieses enthält eine Szenenbeschreibung durch Grauwerte individueller Pixel. Aus diesem Bild kann man eine symbolische Beschreibung in Form von Bildprimitiven erhalten. Ein solches Bildprimitiv ist ein Objekt mit einer Anzahl interner Attribute, welche die konkrete Ausprägung des Objektes festlegen.

Generell lassen sich zumindest zwei Klassen von Modellparametern eines Objektes unterscheiden, nämlich:

- geometrische Parameter
  - Schwerpunkskoordinate
  - Anfangs- und Endpunkte gerader Kanten
  - Orientierung in der Ebene
  - Kurvenparameter (z.B. Splines, Ellipse, Hyperbel)

  ...
- texturelle Parameter
  - Helligkeit
  - Kantenparameter (siehe Abb. 3.3)
  - Parameter zur Beschreibung texturierter Flächen

  ...

Für das Ziel der Modellbeschreibung eines *realen* Bildes werden beide Parametertypen benötigt. Der CAD - Entwurf ermöglicht in Verbindung mit der bekannten Perspektive auf die Szene die Bestimmung der geometrischen Parameter des Modelles. Für ein zufriedenstellendes Modell werden zumindest noch folgende Informationen benötigt:

- Art der beteiligten Kantenübergänge, vergl. auch Abb. 3.12.
- Intensitäten der beteiligten Regionen.

Unter der Annahme, daß die Regionen *ausschließlich* durch die beteiligten Objekte selbst bestimmt werden, kann dann insgesamt ein erstes Modell der erwarteten Bilder generiert werden.

Abbildung 3.8 zeigt ein einführendes Beispiel. Man erkennt im oberen Teil einen Ausschnitt aus einem real denkbaren Grauwertbild. Es handelt sich um eine Kreisfläche mit den Intensitäten I_1 im Inneren und I_0 außerhalb des Kreises. Im Gegensatz zu einer rein geometrischen Darstellung, etwa auf einem CAD - System, zeigt dieses Bild einen stetigen Intensitätsverlauf zwischen den Werten I_0 und I_1 mit einer Breite b. Im vorliegenden Beispiel wird dieser Übergang durch den Tangenshyperbolicus definiert, also zunächst eindimensional:

Die Breite des Überganges hängt ab von der willkürlichen Definition, ab welcher Sättigung der Übergang als erreicht gilt. Die Funktion $\tanh(x)$ in Abb. 3.9 hat für $x \to \pm\infty$ die Grenzwerte ± 1. Für $x = \pm\frac{b}{2}$ werden 76% dieser Sättigungswerte erreicht. Der willkürlich gewählte Faktor 2 kann durch einen anderen Wert ersetzt

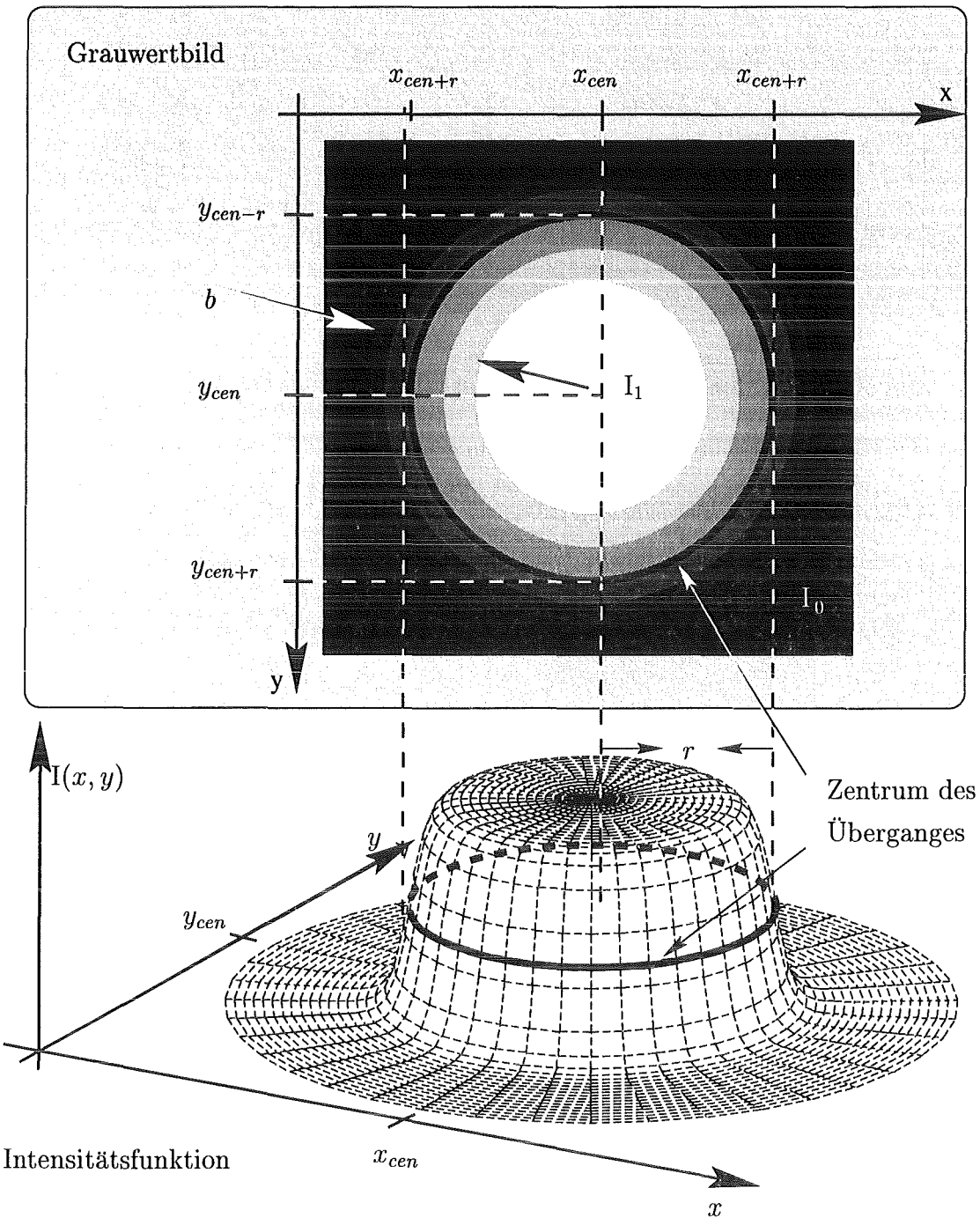


Abbildung 3.8: Intensitätsverlauf eines Kreises. Die dreidimensionale Darstellung im unteren Bildteil zeigt die zweidimensionale Intensitätsfunktion der Grauwerte für einen Kreis mit Radius r. Der Übergang von der niedrigeren Intensität im Außenraum zur höheren Intensität im Inneren des Kreises erfolgt kontinuierlich, d.h. es treten verschiedene Zwischenstufen von Grauwerten auf, wie sie in allen realen Bildern durch unterschiedliche Effekte, beispielsweise Interferenz, verursacht werden. Dies wird im oberen Teil durch konzentrische Ringe unterschiedlicher Intensität dargestellt. Die schwarze Kreislinie kennzeichnet das Übergangszentrum in beiden Teildarstellungen.

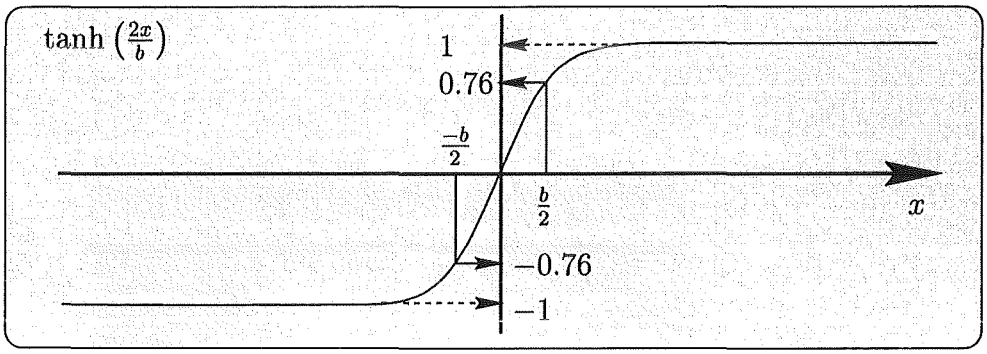


Abbildung 3.9: Dies ist ein Beispiel für eine Übergangsfunktion mit Sättigungswerten. es handelt sich um die Funktion $\tanh(x)$*, welche für* $x \mapsto \pm\infty$ *gegen die Werte* ± 1 *strebt. In dieser Darstellung tritt zusätzlich der Faktor* $\frac{2}{b}$ *auf, welcher die Breite des Überganges bestimmt. Für* $x = \frac{b}{2}$ *erreicht die Funktion 76% ihres Sättigungswertes.*

werden und kennzeichnet den Funktionswert, ab dem der asymptotische Übergang als erreicht gilt. Für $x = \pm\frac{b}{2}$ werden beispielsweise 95% der Sättigungswerte erreicht.

Damit ergibt sich das Modell für die Intensität als Funktion der Koordinaten des Kreises mit Radius r und einem Übergang der Breite b in Abb. 3.8 zu:

$$\mathrm{I}(x,y) = \mathrm{I}_0 + \tfrac{1}{2}(\mathrm{I}_1 - \mathrm{I}_0)\left\{1 + \tanh\left(2\ \frac{r - \sqrt{(x-x_{cen})^2 + (y-y_{cen})^2}}{b}\right)\right\}$$

Dies ist ein sehr *einfaches* parametrisches Modell für ein reales Bild, aus diesem Grund ist die Zahl der enthaltenen Parameter gering. Wir können hier eine Unterscheidung zwischen *geometrischen* und *optischen* Parametern treffen:

1. Drei Geometrische Parameter:
   (a) Kreisradius r
   (b) Zwei Mittelpunktskoordinaten x_{cen} und y_{cen} des Kreises
2. Drei Optische Parameter:
   (a) Zwei Intensitäten I_0 und I_1
   (b) Die Breite b des Kantenüberganges

Die Breite b definiert insofern einen *optischen* und nicht etwa einen geometrischen Parameter des Modells, als sie einen Teil der Beschreibung des Intensitätsverlaufs darstellt.

Es wird sich später herausstellen, daß dieses Modell in zweifacher Hinsicht unzureichend ist. Es berücksichtigt nur eine Art von Kantenübergang. Ferner werden keine texturellen Merkmale erfaßt, wie sie in Kapitel §4 behandelt werden.

3.3 Erkennung geometrischer Primitive in realen Bildern

Das Ziel der subsymbolischen Erkennung sind Linien und Kanten. In [Wei89] wird dies am Beispiel der Linienerkennung mittels linearer Regression in einem verrauschten Pixelbild gezeigt. Arbeiten zur Vertexerkennung finden sich in [DG91]. Da Ecken sehr viel schwieriger zu detektieren sind als Kanten, wird ein Modell einer Ecke benötigt. Zu diesem Zweck ist es zunächst notwendig, die Typen möglicher Ecken zu definieren. In *COSMOS-2D* existieren u.a. die geometrischen Primitive *Gerade* (g), *Kreisbogen*(k). Der Typ *Spline*(s) befindet sich noch in der prototypischen Entwicklung.

Solange die Erkennung auf *senkrechte* Projektion der Szene beschränkt bleibt, treten keine zusätzlichen Primitive auf. Bei schräger Projektion, wie sie bei einem 3-D System unvermeidbar wird, treten zusätzlich noch Ellipsenbögen auf, die als Kegelschnitte aus Kreisbögen entstehen. Dies wird noch dadurch erschwert, daß entsprechend der gewählten Blickrichtung Teilverdeckungen möglich sind, die bei senkrechter Projektionsrichtung bei *COSMOS-2D* nicht eintreten.

Der Fall einer Ecke ist daher komplizierter als im vorangegangenen Beispiel eines Kreises. Neben den geometrischen und physikalischen Attributen wird zusätzlich eine topologische Klassifikation benötigt. Dies mündet in diskrete Attribute und beinhaltet:

- Die Anzahl der aus dem Eckpunkt auslaufenden Kanten
- Den Typ der jeweiligen Kanten

Abb. 3.10 zeigt zwei Beispiele für denkbare Ecken mit jeweils drei auslaufenden Linien.

Als einfachster Fall soll hier das Beispiel einer Ecke mit drei auslaufenden Geraden betrachtet werden, man betrachte auch die zweidimensionale Darstellung in Abb. 3.11.

Wie bereits im Beispiel eines Kreises gezeigt, entsteht ein einfaches parametrisches Modell einer Ecke, indem die Intensitäten gemäß einer Übergangsfunktion, wie in Abb. 3.12, gewählt werden. Dabei ist dann das zugrundliegende geometrische Modell verschieden.

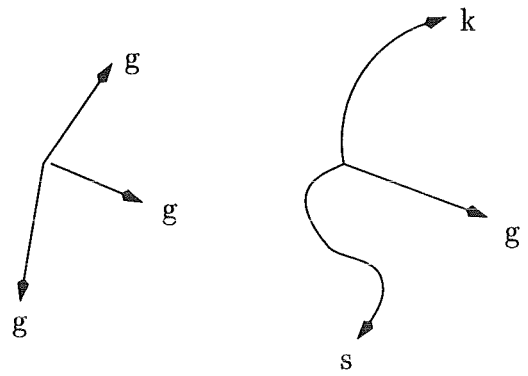


Abbildung 3.10: Beispiele möglicher Eckentypen mit drei auslaufenden Linien

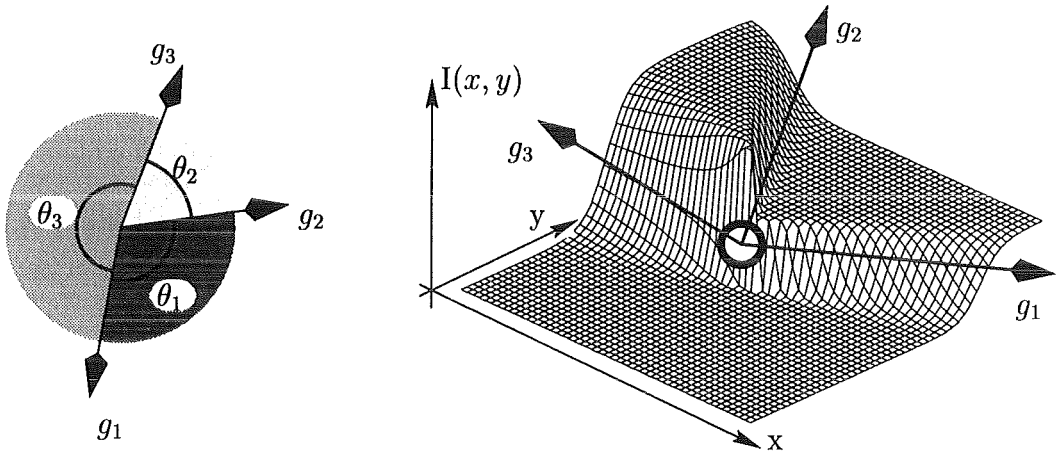
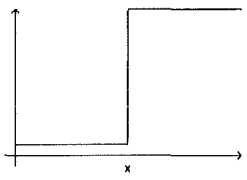
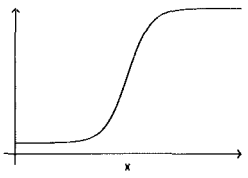


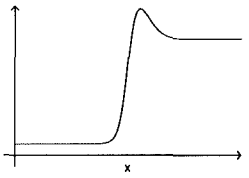
Abbildung 3.11: Darstellung einer Kante mit drei auslaufenden Linien. Der linke Teil zeigt die geometrische Struktur des zweidimensionalen Bildes, wobei im Vergleich mit Abb. 3.8 die Mittelpunktskoordinate, sowie das Koordinatensystem, weggelassen wurden. Der rechte Teil zeigt wieder eine perspektivische Darstellung der Intensitätsfunktion $\mathrm{I}(x, y)$ *als Höhenzug über der x/y -Ebene. Der Kreis im Zentrum markiert die geometrische Position des Übergangszentrums.*

1. 

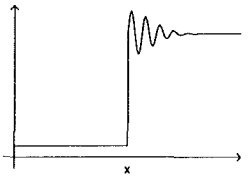
Ideale Kante:
Dies ist der Grenzfall für optische Systeme mit Wellenlänge $\lambda \rightarrow 0$. Es werden keinerlei physikalische Effekte berücksichtigt, daher ist die Intensitätsfunktion ausschließlich durch die Geometrie des Objektes bestimmt.

2. 

Symmetrischer Übergang:
Dies ist eine einfache Erweiterung des vorherigen Modelles. Hier werden bereits physikalische Eigenschaften des Kamerasystems berücksichtigt, wie etwa die endliche Bandbreite in der Signalverarbeitung.

3. 

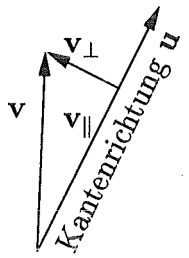
Überstrahlung:
In diesem Beispiel tritt eine Überstrahlung auf, welche typisch ist für optische Systeme.

4. 

Interferenz:
Alle realen Systeme sind gekennzeichnet durch eine endliche Wellenlänge λ des zugrundeliegenden Aufnahmesystems(Elektromagnetisch, Elektronen, Neutronen usw.. Daher sind Kantenübergänge mit Breiten von etwa λ gekennzeichnet durch Interferenzen und Beugungserscheinungen.

Abbildung 3.12: Mögliche Intensitätsfunktionen für Kantenübergänge

Eine Vorschrift zur Konstruktion zeigt Abb. 3.13.



$$v_{\parallel} = \frac{\mathbf{v} \cdot \mathbf{u}}{u}$$

$$\mathrm{I}(\mathbf{v}) = \mathrm{I}_0\left(\frac{\mathbf{v} \cdot \mathbf{u}}{u}\right)$$

Abbildung 3.13: Vorschrift zur Konstruktion der Intensitätsfunktion einer Ecke. Für die Intensität eines Bildpunktes an der Stelle $\mathbf{v}$ *ist der Lagevektor* $\mathbf{v}_\perp$ *maßgeblich. Dieser gibt die Lage und den Abstand vom Kantenzentrum an.*

Im Fall einer einzelnen Kante ist dies trivial, da lediglich der Abstand des jeweiligen Punktes vom Kantenzentrum in die Intensitätsfunktion eingeht. $\mathrm{I}_0(x)$ ist dabei die eindimensionale Intensitätsfunktion senkrecht zu einem Übergang, wie in Abb. 3.12 dargestellt.

3.4 Gewinnung der Parameter

Das Ziel des im letzten Abschnitt beschriebenen parametrischen Modells ist die Beschreibung von Objekten in realen Szenen. Zu diesem Zweck müssen die Parameter so ermittelt werden, daß nach einem bestimmten Kriterium eine optimale Anpassung erzielt wird.

Hierzu existiert ein Standardverfahren, welches die Parameter so bestimmt, daß die Summe der Fehlerquadrate minimiert wird. Es handelt sich dabei um die „Methode der kleinsten Quadrate“, welche im Anhang auf Seite 89 im Detail beschrieben wird. Weitergehende Darstellungen sowie Modifikationen zur Konvergenzbeschleunigung finden sich in [Sto83].

Zu diesem Zweck wird die Differenz der Modell- und Szenenintensitäten für einen bestimmten Szenenausschnitt pixelweise gebildet. Die so erhaltenen Differenzen werden quadriert und aufsummiert. Auf diese Weise ergibt sich eine Fehlerfunktion der auftretenden Parameter:

$$\mathbf{E}(\lambda_1, \ldots, \lambda_n) = \sum_{\nu \in S} [p_\nu - \mathrm{I}_\nu(\lambda_1, \ldots, \lambda_n)]^2$$

Die Menge S wird durch den gewählten Bildausschnitt bestimmt und bedeutet die Menge der betrachteten Pixel. Der Wert n bezeichnet die Anzahl der Parameter des gewählten Modells.

Falls ein (absolutes) Minimum existiert, so ist das Problem definiert. Es besteht dann in Bestimmung des Parametertupels $\lambda_1, \ldots, \lambda_n$, für welches das Minimum angenommen wird.

$$\min_{\lambda_1,\ldots,\lambda_n} \mathbf{E}(\lambda_1, \ldots, \lambda_n)$$

Dies geschieht i. allg. iterativ. Der Erfolg iterativer Verfahren hängt von zwei Faktoren ab:

- Der Fehlerfunktion $\mathbf{E}(\lambda_1, \ldots, \lambda_n)$
- Geeigneten Startwerten für die Parameter $\lambda_1, \ldots, \lambda_n$

Die Fehlerfunktion besitzt i. allg. mehrere lokale Minima. In diesem Fall kann die Iteration, z.B. bei schlechten Startwerten, in einem lokalen Minimum stecken bleiben, was ein sehr häufiges Problem der nichtlinearen Optimierung ist. Ein anderes Problem kann die Konvergenzgeschwindigkeit sein, was durch Zusatzterme verbessert werden kann.

Die Form der Fehlerfunktion kann prinzipiell durch die Wahl der Modellierung beeinflußt werden. Hier ist eine Beschränkung hinsichtlich der Komplexität notwendig. Die Zahl der Parameter darf nicht zu groß gewählt werden, da sonst der Konvergenzbereich so klein wird, daß eine ausreichend genaue Bestimmung der Startwerte nicht mehr zuverlässig erreicht werden kann.

Die Startwerte selbst sind nur teilweise aus dem CAD - Entwurf bekannt. Es handelt sich um jene Parameter, welche den rein geometrischen Anteil des Modells beschreiben. Es stellt sich daher die Frage, wie die *nichtgeometrischen* Startparameter gewählt werden müssen.

Das Modell betrachtet die Szene als eine Menge benachbarter Regionen, welche nahezu konstante Intensitäten besitzen. Im Beispiel von Abb. 3.8 sind dies I_0 und I_1.

Die Parameter für den Kantenübergang bestimmen die konkrete Ausprägung eines bestimmten Kantentyps. Im Beispiel 2von Abb. 2 ist dies ein einziger Parameter für die Breite des Überganges.

Generell kann das Zentrum einer nicht idealen Kante nur dann bestimmt werden, wenn ein Modell vorhanden ist. Dieses Modell kann physikalischer oder auch heuristischer Natur sein. In Abb. 3.14 wird der dritte Fall aus Abb. 3.12 einer Überstrahlung genauer dargestellt. Der weiße Kreis symbolisiert das Zentrum x_0 des Intensitätsübergangs. Dieses Zentrum kann aus einer realen Aufnahme durch Parameteranpassung gefunden werden.

Die Parameter für die Kantenübergänge sind insofern *nicht* wesentlich für die Bestimmung von Startwerten, als das Verfahren der Parameteranpassung auch für das Modell einer idealen Kante bei einer real nicht idealen Kante stets konvergiert. Dabei tritt allerdings in der Regel ein höherer Fehler für die Lokalisierung des Kantenüberganges auf. Der Grund für diese Konvergenz ist die geringe räumliche Ausdehnung des Überganges im Verhältnis zur Größe der nahezu homogenen Regionen der zu untersuchenden Objekte. Daher kann man in erster Näherung alle Grauwertübergänge als ideale Kanten betrachten.

Der durch Wahl einer idealen Kante wie im Beispiel 1 von Abb. 3.12 eintretende Fehler wird deutlich, wenn man das Beispiel 3 aus der selben Abbildung betrachtet.

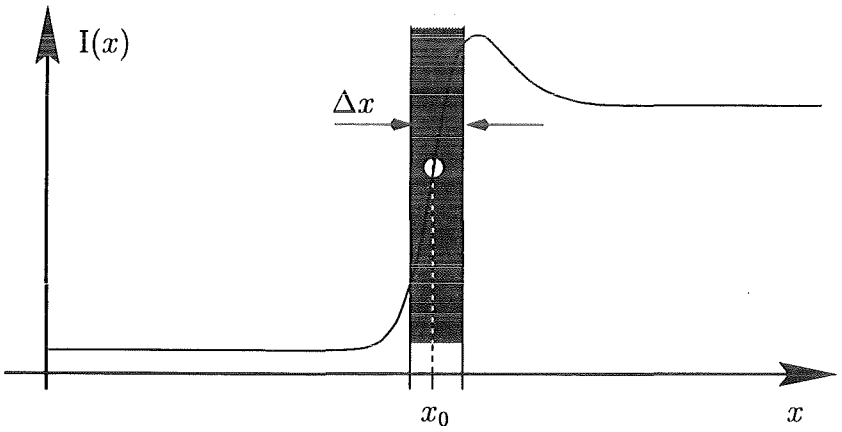


Abbildung 3.14: Vergrößerung der dritten Funktion aus Abb. 3.12. Der grau schraffierte Bereich kennzeichnet den Bereich, in dem Kantenoperatoren ohne *Annahme eines speziellen Modells das Zentrum des Übergangs finden.*

Der Fehler in der Lokalisierung des Übergangszentrums beträgt Δx. Dies ist der maximal mögliche Fehler, der eintritt, falls für die Modellierung der Kante eine ideale Stufe benützt wird. Δx liegt typischerweise in der Größenordnung von ein bis zwei Pixeln. Dieser Wert hängt stark von der gewählten Vergrößerung des Aufnahmesystems ab. Wie zuvor erläutert kann die Wahl des Startparameters *Kantenbreite* so erfolgen, daß diese Breite faktisch Null ist, ohne die Konvergenz des Verfahrens zu gefährden.

Problematischer für den Erfolg des Verfahrens ist die Kenntnis der Intensitätswerte I, die im Folgenden als Grundintensitäten bezeichnet werden. Diese sind insofern kritisch, als über sie kein Vorwissen aus dem Entwurf der Struktur existiert.

Dieses Vorwissen wäre theoretisch dadurch zu erreichen, indem ein umfassenderes physikalisches Modell sowohl der Szene, als auch des Aufnahmesystems

entwickelt wird. Für eine rein optische Aufnahmetechnik bestünde die praktische Realisierung beispielsweise in einem Raytracer. Ein Raytracer gestattet die Generierung photorealistischer Aufnahmen einer gegebenen Szene. Dazu werden die physikalischen Eigenschaften der beteiligten Elemente, also deren Reflexions-, Adsorptions- und Brechungsverhalten, benützt, um die für einen Beobachter an einem gegebenen Punkt sichtbare Intensität zu berechnen. Dazu wäre dann allerdings eine genaue Kenntnis der Absorptions- und Reflexionseigenschaften der im *LIGA* Verfahren verwendeten Materialien notwendig. Für die ebenfalls zu betrachteten REM Aufahmen wäre eine Erweiterung notwendig, da hier andere physikalische Grundlagen, als im lichtoptischen Fall, eingehen.

Diese Methoden zur Bestimmung der Grundintensitäten sind zumindest sehr aufwendig in der praktischen Realisierung. In dieser Arbeit wird jedoch ein einfacherer Ansatz gewählt. Es handelt sich dabei um eine Festsetzung unter Zuhilfenahme der geometrischen Näherungswerte.

Solange die *geometrischen* Startwerte hinreichend genau bekannt sind, kann eine a priori Abschätzung der Grundintensitäten durch Mittelwertbildung aus einem geeigneten Bereich der realen Szene getroffen werden, wie in Abb. 3.15 dargestellt wird.

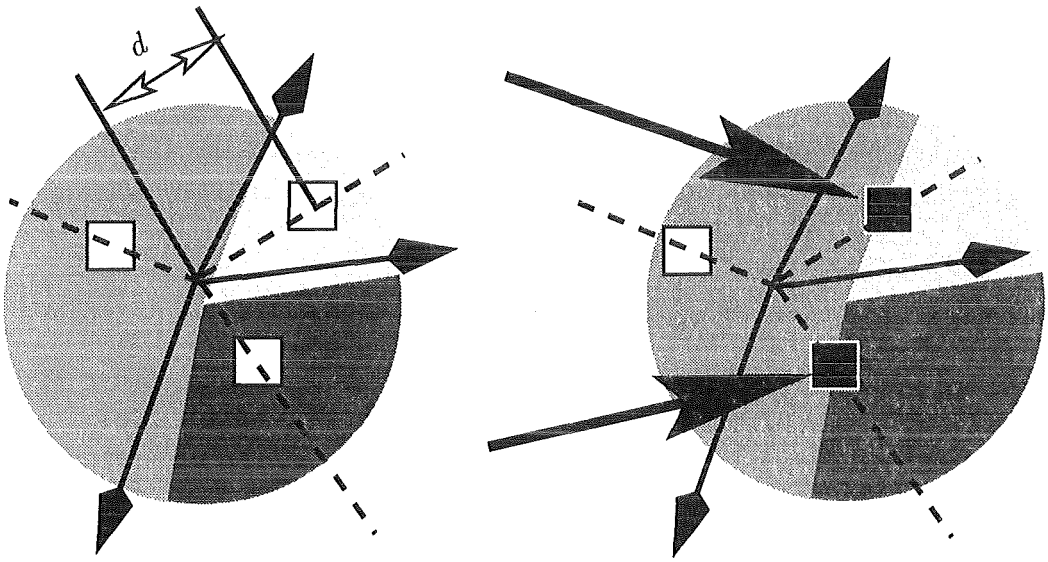


Abbildung 3.15: Bestimmung der Grundintensitäten. Der Abstand d muß so gewählt sein, daß die Meßfenster innerhalb eines Bereiches homogener Intensität liegen. Dies kann immer erreicht werden, indem d hinreichend groß gewählt wird. Allerdings kann dann der Fall eintreten, daß benachbarte Strukturen die Bestimmung der Intensitäten stören. Das rechte Bild zeigt einen Fall, wo zwei durch große schwarze Pfeile markierte Meßfenster aufgrund einer zu kleinen Wahl von d einen Übergang schneiden.

Die beiden Diagramme skizzieren jeweils den gleichen Ausschnitt einer realen Szene. Die durchgezogenen Linien stellen den geometrischen Teil des Modells dar. Dieses Modell würde in der Realität durch die Näherungswerte für die Modellparameter bestimmt und hat daher i. allg. die deutlich erkennbaren Abweichungen zur realen Szene. Entlang jeder gestrichelt eingezeichneten Winkelhalbierenden befindet sich ein quadratisches Fenster. Diese Fenster dienen der Bestimmung einer gemittelten Intensität für die gesuchten Startparameter I.

Im linken Teil der Darstellung ist die Abweichung zwischen Modell und Realität gering, daher befinden sich alle drei weiß eingezeichneten Meßfenster im Inneren eines homogenen Bereiches. Im rechten Teil hingegen sind die Abweichungen der geometrischen Parameter so groß, daß zwei der drei Fenster eine Kante schneiden, diese sind schwarz markiert. Dies führt zu einer unerwünschten Mittelwertbildung und somit zu fehlerhaften Startwerten.

Erste Versuche mit diesem simplen Modell zeigten eine relativ hohe Rate von Fehlklassifikationen.

Das Problem besteht darin, daß einerseits der Abstand d möglichst groß gewählt werden muß, um Grenzverletzungen zuverlässig auszuschließen. Große Werte von d können aber ebenfalls zu Fehlern führen, falls Grenzen zu benachbarten Bildelementen überschritten werden.

Dieses Dilemma kann teilweise behoben werden, wenn ein Kriterium existiert, welches das Schneiden einer Grenze anzeigt. Man kann dann mit kleinstmöglichem Abstand beginnen, und d solange vergrößern, bis das Kriterium erfüllt ist.

Ein mögliches Kriterium ist jedes Maß für die Homogenität des betrachteten Bildfensters. Die maximale mögliche Homogenität der Intensitätsverteilung ist gegeben durch den Rauschanteil. Man definiert daher:

$$\begin{aligned} \mu &= \frac{1}{N}\sum_{k=1}^{N} \mathrm{I}_k = \langle \mathrm{I} \rangle \\ \sigma &= \frac{1}{N}\sum_{k=1}^{N} (\mathrm{I}_k - \mu)^2 = \langle \mathrm{I}^2 \rangle - \langle \mathrm{I} \rangle^2 \end{aligned} \tag{3.1}$$

Der Wert μ bedeutet den Mittelwert und σ die Streuung der Intensitätswerte des zu mittelnden Bildfensters.

Für einen Bildausschnitt in einem homogenen Bereich des Bildes liegt das so gewonnene σ im Bereich der durch das Rauschen bestimmten Varianz. Die Wahrscheinlichkeit für das Schneiden einer Grenze wächst mit steigendem σ an. Durch Definition eines geeigneten Abschneideparameters kann dann der notwendige minimale Abstand d vom Zentrum ermittelt werden.

Trotz dieser Verbesserung verbleibt eine signifikante Zahl von Fällen, in denen das Verfahren aufgrund schlechter Startparameter entweder gar nicht, oder aber gegen ein lokales Minimum konvergiert.

3.5 Verbesserung geometrischer Startparameter

3.5.1 Grundlegendes Verfahren

Das Problem geeigneter Startwerte wird in der Literatur nur unzureichend beantwortet. Im vorliegenden Fall stellt sich die Situation wie folgt dar. Die geometrischen Startwerte eines iterativen Verfahrens werden aus dem CAD- System gewonnen. Problematisch sind dabei Abweichungen, die sowohl durch den Fertigungsprozeß, als auch durch Toleranzen in der Positionierung des Bildaufnahmesystems entstehen.

Zur Lösung dieses Problems wurde eine Kombination mit einem bekannten Verfahren durchgeführt, welches in [KR82], [WR83] und [PJJJ84] beschrieben wird.

Es handelt sich um ein Verfahren, „einfache" Ecken direkt zu erkennen. Der Ausdruck „einfache" kennzeichnet dabei Ecken, deren definierende Kanten deutliche Übergänge aufweisen. Zur Illustration des Prinzips dient Abb. 3.16.

Es wird das Merkmal ausgenutzt, daß die senkrechten Projektionen der Intensitäten an jeder Ecke innerhalb eines Bildes eine Änderung der Steigung aufweisen. Diese Projektionen werden durch die Zeilen- und Spaltensummen der Intensitäten gebildet. Für einen Bildausschnitt der Größe $w \times h$ definiert man die Projektionen als:

$$\begin{aligned} \mathbf{P}_x(i) &= \sum_{j=1}^{h} g_{ij}, \quad i \in \{1, \ldots, w\} \\ \mathbf{P}_y(j) &= \sum_{i=1}^{w} g_{ij}, \quad j \in \{1, \ldots, h\} \end{aligned} \tag{3.2}$$

Dies entspricht den Graphen I und II in Abb. 3.16.

Ein wesentlicher Punkt bei dieser Summenbildung ist die relative Robustheit gegen überlagertes Rauschen. Betrachtet man nämlich das Signal g_{ij} als Summe aus einem ideal ungestörten Signal q_{ij} und einem überlagerten Rauschen r_{ij} mit Erwartungswert $\langle r_{ij} \rangle = 0$, so folgt aus dem Gesetz der großen Zahlen:

$$\mathbf{P}_x(i) = \sum_{j=1}^{h} (q_{ij} + r_{ij}) = \sum_{j=1}^{h} q_{ij} + \sum_{j=1}^{h} r_{ij} \approx \sum_{j=1}^{h} q_{ij}$$

weil gilt:

$$\lim_{h \to \infty} \sum_{j=1}^{h} r_{ij} = 0$$

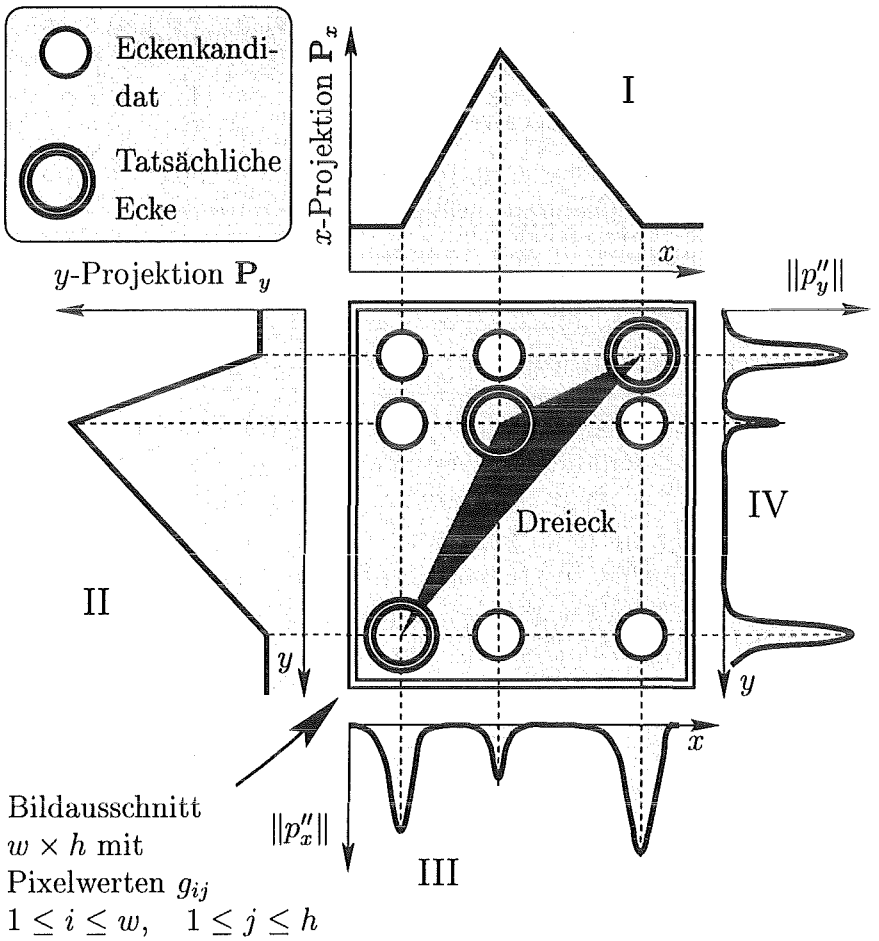


Abbildung 3.16: Die Abbildung zeigt im Zentrum die Skizze eines Grauwertbildes mit einem dunklen Dreieck im Zentrum. Die Diagramme I und II stellen die Projektionen der Grauwerte gemäß Gleichung 3.2 dar. Aus diesen Projektionen werden dann die Differenzen zweiter Ordnung in III und IV gebildet, wodurch die Koordinaten der möglichen Eckenpositionen erhalten werden. Aus dieser Kandidatenmenge werden dann die tatsächlichen vorhandenen Ecken ausgewählt.

Dieses Ergebnis gilt grundsätzlich auch im Fall des Vorhandenseins textureller Merkmale, solange es sich um eine statistische Verteilung handelt.

Diese Projektionsfunktionen haben nun folgende Eigenschaft: Solange die untersuchte Szene ausschließlich gerade Begrenzungslinien enthält, verlaufen die Projektionen außerhalb auftretender Eckenpunkte linear. In Eckenpunkten der Szene tritt eine Änderung der Steigung auf, solange keine Aufhebung durch andere Effekte wie in Abb. 3.17 eintritt.

Um nun die Koordinaten möglicher Ecken zu bestimmen, muß die Änderung der Steigung detektiert werden. Dies kann beispielsweise durch den Betrag eines Differenzenoperators zweiter Ordnung erfolgen. Die Koordinaten der Kandidaten werden dann als „Peaks" wie in den Diagrammen III und IV von Abb. 3.16 erhalten. Solche Peaks sind Erhebungen über ein Grundniveau mit geringer Breite, beispielsweise eine Gauß'sche Glockenkurve mit geringer Streuung σ. Diese Peaks können dann durch Schwellwertbildung detektiert werden.

Wie das Beispiel zeigt, gibt es bei jeweils drei x und y Koordinaten insgesamt $3 \times 3 = 9$ mögliche Eckenkandidaten, von denen in diesem Fall nur drei wirkliche Eckpunkte darstellen. Die neun Kandidaten müssen daher einer Klassifikation unterzogen werden. Als Merkmal kann auch hier wieder ein Maß für die Homogenität einer Umgebung des fraglichen Punktes herangezogen werden, etwa σ in Gleichung 3.1. In [WR83] wurde die Differenz der jeweils drei hellsten und der drei dunkelsten Intensitätswerte des Histogramms der Grauwerte einer 5×5 Umgebung als Maß benützt.

Die in Abb. 3.17 als Kreise mit weißem Inneren markierten Kandidaten liegen jeweils auf einem Gebiet homogener Intensität und werden somit nicht als Ecken klassifiziert.

3.5.2 Artefakte bei speziellen Lagen

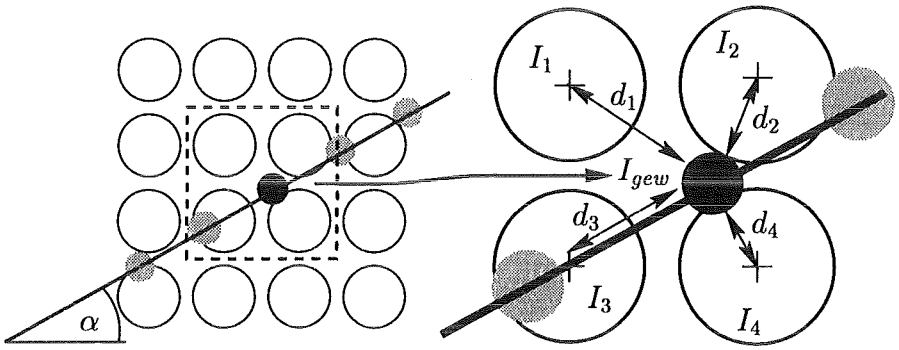
Der oben erwähnte Fall der Aufhebung von Änderungen der Projektionsstärken kann immer dann eintreten, wenn die Veränderungen der Projektion zweier Ecken sich gegenseitig kompensieren. Betrachtet man Abb. 3.17, so werden die markierten Ecken in der y- Projektion nicht erkannt, da keine Änderung der Steigung auftritt.

Abbildung 3.17: Im hier dargestellten Beispiel heben sich die Änderungen in der x- Projektion zweier Kanten gegenseitig auf, so daß die zugehörigen Ecken nicht identifiziert werden können. Als Ausweg können nicht achsenparallele Projektionsrichtungen gewählt werden.

Als Ausweg wurde folgender Algorithmus entwickelt: Wenn der obige Fall eintritt, werden die Projektionen nicht in Pixelrichtung und damit axenparallel durchgeführt, sondern entlang einer Schar von Parallelen, welche einen Winkel α mit dem Koordinatensystem bilden, welcher kein Vielfaches von 90° ist. Auf diese Weise können Artefakte wie in Abb. 3.17 beseitigt werden.

Die verwendete Implementierung dieser alternativen Form zur Bestimmung von Projektionen entlang einer beliebigen Richtung basiert auf Mittelung von maximal vier benachbarten Pixeln. Damit wurden gute Ergebnisse erzielt.

Eine Verbesserung ist denkbar mit einem Verfahren, wie es beim sogenannten *anti aliasing* in Grauwertbildern zur Vermeidung störender Rasterung verwendet wird. Das Prinzip des *anti aliasing* mittels Graustufen besteht darin, das Prinzip des „Alles oder nichts" für das Setzen von Pixeln zu ersetzen durch eine Graustufendarstellung. Dabei werden auch Pixel in einer geringeren Intensität gesetzt, die nicht zu einem abzubildenden Objekt gehören. Für den hier zu betrachtenden Fall bedeutet dies eine bessere Gewichtung der Grauwerte umgebender Pixel, wie die Abb. 3.18 zeigt. Die hellgrauen Kreisflächen entlang der durchgezogenen Linie symbolisieren die „Pixel" entlang einer bestimmten Richtung α



$$I_{gew} = w_1 I_1 + w_2 I_2 + w_3 I_3 + w_4 I_4$$

$$w_k = \frac{\frac{1}{d_k}}{\frac{1}{d_1}+\frac{1}{d_2}+\frac{1}{d_3}+\frac{1}{d_4}} \Longrightarrow w_1 + w_2 + w_3 + w_4 = 1$$

Abbildung 3.18: Wichtung von Intensitäten benachbarter Pixel. Der Abstand der Meßpunkte auf der Linie entlang des Winkels α ist gleich dem Pixelabstand.

3.5.3 Rangfolge und Bewertung

Das Ergebnis der Bestimmung von Kandidaten über Intensitätsprojektionen sind bei realen Bildern i. allg. mehrere mögliche Eckenpositionen. Dies liegt daran, daß trotz der erwähnten glättenden Wirkung einer Projektion bereits das Bildrauschen einen Einfluß hat. Dies gilt insbesondere, da die Anwendung jedes Differenzenoperators ein numerisch schlecht konditioniertes Ergebnis hat.

Aus diesem Grund wird eine Strategie benötigt, unter den möglichen Eckenpositionen eine Rangfolge der Wahrscheinlichkeit zu definieren. Diese Positionen werden dann nacheinander als Startwerte für das Ausgleichsproblem in Gleichung B.4 benützt, bis zum ersten Mal Konvergenz eintritt.

Abb. 3.19 zeigt ein Beispiel mit fünf möglichen Eckenpositionen.

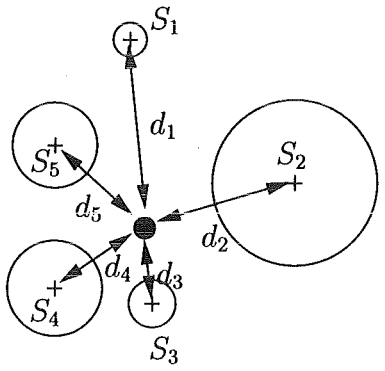


Abbildung 3.19: Rangfolge für Eckenkandidaten

Die Stärke S dieser Ecken wird gemäß Gleichung 3.3 gewonnen und ist durch die unterschiedlichen Radien der Kreise symbolisiert. Ein großer S- Wert bedeutet, daß die jeweilige, aus der Intensitätsprojektionen gewonnene, Position sowohl für die x-, als auch für die y- Koordinate, einen starken Übergang S_x bzw. S_y aufweist. Um dies zu erreichen, wird S als harmonisches Mittel der Komponenten definiert:

$$S = \sqrt{S_x S_y} \tag{3.3}$$

Diese Festsetzung garantiert, daß *beide* Komponenten mit einer Mindeststärke zu S beitragen.

Der schwarze Kreis im Zentrum von Abb. 3.19 kennzeichnet die durch die CAD-Daten ermittelte Näherung für die Position der gesuchten Ecke. Je weiter ein Eckenkandidat von dieser Koordinate entfernt ist, desto geringer ist die Wahrscheinlichkeit, daß es sich um die gesuchte Ecke handelt.

Diese beiden Kriterien:

- Abstand vom CAD- Wert d
- Intensität der Projektion S

werden nun in einer empirisch ermittelten Funktion zu einer Kostenfunktion $\mathbf{R}(d, S)$ (Rating) kombiniert. Dabei wurden folgende Varianten getestet:

- $\mathbf{R}(d, S) = \alpha S - \beta d, \quad 0 \leq \alpha, \beta$
- $\mathbf{R}(d, S) = (1 - e^{-\alpha S}) - \beta d$

Die erste Funktion stellt den einfachsten denkbaren Fall dar. Beide Funktionen enthalten einen „Strafterm“ $-\beta d$. Dies bedeutet, daß Kandidaten, welche weit entfernt von der CAD- Position sind, in der Rangfolge spät berücksichtigt werden. Die zweite Funktion enthält den Faktor $1-e^{-\alpha S}$. Dieser bewirkt eine „Sättigung“. Der Grund dafür ist heuristischer Natur. Ecken ab einer gewissen Qualität sind eindeutig als solche klassifizierbar, der Faktor bewirkt gerade ein Sättigungsverhalten. Empirisch zeigte sich, daß die zweite Funktion im Mittel eine wesentlich bessere Ordnung hinsichtlich der Startparameter definierte, d.h., die korrekten Startwerte lagen in 95% der Fälle an erster Stelle und in 98% aller Fälle unter den ersten drei von fünf möglichen Positionen. Das schlechtere Abschneiden des linearen Ansatzes liegt an dem fehlenden Sättigungsverhalten.

3.5.4 Integration der Komponenten

Das bisher in Einzelkomponenten beschriebene Gesamtsystem zur modellbasierten Erkennung ist im dunkel schattierten Bereich von Abb. 3.20 dargestellt. Die gesamte Abbildung zeigt die Einbettung in ein Gesamtsystem.

Als Ergebnis eines Meßauftrages liefert das System dann:

1. die geometrischen Parameter der vermessenen Struktur
2. eine Bewertung hinsichtlich der Qualität der Vermessung

Die Notwendigkeit des zweiten Punktes wird ersichtlich, wenn man die Tabelle in §5.1.1 betrachtet. Bei einer schlechten Bildqualität sinkt auch die Qualität der Erkennung und damit steigen die Fehlergrenzen der ermittelten Parameter. Aus diesem Grund ist es notwendig, ein Gütemaß zu definieren.

Die mit den bisher dargestellten Verfahren erzielten Ergebnisse führen zur Berücksichtigung textureller Merkmale für die Erkennungsaufgabe. Um einen Vergleich zu ermöglichen, werden die Ergebnisse erst in §5.1 beschrieben.

3.5.5 Elastische Constraints

In Abb. 5.2 tritt der Fall ein, daß zwei der drei zu einer Kante gehörenden Übergänge eindeutig bestimmt werden können. Wenn man nun annimmt, daß der fehlende Übergang aufgrund eines Artefaktes nicht erkannt werden kann, so läßt sich die Konvergenz des Verfahrens erreichen, indem Beschränkungen für die Wahl der Modellparameter eingeführt werden.

Die Einführung elastischer Constraints bietet eine einfache Möglichkeit, das in Abb. 5.2 aufgetretene Problem zu lösen. Dabei wird die Tatsache benützt, daß bestimmte Sollmaße eine hohe Genauigkeit haben, welche als Referenz dienen kann.

Dies wurde im konkret vorliegenden Fall realisiert, indem ein Strafterm für relative Änderungen der Sollwinkel eingeführt wurde. Dies ist insofern gerechtfertigt, als die Ist- Werte der Winkel sehr genau mit den Sollwerten übereinstimmen, was hingegen nicht für die absoluten Positionen gilt.

Dies Vorgehensweise ist *keine* Alternative zur *grundlegenden* Behandlung in §4, bietet jedoch eine Möglichkeit, generell die durch fehlende Konvergenz verursachte Rate der Nichtklassifikationen zu vermindern. Sie kann immer dann eingesetzt werden, wenn ein Teil der Parameter sehr genau vorausberechnet werden kann.

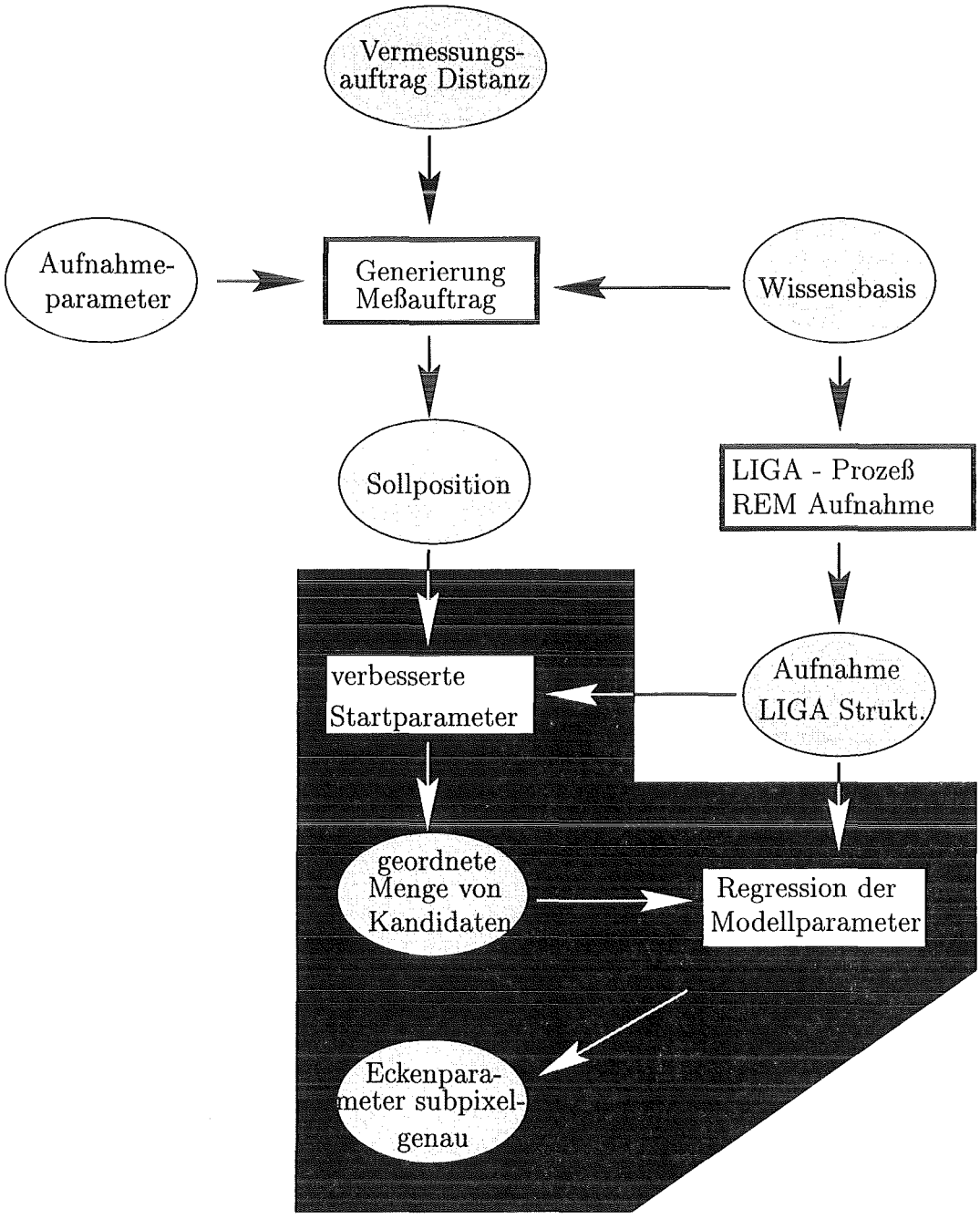


Abbildung 3.20: Dies zeigt die Realisierung eines Gesamtsystemes zur modellbasierten Erkennung. Der schwarz hinterlegte Teil kennzeichnet den in dieser Arbeit behandelten Teil eines solchen Systems.

Kapitel 4

Texturbasierte Unterstützung

4.1 Problemstellung

Das Beispiel von Abb. 5.2 zeigt die Problematik einer fehlerhaft erkannten Ecke. Abb. 4.1 zeigt eine Vergrößerung des fraglichen Bereiches im Grauwertbild.

Eine Analyse dieser Vergrößerung zeigt die Ursache für das Versagen des Verfahrens. Die fehlerhaft erkannte Ecke wird durch drei Gebiete mit unterschiedlichen mittleren Grauwerten $\bar{I}_k, \quad k \in \{1,2,3\}$ definiert. Die Mittelwertbildung erfolgt innerhalb der drei weiß eingezeichneten 7×7 Bildfenster. Zusätzlich zum Mittelwert wurde jeweils die Streuung σ_k angegeben.

Wie aus der Abbildung ersichtlich ist, unterscheiden sich die Mittelwerte der Graustufen zwischen den Regionen zwei und drei lediglich um den Wert 0.5. Dies entspricht einer relativen Änderung der Intensität von 0.7%. Die mittlere Streuung der beiden beteiligter Gebiete übertrifft die absolute Differenz um etwa eine Größenordnung. Das Verfahren erkennt daher diese Differenz nicht mehr. Es tritt dennoch Konvergenz ein. Diese wird durch die stärkeren Intensitätsdifferenzen $I_1 \rightarrow I_2$ und $I_1 \rightarrow I_3$ erzielt. Evident wird dieses Verhalten, wenn die Konvergenz genauer untersucht wird. Dabei zeigt sich, daß lediglich der Parameter für die Richtung der problematischen Kante oszillierte, während alle anderen Parameter konvergierten.

Dies gestattet die folgende einfache Interpretation:

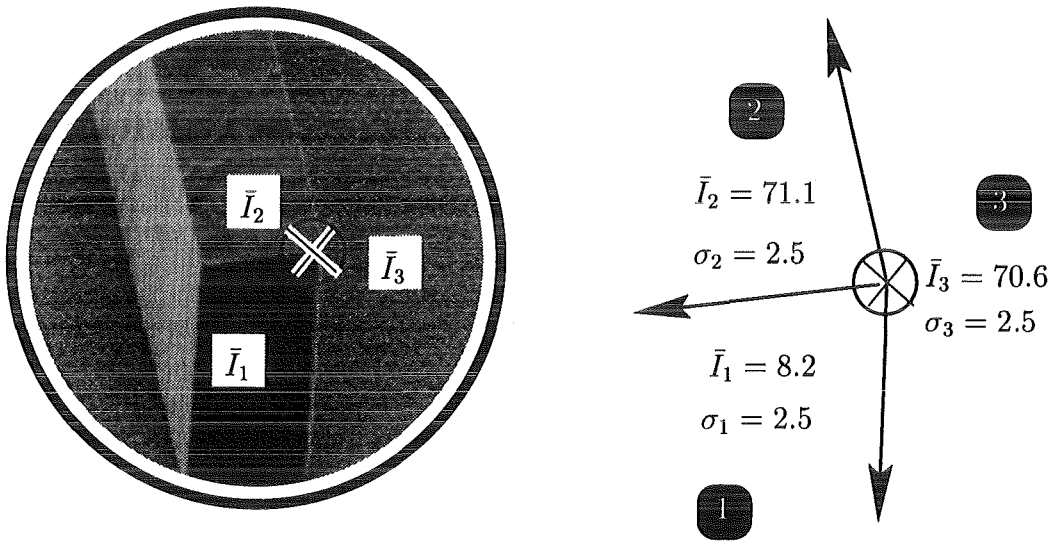
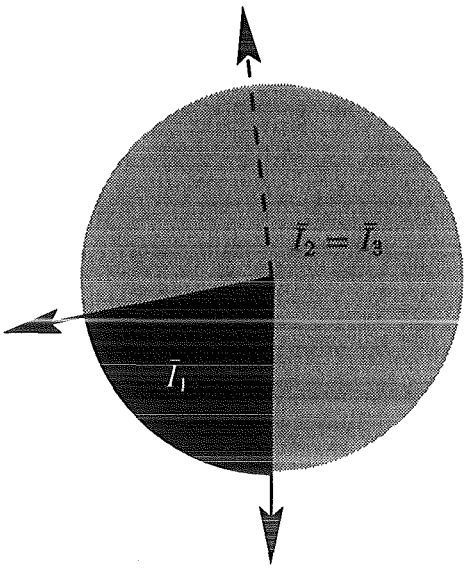


Abbildung 4.1: Dies ist eine Vergrößerung des Bereiches der fehlerhaft erkannten Ecke in Abb. 5.2. Es erscheint auf den ersten Blick fraglich, warum diese für das menschliche Auge deutlich erkennbare Ecke durch das System nicht erkannt wurde. Eine genauere Analyse zeigt die Ursache: Die über die weiß dargestellten Bildfenster gemittelten Intensitäten $\bar{I}_2$ und $\bar{I}_3$ unterscheiden sich um einen Betrag, der deutlich unter der Streuung des Rauschanteiles der Aufnahme liegt. Die scheinbar einfache Segmentierung erfolgt hier nicht auf der Basis von Grauwertunterschieden, sondern durch die unterschiedlichen Texturen der beiden benachbarten Regionen.



Das Verfahren *erkennt* eine Ecke mit *zwei* auslaufenden Linien. Diese Linien begrenzen zwei Regionen mit unterschiedlichen Intensitäten $I_1 = 8$ und $I_2 = 71$. Die Intensitätsdifferenz ist hier klar definiert, daher konvergiert das Verfahren und liefert die korrekten Werte für Zentrumskoordinaten und Richtungen.

Die Betrachtung von Abb. 5.2 suggeriert eine unproblematische Erkennungsaufgabe. Für einen menschlichen Betrachter handelt es sich unzweideutig um eine Ecke mit *drei* Kanten. Diese Erkennung wird dadurch möglich, daß die Regionen 2 und 3 zwar einen nahezu identischen mittleren Grauwert, aber deutlich unterschiedliche *texturelle* Merkmale besitzen.

Die Erkennung solcher Merkmale durch biologische neuronale Systeme, wie dem menschlichen Gehirn, ist in der Natur in beeindruckender Weise realisiert, bereitet indes maschinellen Systemen enorme Schwierigkeiten. Es stellt sich die Frage, wie dieses zusätzliche Merkmal „Textur"verwendet werden kann.

4.1.1 Zur Begriffsdefinition der Textur

Eine exakte Definition des Texturbegriffes erscheint zum gegenwärtigen Zeitpunkt unmöglich. Die Abbildungen Abb. 4.2 und Abb. 4.3 dienen als intuitive Beispiele zum Verständnis des Texturbegriffes.

Die Definitionen in der Literatur sind uneinheitlich und zum Teil widersprüchlich. Ebenso fehlt bislang ein klares Schema für eine generelle Einteilung in Kategorien. In [Pra78] wird eine Einteilung hinsichtlich natürlicher und künstlich erzeugter Texturen vorgenommen. Ein Merkmal ist die regelmäßige Wiederholung einer elementaren Grundstruktur, wie sie in Abb. 4.2 auftritt. Eine gute Darstellung und Klassifikation gebräuchlicher Verfahren zur Texturanalyse findet sich in [Zha94].

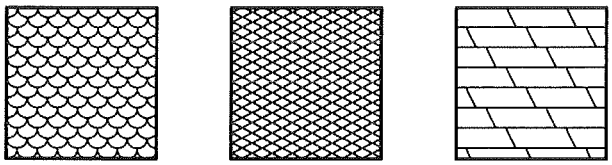


Abbildung 4.2: Die drei Texturen entstehen durch Wiederholung einer elementaren Struktur. Ein alltägliches Beispiel ist der Aufdruck einer Tapete.

Abb. 4.3 zeigt Beispiele für stochastisch definierte Texturen. Diese wurden durch einen Raytracer synthetisch generiert, wie er kurz auf Seite 49 erwähnt wurde.

[Pic70] definiert Texturen als zweidimensionale Felder von variierender Intensität, wobei die Regeln zur Variation der Anordnung beliebig sind, solange eine charakteristische Wiederholung der Grundelemente vorhanden ist.

Eine etwas detailliertere Beschreibung findet sich in [Haw70]. Danach beinhaltet der Texturbegriff drei Merkmale:

1. Die Wiederholung einer lokalen „Ordnung " über eine Region, welche eine große räumliche Ausdehnung gegenüber der zur Definition der „Ordnung" notwendigen Größe hat.

2. Die Ordnung besteht aus einer nicht zufälligen Verteilung elementarer Teilelemente.

3. Die Teilelemente sind annähernd einheitliche Strukturen mit vergleichbarer räumlicher Ausdehnung über den Bereich der Textur.

Diese deskriptive Formulierung präzisiert zwar den rein intuitiven Texturbegriff, liefert aber keine strenge Definition und auch keinen Weg zum Auffinden von Merkmalen texturierter Regionen.

Texturen werden oft durch ihre „Granularität" beschrieben. Eine Sandoberfläche besitzt bei gleichen Aufnahmebedingungen eine „feinere" Oberflächenstruktur als eine Kiesoberfläche. Eine Möglichkeit, dies methodisch zu erfassen, ist die Zerlegung nach Fouriermoden. Eine feiner strukturierte Textur hat gegenüber einer groben Textur eine Amplitudenverteilung mit Schwerpunkt im vergleichsweise hohen Frequenzbereich des Fourierspektrums. Diese Granularität ist allerdings kein Merkmal zur alleinigen Texturklassifikation.

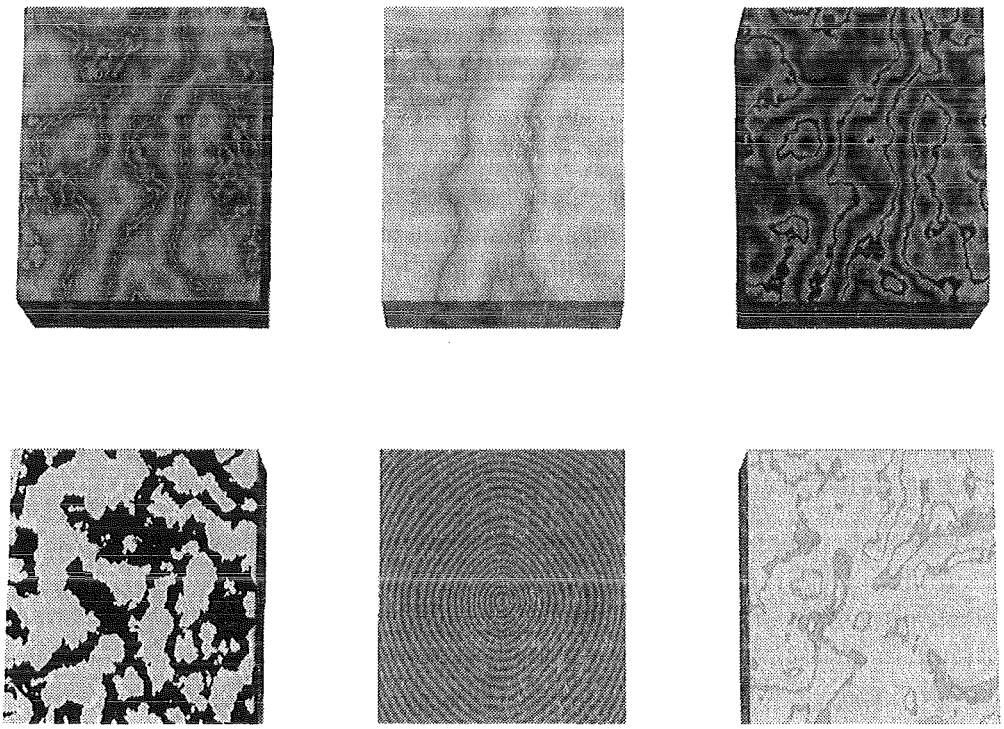


Abbildung 4.3: Beispiele synthetisch generierter stochastischer Texturen. Im Gegensatz zur Abb. 4.2 entsprechen diese Beispiele eher Texturen aus realen Szenen.

4.1.2 Segmentierung durch Texturmerkmale

Wie in [Krö87] dargelegt, spielt die Analyse textureller Merkmale eine entscheidende Rolle in der menschlich-visuellen Objekterkennung. Für das maschinelle Sehen ist es wesentlich, unterschiedlich texturierte Bereiche zu unterscheiden. Im Idealfall kann dann für jeden Bildpunkt der Aufnahme eine Entscheidung getroffen werden, zu welcher Texturklasse er gehört. Die Idee ist in Abb. 4.4 dargestellt.

Das Vorgehen ist dabei analog zu einer intensitätsbasierten Segmentierung, wobei das Merkmal „homogene Intensität“ durch „homogene Textur“ ersetzt wird.

4.1.3 Betrachtung von Nachbarschaften

Die zuvor erwähnten Definitionen unterstreichen die Tatsache, daß Textur eine Eigenschaft einer Nachbarschaft einer Bildkoordinate ist. Aus diesem Grund benötigt ein Texturklassifikator die Größenangabe einer Nachbarschaft. Diese Nachbarschaft muß mindestens die Größe der in Punkt drei erwähnten Teilelemente aufweisen. Eine Vorbedingung für eine erfolgreiche Klassifikation ist, daß diese Nachbarschaft *vollständig* in einer homogen texturierten Region enthalten ist.

Die Notwendigkeit der Betrachtung einer solchen Nachbarschaft ist Voraussetzung eines *jeden* Texturklassifikators. In §C werden grundsätzliche Verfahren zur Konstruktion solcher Klassifikatoren beschrieben.

Die Größe der Nachbarschaften kann in der Praxis zu Beschränkungen in der Anwendbarkeit führen. Falls nämlich die zu untersuchende Szene Bildelemente aufweist, die feiner sind als die Körnung eines homogenen texturierten Bereiches, wird eine ausreichende Auflösung über texturelle Merkmale unmöglich. Dies ist anschaulich gut verständlich. Die Grenze beispielsweise zwischen einer Sand- und einer Kiesschicht kann bestenfalls bis zu einer Auflösung erzielt werden, welche der Korngröße der groberen Kiesregion entspricht. In diesem Beispiel ist *grundsätzlich* keine bessere Ortsauflösung möglich. In der Praxis treten allerdings häufiger Fälle auf, in denen das verwendete Verfahren zur Texturdiskriminierung suboptimal arbeitet, und aus diesem Grund die *prinzipiell* mögliche Auflösung nicht erreicht wird.

Die zur Texturanalyse notwendigen Nachbarschaften führen zu einem Dilemma. Eine Textur kann i. allg. besser klassifiziert werden für Nachbarschaften großer räumlicher Ausdehnung, wodurch die mögliche Auflösung vermindert wird. Typische Fenstergrößen sind 16×16 . Dies ist wesentlich mehr, als üblicherweise für Kantenoperatoren verwendet wird. In der praktischen Anwendung wird daher ein Mittelweg gesucht zwischen der notwendigen Rate erfolgreicher Klassifikationen und der möglichen Strukturauflösung.

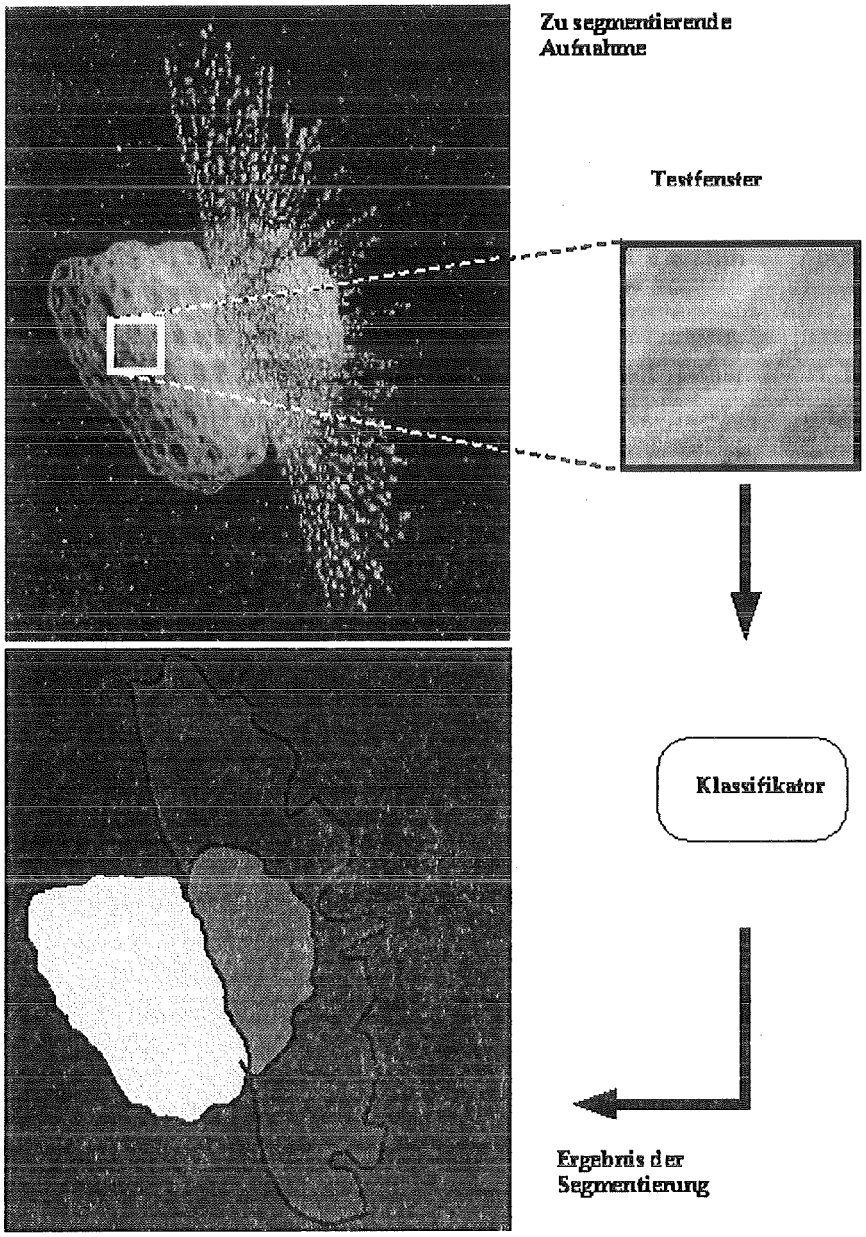


Abbildung 4.4: Segmentierung durch texturelle Merkmale. Die obige Abbildung zeigt einen Kometen, welcher eine kraterförmig texturierte Oberfläche aufweist. Dieses Merkmal kann benutzt werden, die untenstehende Segmentierung der Szene zu erzeugen.

Diese Problematik spielt eine wesentliche Rolle für ein Vermessungssystem, da ja eine bestmögliche Lokalisierung von Bildstrukturen erzielt werden soll.

4.2 Ansatz zur Lösung

Die auf Seite 62 aufgetretene Problemklasse soll durch eine Texturanalyse gelöst werden. Grundsätzlich eignen sich dafür alle der im Anhang §C beschriebenen Verfahren. Es sind allerdings bestimmte Randbedingungen zu beachten, welche die Wahl einschränken.

Die Texturen sollen unabhängig von Grundintensitäten erkannt werden. Diese Grundintensitäten schwanken bei den betrachteten Aufnahmen, da sie bei 3-D Strukturen von den Winkeln abhängen, unter denen die Aufnahmen erfolgen. Die Unabhängigkeit von der Grundintensität einer Aufnahme ist wesentlich, da Texturunterschiede ja gerade jene Regionengrenzen erkennen sollen, für welche die definierenden Intensitäten nahezu identisch sind.

Die in §4.1.3 diskutierte Problematik der Umgebungen spielt eine zentrale Rolle. Da Grenzen von Regionen erkannt werden sollen, kann nicht davon ausgegangen werden, daß die zur Texturdiskriminierung verwendete Umgebung stets in einen homogen texturierten Bereich fällt. Diesen Effekt zeigt Abb. 4.5. Die Größe des zur Texturdiskriminierung verwendeten Fensters ist schattiert eingezeichnet. Die unterschiedlichen Höhen symbolisieren zwei verschiedene Texturen. Liegt die zur Diskriminierung verwendete Umgebung im Bereich des Überganges, so führt dies zu einer Verbreiterung des Überganges.

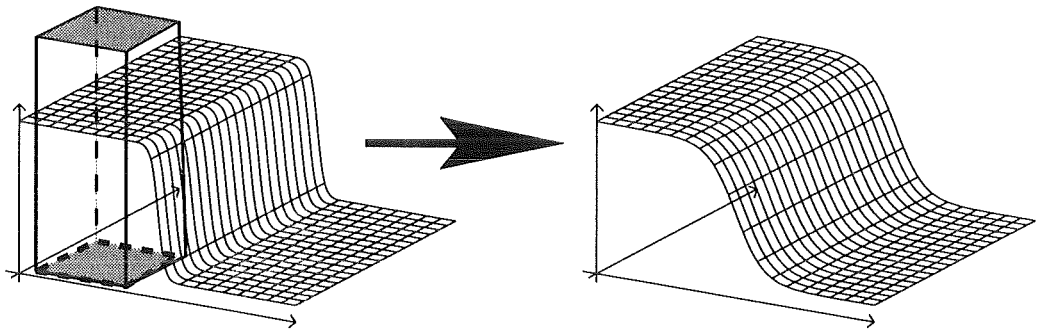


Abbildung 4.5: Verminderung der Auflösung durch Fenstergröße

Die Abbildung suggeriert bereits die Vorgehensweise. Das Ergebnis der Texturanalyse eines einzelnen Bildpunktes ist ein Zahlenwert, beispielsweise die bereits erwähnte Körnigkeit der Umgebung. Es handelt sich um *ein* Merkmal der Textur. Die dabei auftretende Verbreiterung ist prinzipiell unerwünscht, kann jedoch teilweise kompensiert werden, wenn ein Modell für die Form des Überganges, wie

in Kapitel 3.4 beschrieben, zur Verfügung steht. Auf diese Weise kann dann die Koordinate des Übergangszentrums mit der erforderlichen Genauigkeit bestimmt werden.

Das so erhaltene Verfahren kann als *Mehrkanalverfahren* bezeichnet werden. Der erste Kanal trägt einfach die Grauwertintensitäten des Bildes. Die weiteren Kanäle enthalten Funktionswerte von auf Nachbarschaften definierten Funktionen. Die Art dieser Funktionen ist zunächst willkürlich, beispielsweise könnten auch Fouriermoden verwendet werden. Die Konzentration auf *texturelle Merkmale* begründet sich auf die Erfahrung, daß diese in biologischen Vorbildern massiv zur Erkennung unterschiedlicher Regionen beitragen.

Die Realisierung des Systems als Mehrkanalverfahren ermöglicht die leichte Integration weiterer geeignet erscheinender Merkmale.

4.2.1 Gewinnung orthogonaler Merkmale

Für eine feste Anzahl von n möglichen Texturklassen liefert dieses simple Vorgehen i. allg. *keine* vollständige Klassifikation. Um alle n Texturen klassifizieren zu können, müssen weitere Merkmale, wie in Gleichung C.4 beschrieben, hinzugenommen werden.

Dabei tritt nun ein allgemein bekanntes Problem auf. Es stellt sich die Frage, welche Merkmale *orthogonal* zueinander sind. In der Sprache der Statistik bedeutet dies, daß zwei Merkmale miteinander korreliert sein können. Je höher die Korrelation ist, desto weniger sinnvoll ist die Verwendung beider Merkmale. Im Fall maximaler Korrelation sind beide Merkmale vollkommen gleichberechtigt zur Klassifikation verwendbar [TC77].

Die Frage der Orthogonalität von Texturmerkmalen ist abhängig von der Beschaffenheit der n möglichen Klassen. Aus diesem Grund kann keine a Priori Entscheidung über die Auswahl der zu verwendenden Merkmale getroffen werden.

Um zu einer Auswahl zu gelangen, bieten sich prinzipiell zwei Ansätze an. Es ist dies zum einen die statistische Methode der Hauptwertanalyse. Der zweite Weg benützt neuronale Netze zur Klassifikation.

In dieser Arbeit wird ein neuronaler Ansatz gewählt, da die Parallelisierbarkeit im Vergleich zu statistischen Verfahren leichter erzielt werden kann. Beispiele solcher neuronal realisierten Verfahren finden sich in [Vis91], [PS91], [MD92] und [MD93]. Alle vier Arbeiten beschreiben Systeme von statistisch basierten neuronalen Klassifikatoren mittels Co-occurrence Matrizen, wie sie im Anhang §C beschrieben werden. Die Arbeit [ML91] beschreibt ein Verfahren, welches neben der

Klassifikation von Texturen noch die Möglichkeit der Orientierungsbestimmung bietet. Dies spielt im hier untersuchten Kontext bislang keine Rolle. Die Arbeiten [TSK90], [Oe93] und [SBC93] zeigen alternative Möglichkeiten der Merkmalsgewinnung durch autoregressive Modelle, wie sie in [Zha94] beschrieben werden. In [YY92] wird ein Verfahren zur Bestimmung von Grenzen homogen texturierter Bereiche basierend auf einem parallelen Algorithmus dargestellt.

Für die Realisierung eines neuronalen Texturklassifikators müssen die verwendeten Netze mittels Merkmalen aus bekannten Texturmustern trainiert werden. Aus diese Weise erhält man dann Gewichte für die verschiedenen Eingangssignale. Dabei ist zu untersuchen, ob die aus dem Bildausschnitt generierten Merkmale statistisch voneinander unabhängig sind. Diese Frage kann durch Analyse der zugehörigen Gewichte des verwendeten neuronalen Netzes beantwortet werden. Falls sich die entsprechenden Gewichte während der Trainingsschritte stets gleichsinnig mit den korrespondierenden Eingangssignalen verändern, handelt es sich um eine redundante Information. Die zugehörigen Eingangssignale können dann durch *einen* Repräsentanten ersetzt werden.

4.2.2 Praktische Realisierung

Die erste Frage einer Integration von Texturmerkmalen in das Modell betrifft die Auswahl der Merkmale selbst. Aus Laufzeitgründen wurden keine Fouriermoden betrachtet, da diese sehr zeitaufwendig in der Berechnung sind. Das Hauptkriterium zur Auswahl geeigneter Merkmale bestand in der potentiell leichten Möglichkeit zur Parallelisierung.

Im Hinblick auf das Mehrkanalverfahren wurden die in [HS92] beschriebenen Co-occurrence Matrizen zur Merkmalsgeneration verwendet. Ein wichtiger Grund dafür sind biologische Vorbilder. Die menschliche visuelle Erkennung kann zwischen Texturen diskriminieren, die sich in der Statistik zweiter Ordnung unterscheiden. Aus diesem Grund scheint zum gegenwärtigen Zeitpunkt die Betrachtung von Statistiken zweiter Ordnung, wie sie die Co-occurrence Matrizen repräsentieren, ausreichend. Eine Beschreibung findet sich im Anhang §C.

Diese Merkmale dienen dann als Eingabe für ein neuronales Netz vom Feed - Forward Typ. Eine Einführung in die Grundlagen der in dieser Arbeit verwendeten neuronalen Netze findet sich im §D.

Da bislang keine Methoden für die Bestimmung der optimalen Topologie eines neuronalen Netzes bezüglich eines gegebenen Problems existieren, wurden verschiedene Architekturen experimentell getestet. Die Vorgehensweise besteht in einer sukzessiven Reduktion von Neuronen in den inneren Layern des Netzes. Beim Unterschreiten einer „kritischen“ Anzahl erfolgt ein spontaner Anstieg der

Gesamtabweichung zwischen Soll- und Trainingsdaten.

Wie in §D beschrieben, ist für die Bewertung des Erfolges jener Fehler maßgeblich, der auf einer Untermenge des Datensatzes auftritt, welche *nicht* zum Trainingssatz gehört.

Das Training des Netzes erfolgt durch Bildausschnitte aus Aufnahmen realer Strukturen. Zu diesem Zweck müssen möglichst viele unabhängige Ausschnitte gefunden werden, um eine zuverlässige Klassifikation zu erzielen.

Die Aufgabe des Netzes bestand in der Texturklassifikation eines Bildausschnittes. Durch einen Präprozessor wurden aus jedem Bildausschnitt zunächst die vier Merkmale *Kovarianz, Kontrast, Energie* und *Entropie* gemäß Gleichung C.5 und folgende ermittelt. Die getroffene Auswahl reflektiert die relative statistische Unabhängigkeit dieser Merkmale in Bezug auf das verwendete Bildmaterial. Als Ausgabe des Netzes wurde ein einzelnes Neuron mit kontinuierlichen Werten gewählt, wobei der Ausgangspegel nach Einteilung in Intensitätsstufen direkt die verschiedenen Texturklassen wiedergibt. Um die interne Repräsentation der verschiedenen Texturen in Form von Gewichten möglichst einfach zu halten, wurden die Klassen nach steigender Körnigkeit ausgewählt.

Dies ist eine rein heuristische Vorgehensweise. In diesem Beispiel kann die Zuordnung eines kontinuierlichen Ausgangssignals zu verschiedenen Texturklassen theoretisch willkürlich erfolgen. Bei k verschiedenen Klassen sind $k!$ Permutationen möglich. Jede solche Permutation bedeutet nur eine andere *Darstellung* des Ergebnisses. Es zeigt sich allerdings, daß für einen gegebenen Trainingssatz nicht alle Darstellungen gleich gut gelernt werden.

Auf den ersten Blick verwundert dies, da eine Unabhängigkeit des Ergebnisses von der Wahl einer speziellen Darstellung erwartet wird. Es gibt allerdings ein triviales Gegenbeispiel, warum dies gerade nicht der Fall ist. Man betrachte ein Netz, welches die Zuordnung eines Eingangssignals mit n verschiedenen Eingangswerten ξ_i zu genau einem Ausgangssignal mit n Ausgangswerten O_i lernen soll. Dieses Netz kommt mit einer minimalen Anzahl von Gewichten aus, wenn die Zuordnung lediglich eine monotone Umparametrisierung bedeutet, also etwa $\xi_i \leq O_i$. Sobald dies nicht mehr gilt, müssen zusätzlich noch die auftretenden Permutationen gelernt werden. Ein solcher zusätzlicher Lernaufwand tritt auch im Fall der Texturklassifikation auf. Möglicherweise existieren neben der Körnigkeit weitere Merkmale, die zu beachten sind, um den Lernaufwand zu verringern.

Kapitel 5

Ergebnisse

5.1 Reines Intensitätsmodell

5.1.1 Simulierte Daten

Das bislang beschriebene Verfahren wurde zunächst mit simulierten Bilddaten bei unterschiedlichem Signal/Rauschverhältnis *Signal to Noise Ratio* getestet. Das verwendete Rauschen wurde als gaußverteilt angesetzt. In Abb. 5.1 sind zwei der erzeugten Testbilder mit identischen geometrischen Parametern dargestellt.

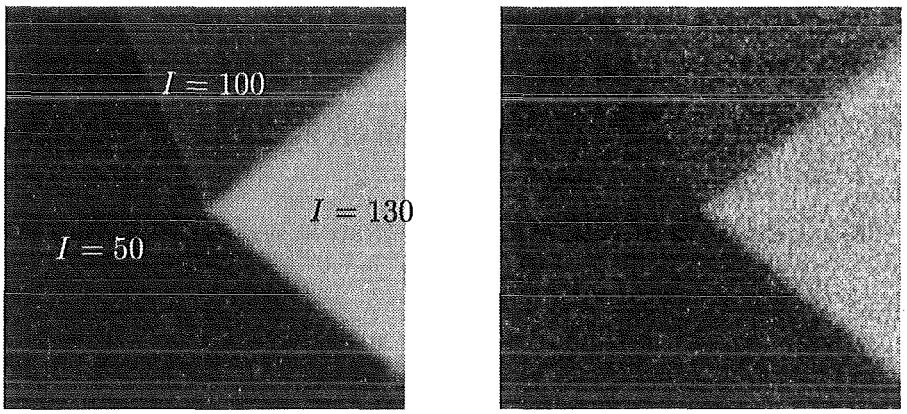


Abbildung 5.1: Zwei Beispiele für simulierte Ecken mit drei auslaufenden Kanten. Das linke Bild zeigt eine Ecke ohne überlagertes Rauschen. Die darin enthaltenen Kanten haben unterschiedliche Breiten, die linke Kante ist eine „ideale" Kante mit stufenförmigem Übergang, während die rechten beiden Kanten eine Breite von jeweils 8 Pixeln aufweisen.

Das linke Bild ist völlig rauschfrei, während das rechte Bild ein SNR von 1.0

aufweist. Die folgende Tabelle zeigt die Ergebnisse dieser Simulationen.

SNR	Fehlerrate in %	$\frac{\Delta x}{b} \cdot 100\%$
12.0	.01	0.25
10.0	.01	0.26
5.0	.01	0.28
1.0	.01	0.37
0.5	.02	.62
0.2	0.82	.83
0.1	12.0	1.15

Die Fehlerrate kennzeichnet den Anteil der nicht erfolgreichen Iterationen. Der Lokalisierungsfehler $\frac{\Delta x}{b}$ bezeichnet den relativen Fehler in der Ortsangabe des Kantenzentrums. Als Maß für den Nenner b wurde dabei die Breite des Übergangs wie in Abb. 3.9 gewählt. Ein Wert von $\frac{\Delta x}{b} = 0.25$ bedeutet also, daß der Fehler in der Lokalisierung des Kantenzentrums gerade ein Viertel der Kantenbreite beträgt. Dabei werden ausschließlich die *erfolgreichen* Fälle gewertet, also jene, in denen das Verfahren konvergierte.

Diese Tabelle bestätigt das gewünschte Verhalten des Verfahrens. Für geringe Rauschwerte hängt weder die Lokalisierung noch die Fehlerrate vom Betrag des Rauschens ab, man erhält die gewünschte Subpixelgenauigkeit in der Ortsauflösung bis zu einem SNR von etwa 5.0 .

Bei stärkerem Rauschen treten signifikante Konvergenzprobleme ein. Die Ortsauflösung bleibt akzeptabel, allerdings sinkt die Zahl der erfolgreichen Klassifikationen.

5.1.2 Reale Daten

Das Ergebnis eines Vermessungsauftrags sind Parameter zur Beschreibung von Objekten. Nachfolgend werden Ergebnisse der einzelnen Iterationen für das Beispiel „Bestimme die Position einer Ecke“ aufgeführt:

Parameter	Ecke 1	Ecke 2	Ecke 3	Ecke 4
x_center	58.0	254.4	309.6	353.0
x_center	115.6	181.3	186.3	372.5
direc_1	1.256	2.812	0.750	1.482
direc_2	-1.657	-1.589	-1.413	-3.227
direc_3	-0.261	-1.221	–	1.744
bright_1	74.7	94.9	42.0	93.3
bright_2	101.1	7.6	99.2	92.3
bright_3	13.2	110.8	–	21.5
alpha_1	1.9	1.8	1.7	1.9
alpha_2	2.3	1.9	2.1	1.6
alpha_3	1.7	1.8	–	2.3

Das Ergebnis ist die Beschreibung der Parameter von Ecken. Diese haben die Richtungen `direc_1` bis `direc_3` gegenüber der x-Achse in Bogenmaß, mittlerer Grauwert der Regionen `bright_1` bis `bright_3`, sowie die Zentrumskoordinate (`x_center`, `y_center`) in Pixelkoordinaten. Die Werte `alpha_1` bis `alpha_3` kennzeichnen die Breite der ermittelten Übergänge. alle längenartigen Angaben in diesem Beispiel haben einen Fehler von weniger als 0.2 Pixeleinheiten.

Die mittlere Grauwertabweichung bezeichnet die Grauwertabweichung, die ein einzelnes Pixel des realen Bildes im Mittel von dem ermittelten Modell hat. Im Idealfall eines rauschfreien Bildes verschwindet dieser Wert für ein vollständiges Modell. Bei realen Bildern ist dieser Wert für ein vollständiges Modell identisch mit der durch das Rauschen verursachten Streuung σ der Grauwerte einer Fläche homogener Intensität. Im gezeigten Beispiel tritt ein Wert von etwa 5 aus 256 möglichen Grauwertstufen auf, dieser entspricht dem Mittelwert der auftretenden Rauschwerte. Das Rauschen beträgt in diesem Beispiel 5.4, daher erfolgt die Erkennung der Ecke mit Subpixelgenauigkeit. Dies kann streng nur für simulierte Bilder wie in §5.1.1 nachgewiesen werden, da die Koordinaten durch ein Modell mit *bekannten* Parametern vorgegeben sind. In diesem Fall eines realen Bildes zeigt die optische Begutachtung einer Ausschnittsvergrößerung, daß bei der Erkennung tatsächlich Subpixelgenauigkeit erzielt wurde.

In Abb. 5.2 ist das Ergebnis mehrerer Vermessungsaufträge dargestellt.

Das Ergebnis eines Vermessungsauftrags sind geometrische Parameter. In diesem Fall handelt es sich um die Koordinaten und die Richtungen von Ecken. Diese Richtungen sind in der Abbildung als Pfeile markiert. Die Parameter wurden mit einer gegenüber klassischen Operatoren sehr hohen Genauigkeit erkannt. Speziell die Genauigkeit in der Lokalisierung liegt etwa um einen Faktor 9 über herkömmlichen Verfahren, wie sie in *COSMOS-2D* Verwendung finden. Bei diesem Vergleich muß berücksichtigt werden, das diese klassischen Verfahren weder Annahmen über die Form der Intensitätsübergänge, noch andere Formen der

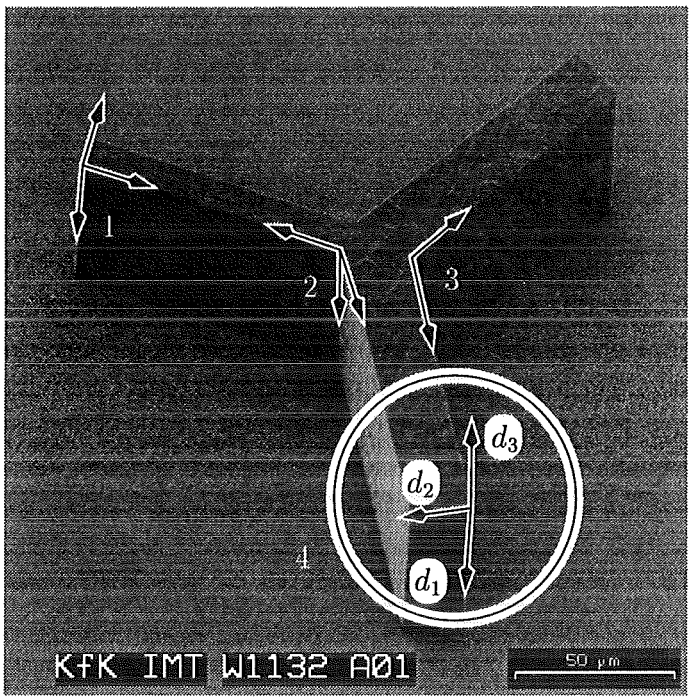


Abbildung 5.2: Eckenerkennung bei einer realen Struktur. Es handelt sich um eine mit einem Rasterelektronenmikroskop (REM) gewonnene Aufnahme einer LIGA- *Struktur auf einer Substratgrundfläche. Die Struktur wurde am IMT gefertigt. Die Paare bzw. Tripel von Pfeilen markieren erkannte Ecken und die ermittelten zugehörigen Richtungen. Die drei oben erkannten Ecken wurden mit der gewünschten Genauigkeit erkannt. Für die durch den Kreis markierte Ecke wurde eine auslaufende Kante mit einem fehlerhaften Richtungswert ermittelt. Dies ist die Folge der fehlenden Integration textureller Merkmale bei dem verwendeten Verfahren.*

Modellbildung machen, wie etwa über den Intensitätsverlauf im Eckenzentrum.

Die Abbildung enthält einen problematischen Fall, welcher durch einen Kreis markiert ist. Es handelt sich um eine Ecke, die zwar als solche erkannt wurde, jedoch einen inakzeptablen Fehler hinsichtlich der erkannten Richtung der auslaufenden Ecke d_3 aufweist. Die Erkennung der Ecke konnte hier dadurch erfolgen, daß zumindest zwei der drei Kanten klar ermittelt wurden. Der Grund für die fehlende Erkennung der dritten Kante wird in §4.1 eingehend diskutiert. Es ist der Ausgangspunkt in dieser Arbeit gewesen, texturelle Merkmale bei der Erkennung zu berücksichtigen.

5.2 Modell mit texturellen Merkmalen

Hier wird nun der Fall beschrieben, daß eine Unterstützung der Kantenerkennung durch Verwendung textureller Merkmale notwendig wird.

Abbildung 5.4 zeigt die Entwicklung der Fehlerrate für die ermittelte Bestarchitektur. Die Fehlerrate ist der gemittelte Fehler zwischen der Netzausgabe und dem Sollwert über alle auftretenden Muster.

Der Begriff der Bestarchitektur entspringt der Frage nach der optimalen Topologie eines neuronalen Netzes. Das Grundprinzip für das hier behandelte Problem der Klassifikation textureller Merkmale zeigt Abb. 5.3.

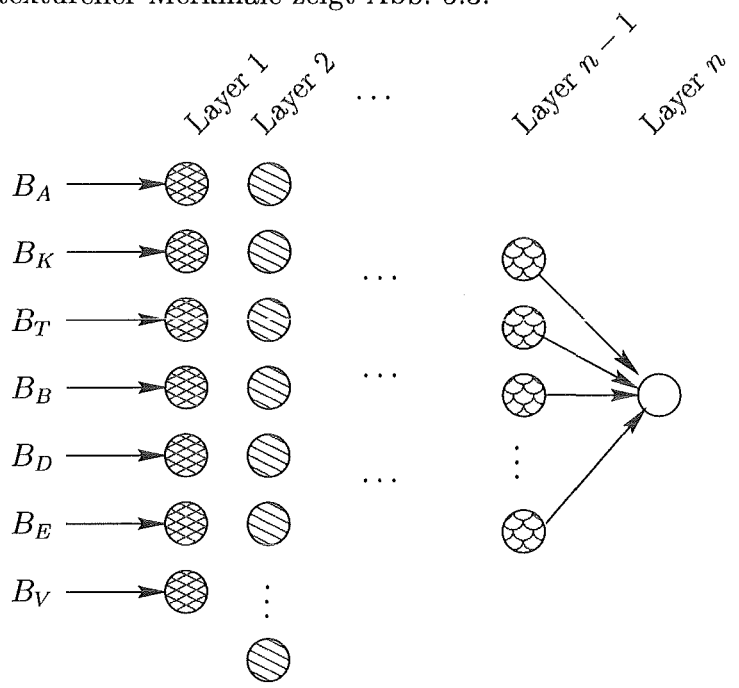


Abbildung 5.3: Beispiel eines Netzes zur Texturklassifikation. Als Eingangssignale werden die aus den Co-occurrence Matrizen ermittelten Merkmale von Gleichung C.4 - C.10 verwendet.

Es gibt allerdings für solche Netze vom Backpropagation Typ keine Konstruktionsvorschrift zur Ermittelung der Anzahl n der benötigten Layer und deren Knotenanzahl für ein gegebenes Problem. Es existieren lediglich Iterationsvorschriften, wie ausgehend von einer bestimmten „Startarchitektur" anhand der betrachteten Fehlerentwicklung sukzessive Neuronen und Layer hinzugefügt oder weggelassen werden können, um auf diese Weise Verbesserungen zu erzielen. Das Ziel besteht immer darin, eine Reduzierung der Fehlerrate auf einem Trainingssatz zu erzielen. Diese Reduzierung *muß* auch auf den Daten des gleichen Problems auftreten, welche *nicht* zum Trainingssatz gehören. Ist dies nicht der Fall,

hat das Netz zwar die Trainingsdaten gelernt, es fehlt aber die Fähigkeit zur Verallgemeinerung.

Im vorliegenden Fall wurden iterativ verschiedene Netzarchitekturen wie in Abb. 5.3 getestet. Dabei zeigte sich, daß für das betrachtete Problem 4 die folgenden der in Gleichung C.4 beschriebenen Merkmale ausreichten, um eine Klassifikation zu erzielen:

- Kovarianz B_K
- Kontrast B_T
- Energie B_E
- Entropie B_V

Das Netz hat daher 4 Neuronen in der Eingangsschicht. Als beste unter den getesteten Architekturen hat sich ein Netz mit zwei hidden Schichten mit jeweils 24 Neuronen erwiesen, das Verhalten dieser „Bestarchitektur" zeigt Abb. 5.4.

Das eingezeichnete Histogramm zeigt die Verteilung der Fehler für Daten, welche nicht zum Trainingssatz gehören. Man erkennt, daß für 98% der Muster die Fehler kleiner als 10^{-2} sind. Dies ist für die hier betrachteten Zwecke ausreichend.

Die Ergebnisse dieser Texturanalyse wurden benutzt, um problematische Objekte wie in Abb. 5.2 zu erkennen. Im genannten Beispiel konnte die fehlende Richtung eindeutig bestimmt werden. Dabei änderte sich der Fehler in der Genauigkeit der Lokalisierung einer Ecke nur unwesentlich, wie erwartet ergab sich aber eine sehr viel höhere Genauigkeit in der Erkennung der fehlerhaften Richtung. Dies ließ sich auch in dem aus der Regression ermittelten Fehlerintervall erkennen.

5.3 Zeitbedarf

Der Zeitbedarf für die Erkennung der hier vorgestellten Objekte beträgt im Mittel etwa 2.5 Sekunden pro erkanntem Objekt. Die Zeit wird bestimmt durch das Konvergenzverhalten und ist daher stark abhängig von den gewählten Startwerten.

Die Ausführung eines Meßauftrages in einem Einzelbild bei *COSMOS-2D* dauert typischerweise etwa 5 Sekunden.

Der direkte Vergleich der Leistungsfähigkeit ist allerdings sehr problematisch aufgrund folgender Randbedingungen:

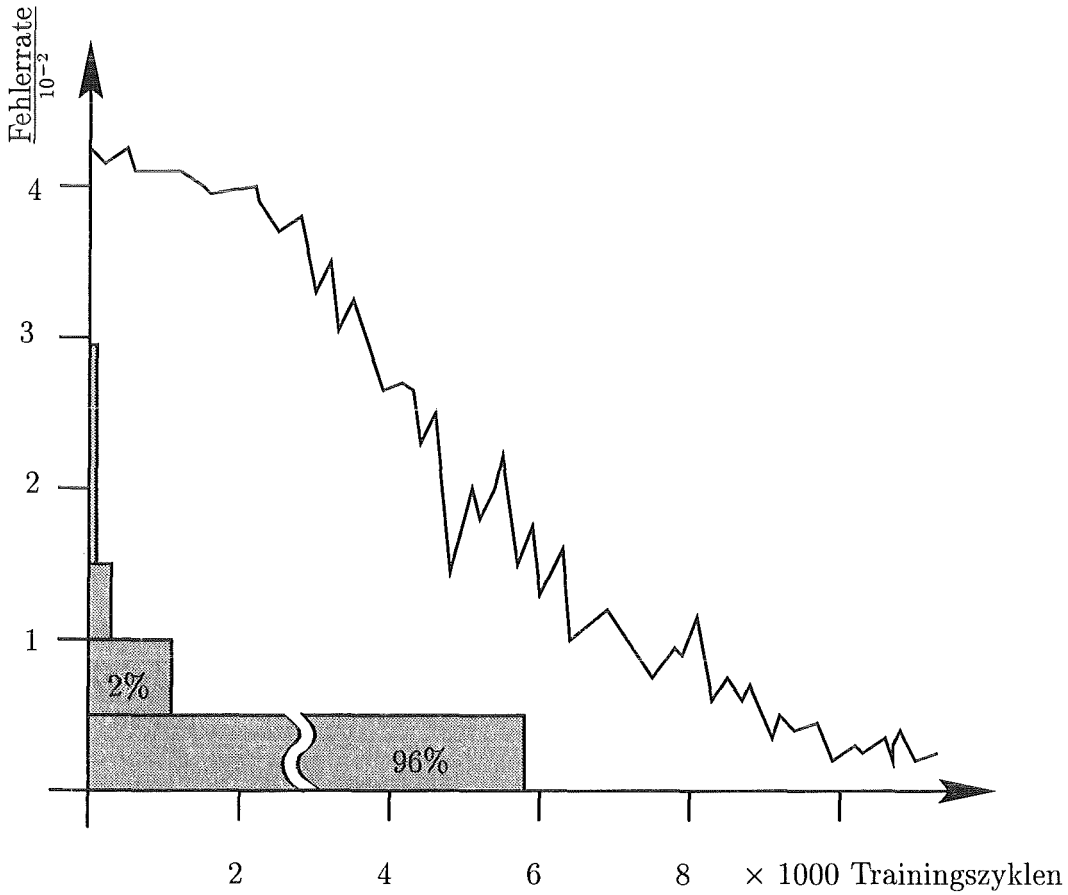


Abbildung 5.4: Fehlerentwicklung für 4-24-24-1 Architektur. Die Kurve zeigt den mittleren Fehler auf einem Datensatz, welches nicht *zum den Trainingsdaten gehört. Das grau hinterlegte Histogramm zeigt die Verteilung der Klassifikationsfehler. Man erkennt, daß die Fehlerrate in 98% aller Fälle kleiner als* 10^{-2} *ist.*

1. *COSMOS-2D* verwendet Spezialhardware zur Kantenextraktion und Mustererkennung. Die Kantenextraktion für ein ganzes Bild erfolgt daher in Video- Echtzeit. Das hier vorgestellte Verfahren läuft hingegen auf einer Workstation.

2. Die erreichbare Genauigkeit der Ortsauflösung des vorgestellten Verfahrens ist höher als bei *COSMOS-2D*.

3. Die von *COSMOS-2D* verarbeiteten Meßaufträge haben eine höhere Komplexität als der hier betrachtete Fall „Distanz zweier Ecken".

Kapitel 6

Zusammenfassung und Ausblick

Die vorliegende Arbeit beschäftigt sich mit zwei Problemkreisen:

- Probleme von *COSMOS-2D* auf der Ebene der Bildverarbeitung und Mustererkennung.
- Notwendige Verbesserungen der Eckenerkennung zum Einsatz in einem 3D - Vermessungssystem.

Für ein 3-D Vermessungssystem besteht die Notwendigkeit einer sehr präzisen Lokalisierung erkannter Objekte, um eine Rekonstruktion brauchbarer Genauigkeit der zugrundeliegenden dreidimensionalen Szene zu ermöglichen.

Das Problem der Kantenerkennung wurde zunächst durch Einführung eines auf einem neuronalen Netz basierenden Kantenfilters angegangen, um eine direkte Erkennung zu erreichen. Das Ziel dabei bestand in einer invarianten Erkennung gegenüber der Bewegungsgruppe und Skalierungen.

Es zeigte sich, daß das untersuchte Verfahren keinerlei Verbesserung gegenüber herkömmlichen Kantenoperatoren brachte. Die mögliche Ursache liegt in der notwendigen Beschränkung auf zu kleine Umgebungen eines Pixels. Wegen deutlicher Überschreitung von Ressourcengrenzen konnte eine Untersuchung größerer Umgebungen nicht durchgeführt werden. Aus diesem Grund wurde versucht, Ecken direkt ohne den Umweg über die Kantenerkennung zu erkennen. Dies geschieht im Hinblick auf die Tatsache, daß Ecken generell einen sehr hohen Informationsgehalt bei der Erkennung von Bildelementen haben.

Aus der Literatur sind Verfahren bekannt, die in der Regel massive Einschränkungen an die zu erkennenden Ecken auferlegen. Beispielsweise beschränken sich viele Vorschläge auf die Erkennung von Ecken, die von trihedralen Objekten stammen, und daher genau drei Kanten besitzen.

Der Ausgangspunkt der Überlegungen gegenüber herkömmlichen Verfahren bestand daher in der Tatsache, daß in der Erkennungsaufgabe ein massives Vorwissen über die geometrische Sollstruktur der betrachteten Objekte vorhanden ist. Daher wurde ein modellbasierter Ansatz für die zu erwartenden *Intensitätsverteilungen* gewählt, wobei ein Teil der Modellparameter aus dem Vorwissen des CAD Entwurfes über die gefertigte Struktur stammt.

Es wurden Erkennungen sowohl für simulierte Bilder, als auch für reale Szenen durchgeführt. Wie in §5.1.1 und in §5.1.2 dargestellt, war das Ergebnis war für eine große Zahl untersuchter Objekte positiv, die gewünschte Genauigkeit für die Lokalisierung im Subpixelbereich wurde erreicht. Allerdings tauchten bei bestimmten Typen aus realen Szenen Probleme auf. Die Objekte wurden zwar erkannt, allerdings konnte nur ein Teil der gesuchten Parameter korrekt bestimmt werden.

Eine genauere Betrachtung zeigte, daß diese Probleme durch die Beschränkung auf ein reines *Intensitätsmodell* verursacht wurden. Dadurch konnten Grenzen nicht erkannt werden, wenn sie sich zwar in texturellen Merkmalen, nicht jedoch in ihrer gemittelten Grundintensität voneinander unterschieden.

Um diese Problematik zu beheben, wurden zunächst die gängigen Verfahren zur Extraktion textureller Merkmale in bezug auf ihre Eignung in einem modellbasierten Verfahren untersucht. Dies führte zur Verwendung von Co-occurrence Matrizen, aus denen die gewünschten Merkmale extrahiert werden können. Diese Extraktion erfolgt mittels neuronaler Netze.

Im Fall der *LIGA* Technik handelt es sich bei den bislang betrachteten Bildbeispielen um eine begrenzte Zahl von Texturen. Dies kann sich allerdings ändern, da eine Vielzahl von unterschiedlichen Werkstoffen nach unterschiedlichen Verfahren behandelt wird. Es steht daher zu erwarten, daß die Zahl der zu klassifizierenden Texturen zunimmt. Es wurde daher ein neuronales Netz eingesetzt, um die Klassifizierung durchzuführen. Die hat u. a. den Vorteil, daß im Falle neu auftretender Texturen, wie z. B. bei Verwendung anderer Werkstoffe, eine Anpassung des Klassifikators durch erneutes Training in einfacher Weise erfolgen kann.

Mit der so eingeführten Verbesserung entstand ein Mehrkanalverfahren, in dem prinzipiell beliebig viele Merkmale eines Bildausschnittes durch ein entsprechendes Modell beschrieben werden können. Die Ergebnisse zeigten die verbesserte Erkennung, die Rate der nicht erkannten Objekte sank auf unter 2%. Allerdings konnte die inhärente Schwäche texturbasierter Verfahren der geringeren Genauigkeit in der Lokalisierung nicht vollständig vermieden werden. Der mittlere Lokalisierungsfehler fiel zwar aufgrund der erfolgten Mittelung gering aus, lag aber deutlich über dem experimentell ermittelten Wert für ein reines Intensitätsmodell.

Das Fazit besteht in folgender Erkenntnis: Die Betrachtung textureller Merkmale ist notwendig zur Erkennung bestimmter Typen von Objekten. Wo dies notwendig ist, kann grundsätzlich nur eine geringere Genauigkeit in der Lokalisierung erwartet werden. Aus diesem Grund müssen die verschiedenen Merkmalskanäle unterschiedlich gewichtet werden, um maximale Genauigkeit erzielen zu können.

Gegen Ende der Arbeit entstand auf Anregung von Herrn Prof. Dr. Menz die Idee, Texturanalysen auch für andere Probleme im Kontext der *LIGA*- Technik zu verwenden.

Als erstes ist die Inspektion unstrukturierter Oberflächen zu nennen. Die im *LIGA*- Verfahren verwendeten Wafer werden einer Qualitätskontrolle unterzogen, um Produktionsfehler durch fehlerhafte Waferoberflächen auszuschließen. Dies geschieht im Rahmen der Maskenblank- Inspektion. Dabei wird die Waferoberfläche von einem Mikroskop in Bildausschnitten gerastert, wobei für jeden Bildausschnitt eine integrale Qualitätsgröße definiert wird. Diese Größen werden nach einfachen Algorithmen ermittelt. Ein verwendeter Algorithmus erzeugt für jeden Bildausschnitt zunächst ein Kantenbild. Als integrale Größe wird dann die Anzahl der ermittelten Kantenpixel verwendet. Dabei können grundsätzlich folgende Fehler gemacht werden:

- Eine fehlerhafte Waferoberfläche wird als „nicht fehlerhaft " bewertet. (Fehler erster Art)
- Eine nicht fehlerhafte Waferoberfläche wird als „fehlerhaft " bewertet. (Fehler zweiter Art)

Insbesondere die Fehlerraten erster Art müssen sehr gering sein, da sonst ungeeignete Eingangswafer die Ausbeute der Produktion zu stark verringern. Andererseits sollen auch nicht zu viele verwendungsfähige Wafer ausgesondert werden. Aus diesem Grund muß das Qualitätsmaß eines Bildausschnittes eine gute Trennung zwischen produktionsbehindernden und unproblematischen Merkmalen der Oberfläche erzielen. Hier bieten sich texturelle Merkmale an, wie die Beispiele in Abb. 6.1 zweier solcher Bildausschnitte zeigen.

Um eine zufriedenstellende Entscheidung über die Verwendungsfähigkeit eines Wafers treffen zu können, müssen die Ursachen für im Produktionsprozeß auftretende Fehler bekannt sein. Es ist nicht erforderlich, den physikalischen Zusammenhang zwischen Eigenschaften der Waferoberfläche und später im Produktionsprozeß auftretenden Fehlern zu kennen, es wird lediglich eine Vorschrift zur Klassifikation benötigt. Ein Beispiel ist die Galvanik. Durch bislang ungeklärte Ursachen treten im Prozeß der Galvanisierung einer Oberfläche Fehler auf, welche möglicherweise *auch* mit Eigenschaften der unstrukturierten Waferoberfläche

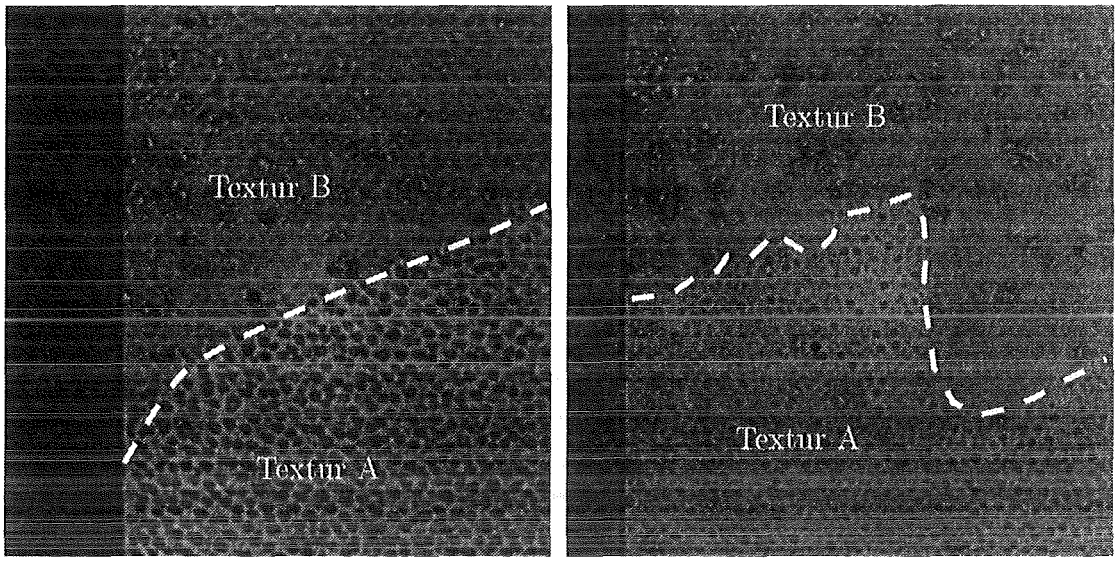


Abbildung 6.1: Zwei Ausschnitte unstrukturierter Waferoberflächen. Man erkennt deutlich die Grenzen zwischen den jeweils zwei unterschiedlichen Texturen A und B.

zusammenhängen. Für die praktische Anwendung ist kein Verständnis dieses physikalischen Kausalzusammenhangs notwendig, es genügt hingegen, die Merkmale zu erkennen, welche zu diesem Fehler führen. Dazu bieten sich neben statistischen Verfahren vor allem Neuronale Netze an, da für deren Anwendung kein explizites Wissen zur Klassifikation benötigt wird.

Anhang A

Mathematische Ergänzungen

A.1 Bildung invariant transformierender Funktionen

In der vorliegenden Arbeit interessieren Invarianzen unter drei auf dem $\mathbb{R}^2$ definierten Transformationsgruppen, siehe auch [RS89]:

- Translationen: $x'_k = x_k + a_k$
- Rotationen: $x'_k = \sum_l \mathbf{D}_{kl} x_l \qquad \mathbf{D}^{-1} = \mathbf{D}^t$
- Skalierungen: $x'_k = \lambda x_k$

Zur Illustration des Konstruktionsprinzips betrachten wir zunächst den einfachen Fall einer eindimensionalen Translation des $\mathbb{R}^1$:

$$x' = x + a; a \in \mathbb{R} \tag{A.1}$$

$\mathrm{f} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ sei eine beliebige Funktion. Invarianz unter Transformationen bedeutet:

$$\forall_{x,y,a \in \mathbb{R}} : \mathrm{f}(x, y) = \mathrm{f}(x + a, y + a) \tag{A.2}$$

Graphisch dargestellt bedeutet dies, daß f auf allen Geraden im $\mathbb{R}^2$ der Steigung 1 konstant ist. Dies bedeutet die Bildung von Äquivalenzklassen im $\mathbb{R}^2$. Die entsprechende Relation lautet:

$$(x, y) \sim (\tilde{x}, \tilde{y}) :\Longleftrightarrow \exists_{a \in \mathbb{R}} : (\tilde{x}, \tilde{y}) = (x + a, y + a) \tag{A.3}$$

Die Äquivalenzklassen lassen sich als Wege im $\mathbb{R}^2$ darstellen:

$$\mathbf{c}(a) = (x_0 + a, y_0 + a); \qquad x_0, y_0 \in \mathbb{R}\text{beliebig} \tag{A.4}$$

Durch eine triviale Reparametrisierung $u = x_0 + a$ schreibt sich Gleichung A.4 als:

$$\mathbf{c}(a \circ u) = (u, u + (y_0 - x_0)) \tag{A.5}$$

Dies gestattet folgende Interpretation: Für jeden festen Wert $y_0 - x_0$ stellt das Bild des Weges eine Hyperebene (=Gerade) des $\mathbb{R}^2$ mit Steigung 1 dar. Alle Punkte dieser Hyperebene gehören zur selben, durch Gleichung A.3 definierten Äquivalenzklasse. f ist daher konstant auf diesen Geraden:

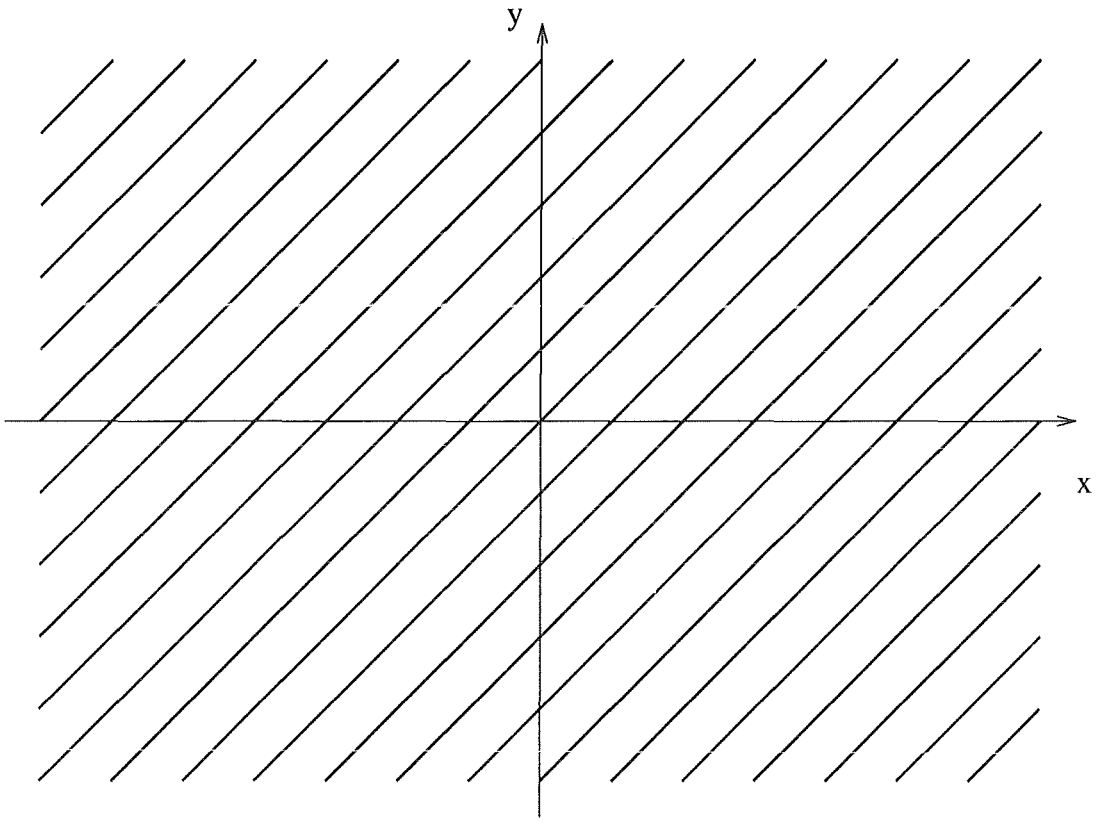


Abbildung A.1: Äquivalenzklassen bei Translation des $\mathbb{R}$

Jeder Punkt des $\mathbb{R}^2$ liegt damit in genau einer solchen Klasse. Betrachten wir nun einen beliebigen Punkt mit Koordinaten (x, y), und wählen als Vertreter der Klasse den Ordinatenabschnitt, so ist letzterer gerade die Differenz $x - y$. Damit kann die allgemeinste Form von f sofort angegeben werden:

$$\mathrm{f}(x, y) = \Phi(x - y) \tag{A.6}$$

$\Phi : \mathbb{R} \longrightarrow \mathbb{R}$ ist dabei eine beliebige Funktion.

Die Form Gleichung A.6 stimmt auch mit dimensionalen Betrachtungen überein: Im $\mathbb{R}^2$ gibt es zwei kontinuierliche Parameter x, y. Durch die Anwendung einer eindimensionalen Transformationsgruppe bleibt davon noch ein frei wählbarer Parameter übrig

Dieses Beispiel ist relativ trivial. Der skizzierte Weg eignet sich aber auch, um kompliziertere Fälle zu betrachten. Als weiteres Beispiel soll die gleichzeitige Invarianz unter Translationen und Skalierungen einer Funktion $\mathrm{f}(x, y, z)$in drei Dimensionen betrachtet werden. Als Parametrisierung analog Gleichung A.4 wählen wir:

$$\mathbf{c}(a, \lambda) = (\lambda(x_0 + a), \lambda(y_0 + a), \lambda(z_0 + a)) \tag{A.7}$$

Mit den Substitutionen $\sigma = \lambda(y_0 - x_0)$ und $q = \frac{x_0+a}{y_0-a}$ geht Gleichung A.7 über in:

$$c \circ (\sigma, q) = \left(\sigma q, \sigma(q+1), \sigma q + \sigma \frac{z_0 - x_0}{y_0 - x_0}\right) \tag{A.8}$$

Für eine invariant transformierende Funktion gilt daher in diesem Fall:

$$\mathrm{f}(x, y, z) = \Phi\left(\frac{z-x}{y-x}\right) \tag{A.9}$$

Φ ist wieder eine beliebige Funktion. Für das in Gleichung 2.7 genannte Transformationsverhalten muß Gleichung A.9 mit einem Vorfaktor versehen werden, um den Faktor $\lambda = \frac{\partial x'}{\partial x}$ zu kompensieren:

$$\mathrm{f}(x, y, z) = \frac{1}{(y-x)^3}\Phi\left(\frac{z-x}{y-x}\right) \tag{A.10}$$

Der Fall einer gleichzeitigen Invarianz unter Translationen, Rotationen und Skalierungen kann wie oben beschrieben für eine Funktion $\mathrm{f}(\mathbf{r}_1, \ldots, \mathbf{r}_n)$, $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^2$, betrachtet werden. Dabei ergibt sich für die allgemeinste Invariante mit Vorfaktor:

$$\mathrm{f}(\mathbf{r}_1, \ldots, \mathbf{r}_n) = \left(\frac{1}{|\mathbf{s}_1|^2 + \ldots + |\mathbf{s}_{n-1}|^2}\right)^n \Phi(\mathbf{q}_1, \ldots, \mathbf{q}_{n-2}) \tag{A.11}$$

wobei:

$$\begin{aligned} \mathbf{s}_i &= \mathbf{r}_{i+1} - \mathbf{r}_1,\ i \in \{1, \ldots, n-1\} \\ \mathbf{q}_j &= \frac{1}{|\mathbf{s}_1|^2 + \ldots + |\mathbf{s}_{n-1}|^2} \begin{pmatrix} \mathbf{s}_1 \mathbf{s}_{j+1} \\ (\mathbf{s}_1 \times \mathbf{s}_{j+1})_z \end{pmatrix},\ j \in \{1, \ldots, n-2\} \end{aligned}$$

Der Ausdruck $(\mathbf{s}_i \times \mathbf{s}_j)_z := \mathbf{s}_{i_x}\mathbf{s}_{j_y} - \mathbf{s}_{i_y}\mathbf{s}_{j_x}$ bedeutet dabei die z -Komponente des Kreuzproduktes der Vektoren $\mathbf{s}_1$ und $\mathbf{s}_j$. Dies ist so zu verstehen, daß eine Einbettung des $\mathbb{R}^2$ in den $\mathbb{R}^3$ vorgenommen wird.

Anhang B

Ausgleichsrechnung

B.1 Allgemeines Ausgleichsproblem

Hier wird zunächst das allgemeine Ausgleichsproblem formuliert und dann eine iterative Lösung durch Rückführung auf eine Folge linearer Probleme gezeigt.

Das allgemeine Problem der Ausgleichsrechnung wird dargestellt durch eine Funktion $\mathrm{f}(p_\lambda; x_1, x_2, \ldots, x_n)$ und Werte q_λ mit $1 < \lambda \leq m$ und $n < m$. Die Größen $(p_\lambda,\ q_\lambda)$ haben dabei eine Interpretation als Meßwertpaare, während die Funktion f das der Messung zugrundeliegende Modell in Abhängigkeit von den Modellparametern x_i darstellt.

Bei einer idealen fehlerfreien Messung gilt dann bei fixen Parametern x_i:

$$q_\lambda = \mathrm{f}(p_\lambda; x_1, x_2, \ldots, x_n) \tag{B.1}$$

Ein Satz realer Datenpaare $p_\lambda,\ q_\lambda$ unterliegt Meßfehlern, so daß auch im Fall eines vollständigen Modelles *immer* Abweichungen zwischen den gemessenen q_λ und den theoretischen Werten $\mathrm{f}(p_\lambda; x_1, x_2, \ldots, x_n)$ bestehen.

Häufig können die Modellparameter selbst nicht direkt durch Messung bestimmt werden. Das Ziel ist es dann, bei *gegebenen* Meßwertepaaren (p_λ, q_λ) die Modellparameter $x_1, \ldots, x_n$ so zu ermitteln, daß der Zusammenhang B.1 *möglichst gut* dargestellt wird. Führt man noch die Schreibweise $f_\lambda(\mathbf{x}) = \mathrm{f}(p_\lambda, \mathbf{x})$ ein, dann läßt sich der Begriff „möglichst gut“ konkretisieren durch die folgenden Normen im Raum $\mathbb{R}^n$ der Parameter $\mathbf{x}$:

$$\|\mathbf{q} - \mathbf{f}(\mathbf{x})\|^2 = \sum_\lambda [q_\lambda - \mathrm{f}(q_\lambda, \mathbf{x})]^2 \tag{B.2}$$

$$\|\mathbf{q} - \mathbf{f}(\mathbf{x})\|_\infty = \max_\lambda |q_\lambda - \mathrm{f}(q_\lambda, \mathbf{x})| \tag{B.3}$$

Es gilt dann, eine solche Norm zu minimieren. In der Regel wird die erste der beiden Normen verwendet. Dies bereits von Gauss betrachtete Verfahren ist bekannt als „Methode der kleinsten Quadrate". Die daraus resultierende Lösung hat besonders einfache statistische Eigenschaften.

Die Norm $\|\ldots\|_\infty$ führt auf das *diskrete Tschebyscheff Problem.* Dieses ist etwas aufwendiger zu ermitteln und wird in dieser Arbeit nicht benützt.

Das Ausgleichsproblem lautet also:

Bestimme die Parameter $x_1, \ldots, x_n$ so, daß für die fest vorgegebenen Paare (p_λ, q_λ) und eine Funktion $\mathrm{f}(p, \mathbf{x})$ der quadratische Fehler

$$\|\mathbf{q} - \mathbf{f}(\mathbf{x})\|$$

minimiert wird. (B.4)

Für eine *beliebige* Funktion f) kann keine allgemeine Aussage über den Lösungsraum getroffen werden. Die oben auftretende Norm $\|\mathbf{q} - \mathbf{f}(\mathbf{x})\|$ definiert ein „Fehlergebirge" also eine n -dimensionale Hyperfläche des $\mathbb{R}^{n+1}$.

Aus diesem Grund können ohne Zusatzannahmen über die Funktion f keine Angaben über die Art des Lösungsraums gemacht werden. Es treten je nach Form des genannten „Fehlergebirges" folgende Fälle auf:

Kardinalität	Beispiel
keine Lösung	Fehler nach unten unbeschränkt
Genau eine Optimallösung	Globales Minimum
Endlich viele Lösungen	Mehrere lokale Minima
Unendlich viele Lösungen	Konstanter Verlauf

Lineares Ausgleichsproblem

Dieses Problem ist analog der Nullstellenberechnung von Funktionen i. allg. nur näherungsweise durch iterative Verfahren lösbar.

Exakt lösbar ist hingegen das lineare Ausgleichsproblem. Dieses ist genau dann gegeben, wenn die Funktion f linear in den Parametern $\mathbf{x}$ ist:

$$\begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix} = \mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} \tag{B.5}$$

In diesem Fall ist folgende quadratische Form zu minimieren:

$$\|\mathbf{q} - \mathbf{A}\mathbf{x}\|^2 = (\mathbf{q} - \mathbf{A}\mathbf{x})^t(\mathbf{q} - \mathbf{A}\mathbf{x}) \tag{B.6}$$

Es gilt der folgende

Satz B.1 *Das lineare Ausgleichsproblem,*

$$\min_{x \in \mathbb{R}^n} \|\mathbf{q} - \mathbf{A}\mathbf{x}\|$$

zu bestimmen, besitzt mindestens eine Lösung $\mathbf{x}_0$. *Ist* $\mathbf{x}_1$ *eine weitere Lösung, so gilt* $\mathbf{A}\mathbf{x}_0 = \mathbf{A}\mathbf{x}_1$. *Das Residuum* $\mathbf{r} := \mathbf{q} - \mathbf{A}\mathbf{x}_0$ *ist eindeutig bestimmt und genügt der Gleichung* $\mathbf{A}^t\mathbf{r} = 0$. *Jede Lösung* $\mathbf{x}_0$ *ist auch Lösung der Normalgleichungen*

$$\mathbf{A}^t\mathbf{A}\mathbf{x}_0 = \mathbf{A}^t\mathbf{y}$$

und umgekehrt.

Die praktische Lösung des Problems erfolgt durch eine Verkettung sogenannter *Householder Transformationen.* Diese sind unitär und ändern daher die gewählte Norm eines Vektors nicht. Die genaue Form soll hier nicht beschrieben werden, da es sich um ein Standardverfahren handelt.

Im Ergebnis erhält man eine unitäre Transformation $\mathbf{U}$ mit:

$$\|\mathbf{q} - \mathbf{A}\mathbf{x}\| = \| \underbrace{\mathbf{U}\mathbf{q}}_{\tilde{\mathbf{q}}} - \underbrace{\mathbf{U}\mathbf{A}}_{\tilde{\mathbf{A}}}\mathbf{x}\|$$

Dabei ist die Matrix $\mathbf{U}$ gerade so gewählt, daß $\tilde{\mathbf{A}}$ mit $n < m$ folgende Form hat:

$$\tilde{\mathbf{A}} = \overbrace{\begin{pmatrix} \mathbf{R} \\ \hline \mathbf{0} \end{pmatrix}}^{n} \begin{matrix} \}n \\ \}m-n \end{matrix} \quad \text{mit} \quad \mathbf{R} = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \end{pmatrix}$$

Bei analoger Aufteilung hat $\mathbf{q}$ die Form

$$\mathbf{q} = \begin{pmatrix} \mathbf{q}_\alpha \\ \mathbf{q}_\beta \end{pmatrix} \text{ mit } \quad \mathbf{q}_\alpha \in \mathbb{R}^n \text{ und } \quad \mathbf{q}_\beta \in \mathbb{R}^{m-n}$$

Insgesamt gilt nun:

$$\tilde{\mathbf{q}} - \tilde{\mathbf{A}}\mathbf{x} = \begin{pmatrix} \mathbf{q}_\alpha - \mathbf{R}\mathbf{x} \\ \mathbf{q}_\beta \end{pmatrix} \begin{matrix} \}n \\ \}m-n \end{matrix}$$

Die Länge $\|\mathbf{q} - \mathbf{A}\mathbf{x}\|$ wird also genau dann minimal, wenn gilt:

$$\mathbf{q}_\alpha = \mathbf{R}\mathbf{x} \tag{B.7}$$

Die Matrix $\mathbf{R}$ ist nichtsingulär genau dann, wenn sie maximalen Rang $= n$ hat. Da auf $\mathbf{A}$ nur unitäre Multiplikationen angewandt wurden, ist der Rang von $\mathbf{A}$ identisch mit dem Rang von $\mathbf{R}$. Daher ist $\mathbf{R}$ invertierbar genau dann, wenn $\mathbf{A}$ maximalen Rang hat, d.h. die Spalten linear unabhängig sind. In diesem Fall existiert ein eindeutiges $\mathbf{x}$, welches das Ausgleichsproblem löst.

Falls $\mathbf{A}$ lediglich den Rang $n-k$ hat, so ist der Lösungsraum ein k - dimensionaler Unterraum des $\mathbb{R}^n$.

Lösung des allgemeinen Ausgleichsproblems

Für die Lösung nichtlinearer Ausgleichsprobleme bietet sich das folgende iterative Verfahren an. Mit $\mathbf{Df}(\xi)$ bezeichnen wir die *Jacobimatrix* der Funktion $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ an der Stelle ξ:

$$\mathbf{Df}(\xi) = \begin{pmatrix} \frac{\partial \mathrm{f}_1}{\partial x_1} & \cdots & \frac{\partial \mathrm{f}_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial \mathrm{f}_n}{\partial x_1} & \cdots & \frac{\partial \mathrm{f}_n}{\partial x_m} \end{pmatrix}_{x=\xi}$$

Aufgrund des *Taylorsatzes* gilt:

$$\mathbf{f}(\mathbf{x}_0 + \mathbf{d}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{Df}(\mathbf{x}_0)\mathbf{d} + \mathcal{O}(\|\mathbf{d}\|)$$

Die entscheidende Aussage dabei ist, daß die Funktion $\mathcal{O}(x)$ von mindestens erster Ordnung gegen 0 strebt, daß also gilt:

$$\lim_{x \to 0} \mathcal{O}(x) = 0$$

Ist daher $\overset{(i)}{\mathbf{x}}$ eine Näherung für die gesuchte lokale oder globale Optimallösung, so können wir das Problem linearisieren:

$$\overset{(i+1)}{\mathbf{x}} = \overset{(i)}{\mathbf{x}} + \lambda \min_{\mathbf{d} \in \mathbb{R}^n} \| \underbrace{\mathbf{q} - \mathbf{f}(\overset{(i)}{\mathbf{x}})}_{\tilde{\mathbf{q}}} - \underbrace{\mathbf{Df}(\overset{(i)}{\mathbf{x}})\mathbf{d}}_{\mathbf{Ad}} \|$$

Dies ist ein lineares Ausgleichsproblem. Der Parameter λ hat die Funktion einer Schrittweite. Eine in dieser Arbeit verwendete Strategie zur Schrittweitensteuerung benützt als Kriterium, bei welchem maximalen $\lambda = 2^{-k}$ sich der Fehler $\|\mathbf{q} - \mathbf{f}(\mathbf{x} + \lambda \mathbf{d})\|$ noch verkleinert.

Anhang C

Texturmerkmale

C.1 Begriffe

Die in [HS92] beschriebenen Merkmale zur Texturerkennung werden durch Relationen definiert. Man betrachtet dazu einen $w \times h$ Ausschnitt eines Bildes. Dieser enthält $w \cdot n$ Pixel p_{ij} mit Grauwerten $g_{ij} \in \{0, \ldots, 2^k\}$, wobei $i \in \{1, \ldots, w\}$ und $j \in \{1, \ldots, h\}$. Der Wert k hat häufig den Wert 8, d.h. es können die Grauwerte $1, 2, \ldots, 256$ dargestellt werden.

Das Symbol $\mathbf{P} = \{p_{ij}\}$ möge die Menge der Pixel bezeichnen. Man betrachtet dann eine Relation $\sim$ auf $\mathbf{P} \times \mathbf{P}$. Dies definiert eine Teilmenge $\mathbf{U} \subset \mathbf{P} \times \mathbf{P}$, nämlich die Menge der Paare von Pixeln, die bzgl. $\sim$ zueinander in Relation stehen.

$$\mathbf{U} = \{(p_{ij}, p_{kl}) \in \mathbf{P} \times \mathbf{P} | p_{ij} \sim p_{kl}\} \tag{C.1}$$

Als Beispiel mag gelten:

$$p_{ij} \sim p_{kl} \Longleftrightarrow (k - i, l - j) = \pm(2, 1) \tag{C.2}$$

In diesem einfachen Beispiel *gehört* ein Pixelpaar zur Relation genau dann, wenn das eine Pixel „schräg rechts oberhalb" des anderen steht, wie in Abb. C.1 veranschaulicht. Alle im Sinne dieser Relation zueinander stehenden Pixel sind durch weiße Doppelpfeile gekennzeichnet.

Im Kontext von Texturanalysen spielen bestimmte geometrische Relationen eine besondere Rolle. Zwei Pixel stehen dabei in Relation zueinander, wenn sie einen Abstand d zueinander haben und der Verbindungsstrahl vom ersten zum zweiten Pixel den Winkel θ mit der Abszisse einschließt.

Man erhält auf diese Weise eine *Familie* von Relationen, die durch den Winkel θ und den Abstand d parametrisiert werden. Das obige Beispiel kann auf diese Weise mit $\theta = \arcsin\left(\frac{1}{\sqrt{5}}\right)$ und $d = \sqrt{5}$ beschrieben werden.

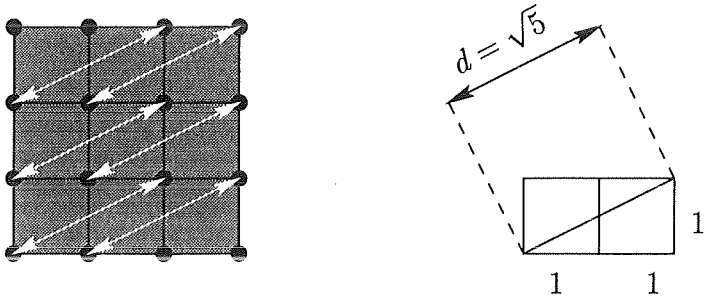


Abbildung C.1: Beispiel für die Relation $\mathbf{p} - \mathbf{q} = \pm(2, 1)$

C.2 Die Co-occurrence Matrix

Eine Relation sagt aus, welche Pixel zueinander in einer gewünschten geometrischen Relation stehen. Damit kann nun die $2^k \times 2^k$ *Co-occurrence Matrix* $\mathbf{C}$ definiert werden:

$$c_{\alpha\beta} = \#\{(p_{ij}, p_{kl}) \in \mathbf{U} | (g_{ij} = \alpha) \wedge (g_{kl} = \beta)\} \tag{C.3}$$

Dabei bezeichnet $\#M$ die Anzahl der Elemente einer Menge M. Das Element $c_{\alpha\beta}$ der Co-occurrence Matrix $\mathbf{C}$ sagt also, wieviele in Relation stehende Pixelpaare es gibt, von denen ein Pixel den Intensitätswert α und das andere den Intensitätswert β hat. Da eine Relation per definitionem symmetrisch ist, ist auch die Co-occurrence Matrix selbst symmetrisch.

Hier wird bereits der Bezug zur Texturanalyse erkennbar. Für eine Umgebung konstanter Helligkeit I ist $c_{\alpha\beta}$ nur für das Element mit $\alpha = \beta = I$ besetzt, alle anderen Matrixelemente verschwinden. Für eine im Verhältnis zur geometrischen Ausdehnung der Relation grobe Textur haben Pixelpaare mit großer Häufigkeit identische Intensitätswerte, daher dominieren die Diagonalelemente von $\mathbf{C}$. Bei einer feinkörnigen Textur schließlich haben die Pixelpaare im Mittel unterschiedliche Intensitätswerte, die Co-occurrence Matrix ist daher gleichmäßig besetzt.

Die Co-occurrence Matrix kann als ein zweidimensionales Histogramm betrachtet werden. Während *das* bekannte Grauwertehistogramm die Häufigkeiten *aller* möglichen Intensitäten der im Bild vorhandenen Pixel wiedergibt, liefert die Co-occurrence Matrix die Häufigkeiten des Auftretens aller möglichen Helligkeitspaare aus einer festen Teilmenge der möglichen Pixelkombinationen.

Die Co-occurrence Matrix hat einen entscheidenden Nachteil zur direkten Verwendung in einem Klassifikator. Sie ist nicht invariant unter Intensitätstransformationen. Zudem ist die Zahl der Merkmale zu groß. Um eine verläßliche Statistik

zu erhalten, muß für jedes Matrixelement von **C** die in Gleichung C.3 auftretende Anzahl ausreichend groß sein. Dies kann sowohl durch eine große Umgebung, als auch durch eine Einteilung der 2^k Intensitäten in eine kleinere Anzahl L von Klassen erfolgen. Bei einer Einteilung in $2^4 = 16$ Helligkeitsklassen enthält die Matrix **C** immerhin noch 256 Elemente.

C.3 Merkmale aus der Co-occurrence Matrix

Aus den Matrixelementen von C lassen sich nun Merkmale zur Verwendung in einem Klassifikator generieren:

Autokorrelation

$$B_A = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} abc_{ab} \tag{C.4}$$

Kovarianz

$$B_K = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} (a-\bar{a})(b-\bar{b})c_{ab} \tag{C.5}$$

Trägheit / Kontrast

$$B_T = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} (a-b)^2 c_{ab} \tag{C.6}$$

Absolutbetrag

$$B_B = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} |a-b| c_{ab} \tag{C.7}$$

Inverser Abstand

$$B_D = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} \frac{c_{ab}}{1+(a-b)^2} \tag{C.8}$$

Energie

$$B_E = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} (c_{ab})^2 \tag{C.9}$$

Entropie

$$B_V = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} c_{ab} \log_2 (c_{ab}) \tag{C.10}$$

Die Co-occurrence Matrix c_{ab} hat dabei die Größe L. $\bar{a}$ und $\bar{b}$ bezeichnen die Mittelwerte der Indizes.

Anhang D

Backpropagation Netze

In dieser Arbeit werden Netze vom Typ Backpropagation zur Klassifikation von Eingabemustern verwendet. Es besteht eine Verwandschaft zu statistischen Verfahren, wo ebenfalls eine Einteilung von Eingabemustern in eine endliche Anzahl von Klassen erfolgt.

Die folgenden Ausführungen beziehen sich auf die in Abb. D.1 verwendete Notation eines neuronalen Netzes mit einem hidden Layer.

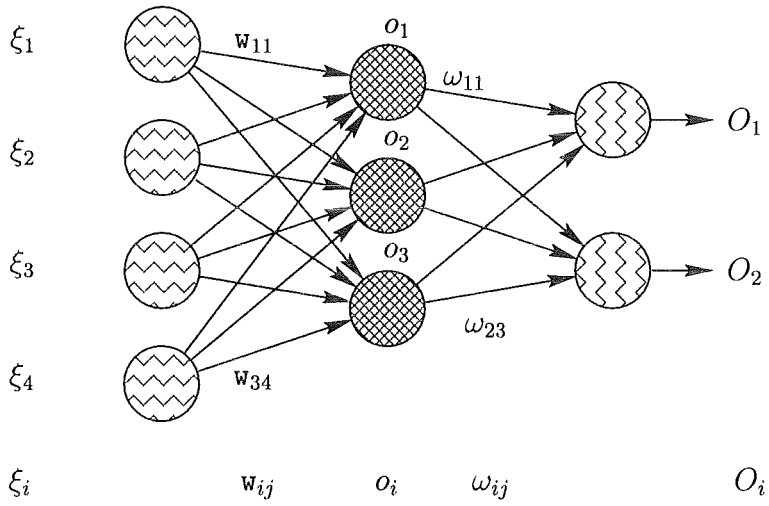


Abbildung D.1: Konventionen zur Notation

Die einzelnen Neuronen bilden den Funktionswert einer Funktion f der Summe der gewichteten Eingangssignale ξ_i wie in Gleichung 2.3. Es soll hier gezeigt werden, auf welche Weise für einen Trainingssatz von Eingabemustern ξ_i^μ mit zugehörigen Sollwerten ζ_i^μ die Gewichte w_{ij} und ω_{ij} bestimmt werden können, damit der Gesamtfehler

$$\frac{1}{2}\sum_{\mu i}[\zeta_i^\mu - O_i^\mu]^2 \tag{D.1}$$

minimal wird.

Für ein gegebenes Muster μ beträgt die Eingangssumme des i-ten Neurons im hidden Layer:

$$a_i^\mu = \sum_j \mathtt{w}_{ij}\xi_j^\mu \tag{D.2}$$

Dieses Neuron liefert daher als Ausgangssignal:

$$o_i^\mu = \mathrm{f}(a_i^\mu) = \mathrm{f}\left(\sum_j \mathtt{w}_{ij}\xi_j^\mu\right) \tag{D.3}$$

Daher erhält das i-te Ausgangsneuron als Eingangssumme:

$$A_i^\mu = \sum_j \omega_{ij}o_j^\mu = \sum_j \omega_{ij}\mathrm{f}\left(\sum_k \mathtt{w}_{jk}\xi_k^\mu\right) \tag{D.4}$$

Dies liefert als Ausgangssignal des Netzes für das μ-te Muster:

$$O_i^\mu = \mathrm{f}\left(\sum_j \omega_{ij}\mathrm{f}\left(\sum_k \mathtt{w}_{jk}\xi_k^\mu\right)\right) \tag{D.5}$$

Unter dem quadratischen Fehlermaß Gleichung D.1 erhält man den Fehler als Funktion der Gewichte:

$$\mathbf{E}(\mathtt{w},\omega) = \frac{1}{2}\sum_{i\mu}\left[\zeta_i^\mu - \mathrm{f}\left(\sum_j \omega_{ij}\mathrm{f}\left(\sum_k \mathtt{w}_{jk}\xi_k^\mu\right)\right)\right] \tag{D.6}$$

Um nun die Gewichte $\mathtt{w}$ und ω so zu bestimmen, daß Gleichung D.6 minimiert wird, bietet sich das Verfahren des Gradientenabstiegs an. Dazu werden partielle Ableitungen nach den Gewichten benötigt. Für die Ausgangsschicht ergibt sich bei einer Schrittweite λ:

$$\Delta\omega_{ij} = -\lambda\frac{\partial \mathbf{E}}{\partial\omega_{ij}} = \lambda\sum_\mu D_i^\mu O_j^\mu \tag{D.7}$$

Wobei

$$D_i^\mu = \mathrm{f}'(A_i^\mu)[\zeta_i^\mu - O_i^\mu] \tag{D.8}$$

definiert wird.

Für die Gewichte $\mathtt{w}_{ij}$ des hidden Layers ergibt sich:

$$\Delta \mathtt{w}_{ij} = -\lambda \frac{\partial \mathbf{E}}{\partial \mathtt{w}[ij]} = \lambda \sum_\mu d_i^\mu o_j^\mu \tag{D.9}$$

Wobei diesmal

$$d_i^\mu = \mathrm{f}'(a_i^\mu) \sum_j \mathtt{w}_{ji} D_j \mu \tag{D.10}$$

gilt.

Für ein allgemeines Netz mit einer beliebigen Anzahl von Schichten lauten die Regeln D.7 und D.9 zum Aktualisieren der Gewichte der Schicht mit Index p:

$$\Delta \overset{(p)}{\omega} = \lambda \sum_{\text{Muster}} \overset{(p)}{d} \times \overset{(p)}{\xi} \tag{D.11}$$

Die d Faktoren ergeben sich allgemein zu:

$$\overset{(p)}{d_i^\mu} = \mathrm{f}'(\overset{(p)}{a_i^\mu}) \sum_j \overset{(p+1)}{\omega_{ji}} \overset{(p+1)}{d_j^\mu} \tag{D.12}$$

Diese letzte Gleichung läßt die Interpretation zu, daß die Signale im Netz vorwärts propagiert werden, die Fehler aber durch die Schichten rekursiv von der Ausgangs- zur Eingangsschicht ermittelt werden. Aus diesem Grund spricht man von *Feed-forward Netzen* oder auch von *Backpropagation.*

Anhang E

Objektorientierter Entwurf

E.1 Grundlagen des Systementwurfes

Der Entwurf des hier vorgestellten Systems basiert auf der Verwendung abstrakter Datentypen. Abstrakte Datentypen werden definiert durch das Angebot von *Services* an den Benutzer. Im vorliegenden Kontext ist der Benutzer ein Programmierer, welcher die bereitgestellten abstrakten Datentypen verwendet.

Ein Objekt `Bild` zur Bilddarstellung bietet dem Benutzer beispielsweise die Services `Visualisierung`, `Grauwert` und `Kontrast`. Die Methode `Visualisierung` ermöglicht die Darstellung auf einem Ausgabegerät, z. B. einem Monitor oder Drucker. `Kontrast` liefert dem Benutzer den Kontrast des gerade betrachteten Bildes und `Grauwert` den individuellen Grauwert eines bestimmten Pixels. In der Sprache des objektorientierten Entwurfs wird in der Regel das Wort *Methode* an der Stelle von *Service* benutzt.

Der entscheidende Punkt bei der Definition eines abstrakten Datentyps ist das Weglassen der Implementationsdetails, diese bleiben dem Anwender verborgen. Beispielsweise kann die interne Speicherung eines Bildes durch Verwendung eins Quadtree Algorithmus, wie in [SRSW84] beschrieben, sowohl die Effizienz des Zugriffs verbesserb als auch den Speicherbedarf reduzieren. Dadurch ist es in diesem Beispiel möglich, nachträglich die Implementation von `Bild` zu verändern, *ohne* entsprechende Korrekturen in einer mit `Bild` erstellten Anwendung vorzunehmen. Dies funktioniert, solange die Schnittstelle nicht verändert wird.

In der Regel entsteht im Lauf der Entwicklung eines Gesamtsystems die Notwendigkeit, Services bestehender Datentypen zu erweitern, was eine Erweiterung der Schnittstelle beinhaltet. Für diesen Fall steht in einem objektorientierten Entwurf das Mittel der *Vererbung* zur Verfügung. Durch Vererbung ist es möglich, einen neuen Typ zu erzeugen, welcher die Services des Basistyps enthält und um die

Erweiterungen ergänzt wird. Bei einem guten Design entfällt dann der Zwang zur Reimplementation, im idealen Fall müssen nur die neu hinzukommenden Services ergänzt werden.

Im obigen Beispiel könnte ein Datentyp `Bildkanten` die Methoden von `Bild` erben. Eine zusätzliche Methode `Kantenerzeugung` könnte intern die Methode `Grauwert` der *Basisklasse* `Bild` in der Implementierung benutzen, um aus einem Grauwertbild Kanten zu extrahieren. Der Typ `Bildkanten` benötigt dazu selbst keinen Zugriff auf die Implementationsdetails von `Bild`. Man spricht von `Bildkanten` als einem von `Bild` abgeleiteten Datentyp.

Eine Einführung in die Details der formalen Spezifikation abstrakter Datentypen findet sich in [Mey88].

E.2 Objektorientierte Programmierung und Effizienz

Zu den Vorteilen eines objektorientierten Entwurfes zählen:

- Wiederverwertbarkeit erstellter Software.
- Einfache Erweiterbarkeit.
- Gute Wartungseigenschaften.

Ein entscheidender Einwand betrifft das Laufzeitverhalten. Objektorientierte Implementierungen sind i. allg. langsamer als konventionell erstellte Programme. Die Ursache dafür liegt in zusätzlich zur Verfügung stehenden Konzepten, etwa der Möglichkeit, Methoden und Attribute zu vererben. Man betrachte folgende Vererbungsstruktur: Figur sei eine abstrakte Basisklasse, in der bereits eine Methode *plotFigur* definiert wird. Diese Methode wird dann in jeder konkreten Klasse redefiniert. Zur Laufzeit soll nun für ein konkretes Objekt aus dieser Vererbungshierarchie ohne a priori Kenntnis der Klasse die richtige Methode *plotFigur* ausgeführt werden (Polymorphie). Um dies zu ermöglichen, muß der Compiler für das Objekt zusätzlichen Speicherplatz belegen, um die Klassenzugehörigkeit als Attribut hinzufügen zu können. Diese Zusatzinformation wird dann benützt, um die korrekte Methode *plotFigur* auszuwählen (dynamisches Binden) und aufzurufen, was Laufzeitverluste bewirkt.

Diesen Nachteilen stehen verschiedene Vorteile gegenüber. So können neue Klassen hinzugefügt werden, ohne an den bereits bestehenden Klassen Veränderungen

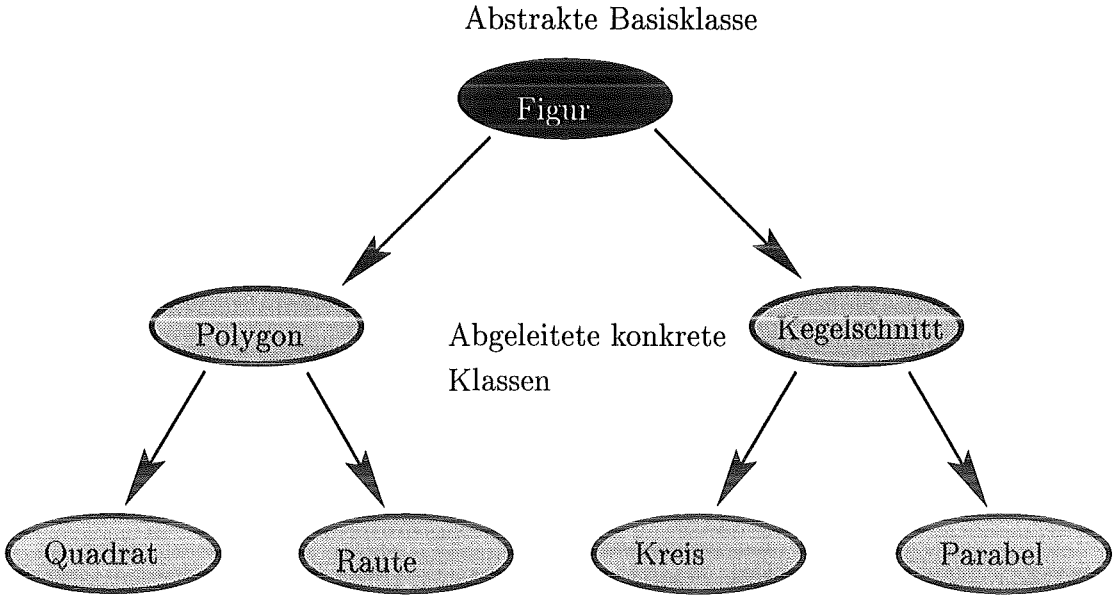


Abbildung E.1: Beispiel einer Vererbungshierarchie

vornehmen zu müssen, wie es bei prozeduraler Programmierung mittels unterschiedlicher Tag's notwendig wäre. Eine Klasse, die in einem bestehenden Entwurf massiv verwendet wird, kann trotzdem erweitert werden, indem von ihr eine Klasse abgeleitet wird. In dieser abgeleiteten Klasse können dann notwendige Erweiterungen implementiert werden, ohne den bestehenden Entwurf anzutasten. Alle so definierten Objekte haben dann zumindest die durch eine gemeinsame Wurzelklasse definierte Schnittstelle gemeinsam, was die Wiederverwertbarkeit erleichtert.

Eine detaillierte Diskussion des objektorientierten Ansatzes findet sich in [Lip90]. Eine ausgezeichnete Referenz ist [Mey88].

E.3 Persistenz

Persistenz bezeichnet die Möglichkeit, verwendete Daten dauerhaft, d.h. über die Verwendung innerhalb eines Programms hinaus, zu speichern. Dies kann prinzipiell auf verschiedenen Ebenen erreicht werden. Beim Entwurf von Klassen sollte hinsichtlich der Wiederverwertbarkeit auf eine Schnittstelle zur Erzielung von Persistenz in einem Gesamtsystem geachtet werden. Im konkreten Fall soll zu einem späteren Zeitpunkt die Integration sämtlicher Werkzeuge in eine Wissensbasis erfolgen. Dies erfolgt über eine objektorientierte Datenbank, daher wird für diesen Zweck eine Schnittstelle bereitgestellt, um Persistenz der verwendeten

Objekte zu ermöglichen.

Innerhalb des objektorientierten Paradigmas kann ein Objekt eine Methode besitzen, um Persistenz zu erreichen. Die folgenden Ausführungen beziehen sich teilweise auf C++ Syntax und Semantik, können aber analog auch in anderen objektorientierten Sprachen verwendet werden.

Als Beispiel soll eine Klasse *Bild* dienen. Diese Klasse benötigt verschiedene Attribute *Breite, Höhe, ...* Die Darstellungsform der Diagramme wird in [CY91] definiert. Um Persistenz zu erzielen, erhält diese Klasse zwei Methoden *saveGuts* und *restoreGuts*, welche die internen Attribute auf einen Stream schreiben, bzw. von einem Stream einlesen:

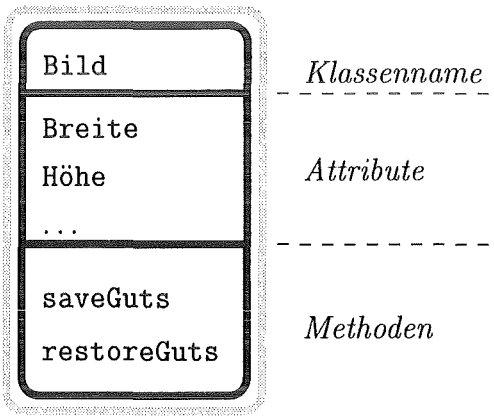


Abbildung E.2: Die Klasse Bild

Die so definierte Benutzerschnittstelle leistet das Gewünschte:

```
...
ostream outStream(persistenteDatei);
Bild dieserTag;
...                                    // Manipulationen
...                                    // des Objektes
...                                    // dieserTag.
...
dieserTag.saveGuts(outStream);         // Objekt ist persistent.
...
istream inStream(persistenteDatei);
dieserTag.restoreGuts(inStream);       // Wiederherstellung.
```

In diesem trivialen Fall ist damit bereits alles getan. Problematischer ist die Situation, wenn das betrachtete Objekt Referenzen auf andere Objekte enthält. Zur Illustration dient eine Klasse *Operator* mit einer Referenz auf zwei Bildobjekte.

Dies ist der Entwurf für einen binären Operator, der zwei Objekte der Klasse Bild als Operanden hat. In diesem Fall kann das Problem der Persistenz durch

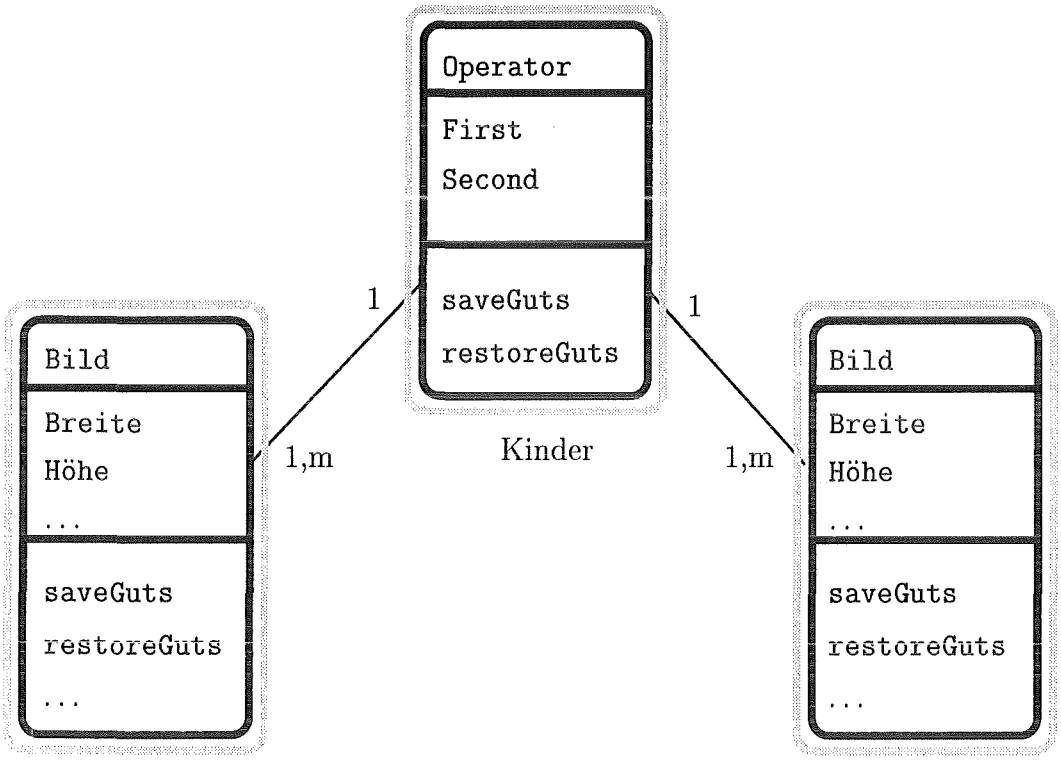


Abbildung E.3: Baumartiger Graph von Referenzen

rekursives Aufrufen der Methode *saveGuts* gelöst werden. Jedes Objekt speichert über *saveGuts* seine eigenen Attribute auf den Ausgabestream. Danach veranlaßt es alle referenzierten Objekte vermöge *saveGuts* ebenfalls deren Attribute auf den Stream zu schreiben. Diese Rekursion endet, sobald alle Blätter des Baumes erreicht wurden. Das Lesen erfolgt ebenfalls rekursiv in genau der umgekehrten Reihenfolge.

Das Schema der Referenzen in Abb. E.3 stellt den Spezialfall eines gerichteten Graphen ohne Zyklen dar, eben einen Baum. Die Objekte werden dabei durch die Knoten und die Referenzen durch die Kanten des Graphen dargestellt.

Das eigentliche Problem taucht nun auf, wenn der Graph der Referenzen Zyklen aufweist: Es handelt sich um die selbe Problematik wie beim Anlegen tiefer Kopien eines Objektes, dem *deepCopy*. Das simple rekursive Aufrufen von *saveGuts* führt in diesem Fall zu einer Endlosrekursion.

Ein möglicher Ausweg wird in [Jos94] beschrieben. Im Prinzip wird dabei festgehalten, ob ein Knoten während eines Speichervorganges bereits besucht wurde

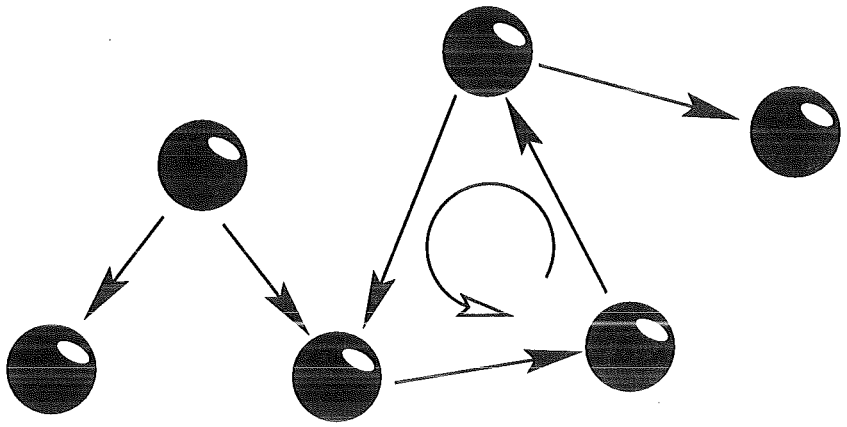


Abbildung E.4: Graph von Referenzen mit einem Zyklus

oder nicht. In diesem Fall endet die Rekursion.

Dieses Verfahren kann unabhängig von einer bestimmten Klasse implementiert werden, lediglich die Methoden *saveGuts* und *restoreGuts* müssen in der konkreten Klasse definiert werden. Aus diesem Grund kann der Mechanismus zur Erzielung von Persistenz in einer Basisklasse implementiert werden, wobei die Member *saveGuts* und *restoreGuts* als virtuelle Funktionen definiert werden. Die hiervon abgeleiteten konkreten Klassen redefinieren dann diese beiden Member wie folgt:

- Aufruf von *saveGuts/restoreGuts* der Basisklasse
- Sichern der eigenen Attribute
- Aufruf von *saveGuts/restoreGuts* für referenzierte Klassen.

Da die Basisklasse selbst keinen weiteren Zweck erfüllt, wird sie als abstrakte Klasse definiert. Insgesamt ergibt sich für das Beispiel Abb. E.3 folgendes Schema:

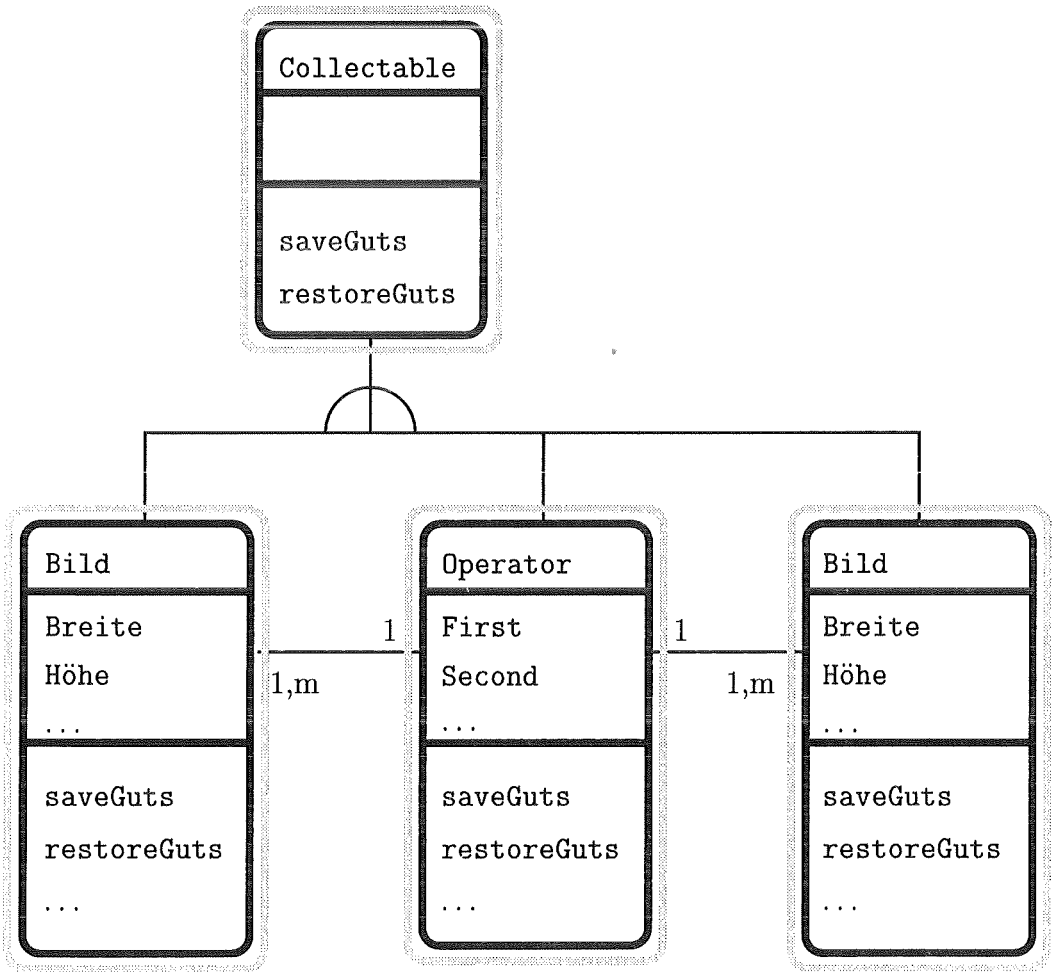


Abbildung E.5: Persistenz über Definition einer Basisklasse

E.4 Copy on Write

Objekte eines Entwurfes benötigen in der Regel eine Methode zur Erzeugung von Kopien und zur Wertzuweisung. Der naheliegende Ansatz ist, in beiden Fällen tatsächlich alle benötigten Attribute eines Objektes A in ein Objekt B zu duplizieren. Bei den in dieser Arbeit benötigten Bildobjekten kann der dadurch entstehende Speicherbedarf kritisch sein.

Oft werden die so kopierten Daten zwar verwendet, aber nicht mehr verändert. Aus einem Bild B wird beispielsweise ohne Veränderung von B eine Liste von Kantenpixeln erzeugt. Solange B nicht verändert wird, kann es daher seine Bildinformation mit A teilen.

Evident wird dieses Problem, wenn Listen von Objekten betrachtet werden. Der Eintrag in eine Liste kann Referenz- oder Wertesemantik haben.

Referenzsemantik hat zunächst einen Laufzeitvorteil, da keine internen Attribute dupliziert werden müssen. Ein grundsätzlicher Nachteil betrifft die Lebensdauer des referenzierten Objektes. Falls das Objekt gelöscht wird, enthält die Liste eine Referenz auf ein nicht mehr existentes Objekt. Dies sollte bei sorgfältigem Entwurf nicht passieren, führt aber in der Praxis oft zu Systemabstürzen.

Wertesemantik vermeidet dieses Problem. Beim Eintrag in eine Liste wird über einen Konstruktor eine Kopie des Objektes erzeugt. Hier taucht wieder die eingangs dargestellte Problematik auf: Der Overhead durch Kopieren. Die Lebensdauer des Objektes ist dafür mit der Lebensdauer der Liste identisch, da die enthaltenen Objekte durch den Destruktor der Liste gelöscht werden.

Als Ausweg verbindet die Technik des *Copy-on-write* die Vorteile der Wertesemantik mit der Vermeidung von *unnötigen* Kopievorgängen. Ein Teil der hier beschriebenen Techniken ist in [Hei94] beschrieben.

Solange ein Objekt beispielsweise nur in eine Liste *eingetragen* wird, brauchen die enthaltenen Daten nicht kopiert zu werden. Dies geschieht erst bei Bedarf, nämlich dann, wenn über die Objektschnittstelle eine Veränderung der Daten erfolgt. Das Prinzip des *Copy-on-write* zeigt Abb. E.6.

Die Daten des Objektes A werden in einer eigenen Datenstruktur verwaltet und um ein zusätzliches Attribut *ref* erweitert, welches die Zahl der Referenzen enthält. Das Objekt selbst erhält eine Referenz auf diese Struktur. Bei einer Zuweisung von A an ein Objekt B geschehen zwei Dinge:

- Das Objekt B erhält eine Referenz auf die Datenstruktur.
- *ref* wird um eins erhöht.

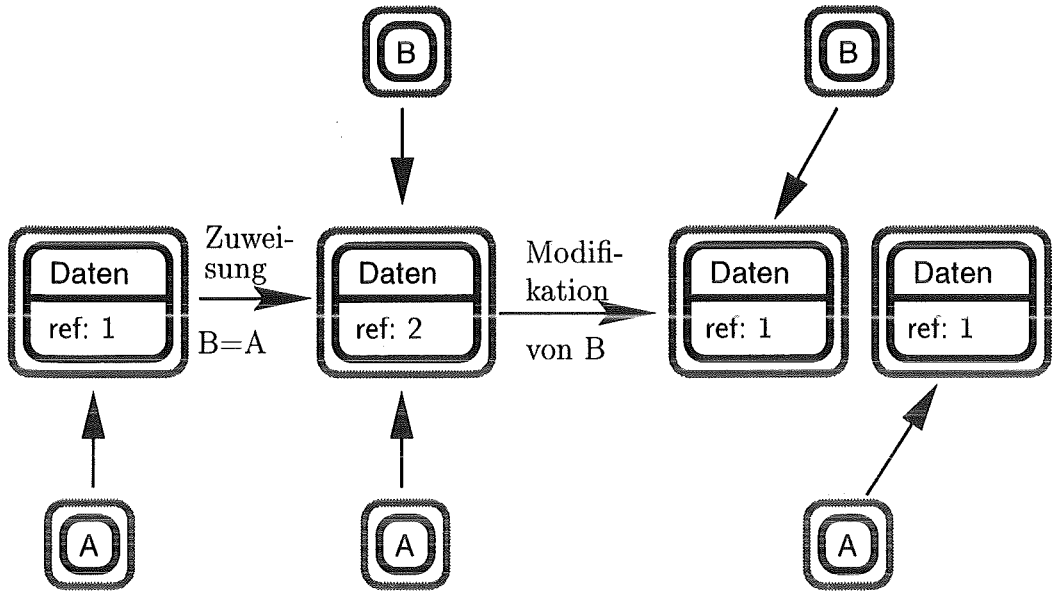


Abbildung E.6: Prinzip des Copy-on-write

A und B teilen sich nun den Datensatz. Falls etwa das Objekt B versucht, die Daten zu ändern, so erfolgt zunächst über *ref* ein Test, ob weitere Referenzen existieren. Dies ist in diesem Beispiel der Fall, daher werden die Daten kopiert, bevor eine Veränderung erfolgt.

Die Einführung eines zusätzlichen Attributs für die Anzahl der Referenzen bedeutet im Fall der Datenstruktur eines Bildes einen sehr geringen zusätzlichen Overhead.

Auf diese Weise lassen sich *Sichten* erzeugen, wie sie bei Datenbanken auftreten. Sichten entstehen durch Einschränkung eines Datensatzes auf eine Untermenge. Beispielsweise kann dies eine Bildzeile aus einem Gesamtbild sein. Ein Objekt B kann wie in Abb. E.6 eine Sicht auf die in A enthaltenen Daten realisieren. Dies erfolgt durch zusätzliche Attribute in B für die Einschränkungen, welche die Sicht definieren.

Die Funktionalität des *Copy-on-write* kann analog zur Persistenz in einer Basisklasse implementiert werden, und für konkrete Objekte durch Vererbung zur Verfügung gestellt werden.

E.5 Speicherverwaltung

Die Technik des *Copy-on-write* beinhaltet ein clean-up nicht mehr benötigter Objekte. Dies geschieht nach der Methode „*Der Letzte macht das Licht aus*". Wenn der Destruktor des letzten auf die Datenstruktur zugreifenden Bildobjektes aufgerufen wird, prüft dieser die Zahl der Referenzen, ermittelt den Wert eins (Die des eigenen Objektes), und gibt den Speicherbereich frei. Dies bedeutet, daß die Objekte selbst für das Memorymanagement zuständig sind. Diese Technik versagt bei beliebigen Objekten, wenn Referenzzyklen auftreten. dies ist aber in diesem Fall ausgeschlossen, da die Datenstruktur selbst keine weiteren Referenzen enthält.

Unter der Sichtweise eines streng objektorientierten Designs bedeutet dieser Weg allerdings eine Prinzipienverletzung. Der durch nicht mehr erreichbare Objekte belegte Speicher wird in einer streng objektorientierten Sprache durch einen Mechanismus des Laufzeitsystems, einem sogenannten *Garbage Collector*, erkannt und automatisch deallokiert (Mark - and - sweep Collector). Dies geschieht im Standardfall ohne zusätzliche Maßnahmen im Systementwurf und ist beispielsweise bei den Sprachen *LISP* und *Eiffel* Teil des Kernsystems.

Im Fall von C++ ist *Garbage Collection* kein Bestandteil des Laufzeitsystems, und die Implementierung eines Kollektors wird durch die Sprache auch nicht unterstützt. Der Verzicht auf C++ andererseits zugunsten einer wirklich objektorientierten Sprache erschien aufgrund der großen Vielfalt existierender Bildverarbeitungssoftware nicht sinnvoll. Aus diesem Grund wurde bei dem hier vorgestellten System das *Memory Management* auf konventionelle Weise via expliziter Deallokation durch Destruktoren realisiert. Dies bedeutet, daß auf der Client Seite, also dem Anwender der Objekte, keine Maßnahmen getroffen werden müssen, außer natürlich das clean-up des Objektes selbst im Fall einer dynamischen Allokation auf dem Heap Speicher.

Eine grundlegende Diskussion zur Frage von Garbage Collection im Allgemeinen und einer Erweiterung des C++ Sprachstandards bezüglich *Garbage Collection* findet sich in [ED93].

Literaturverzeichnis

[BGH89] B. Bürg, H. Guth und A. Hellmann. Bildanalytische Qualitätskontrolle in der Mikrofertigung: Ein vollautomatisches, hochflexibles und schnelles Strukturmeßsystem. In *DAGM-Symposium, Hamburg*, Jgg. 219 of *Informatik-Fachberichte*, Seiten 168–172. Springer-Verlag, 1989.

[BGH91a] B. Bürg, H. Guth und A. Hellmann. COSMOS-2D: Ein System zur Verifikation und Vermessung von zweidimensionalen geometrischen Formen. *KfK-Nachrichten*, 23(2-3):100–109, 1991.

[BGH91b] B. Bürg, H. Guth und A. Hellmann. Parametric Optical Measurement of Micromechanical Structures with arbitrary plane Surface Geometries: The COSMOS-2D System. In H. Reichl, Hrsg., *Micro Systems Technologies '91*, Seiten 280–287. VDE-Verlag, Berlin, Februar 1991.

[BGH91c] B. Bürg, H. Guth und A. Hellmann. Vollautomatische Vermessung von ebenen Oberflächen mit beliebiger Form, Unveröffentlichter Bericht, 1991.

[BM91] C. Burbaum und J. Mohr. Herstellung von mikromechanischen Beschleunigungssensoren in LIGA-Technik. KfK-Bericht 4859, 1991.

[Bür91a] B. Bürg. Parametrisches optisches Messen bei der Herstellung von Mikrostrukturen mit beliebiger, ebener Oberflächengeometrie. Dissertation, Universität Karlsruhe, Fakultät für Maschinenbau, 1991.

[Bür91b] B. Bürg. Parametrisches optisches Messen bei der Herstellung von Mikrostrukturen mit beliebiger, ebener Oberflächengeometrie. KfK-Bericht 4714, 1991.

[Can86] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), November 1986.

[Car89] Gail A. Carpenter. Neural Network Models for Pattern Recognition and associative Memory. *Neural Networks*, 2:243–257, 1989.

[CD86] Roland T. Chin und Charles R. Dyer. Model-Based Recognition in Robot Vision. *acm computing surveys*, 18, März 1986.

[CY91] Peter Coad und Edward Yourdon. *Object-Oriented Analysis.* Prentice-Hall, 2. Auflage, 1991.

[DG91] Rachid Deriche und Gerard Giraudon. On Corner and Vertex Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[DIN90] DIN. DIN ISO 9000, 5 1990.

[ED93] John R. Ellis und David L. Detlefs. Safe, Efficient Garbage Collection for C++. *Xerox Corporation*, 1993.

[EOS92a] E. Erwin, K. Obermayer und K. Schulten. Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, 67:47–55, 1992.

[EOS92b] E. Erwin, K. Obermayer und K. Schulten. Self-organizing maps: stationary states, metastability and convergence rate. *Biological Cybernetics*, 67:35–45, 1992.

[E.R88] E.R.Davies. *Machine Vision: Theory, Algorithms, Practicalities.* Academic Press, 1988.

[FL90] P. Fua und Y. G. Leclerc. Model Driven Edge Detection. *Machine Vision and Applications*, Seiten 45–56, 1990.

[GDG+93] H. Guth, C. Düpmeier, M. Goik, A. Hellmann und U. Stucky. Automatische Vermessung von 2D- und 3D-LIGA-Strukturen. In *1. Statuskolloquium des Projektes Mikrosystemtechnik*, Seiten 176–182. KfK Bericht 5238, September 1993.

[GWCK92] Lalit Gupta, Jiesheng Wang, Alain Charles und Paul Kisatsky. Prototype Selection for Neural Network Training. *Pattern Recognition*, 25(11):1401–1408, 1992.

[Haw70] J. K. Hawkins. Textural Properties for Pattern Recognition. *Picture Processing and Psychopictorics*, Seiten 347–370, 1970.

[HC92] Gongzu Hu und Xinping Cao. A Subpixel Edge Detector Using Expectation of First-Order Derivatives. *SPIE*, 1657, 1992.

[Hei94] Sigfried Heintze. Designing efficient Container Classes. *Journal of Object Oriented Programming*, 7(6):22–27, 1994.

[HKP91] J. Hertz, A. Krogh und R. G. Palmer. *Introduction To The Theory Of Neural Computation*, Jgg. 1 of *Computational and neural systems series*. Addison-Wesley, 1991.

[HP92] K. Hirota und W. Pedrycz. Prototype Construction and evaluation as Inverse Problems in Pattern Classification. *Pattern Recognition*, 25(6):601–608, 1992.

[HS92] R. M. Haralick und L. G. Shapiro. *Computer and Robot Vision*, Jgg. 1. Addison-Wesley, 1992.

[HS93] C. M. Haffer und R. Seppelt. Neuronale Netzwerke als Werkzeuge zur Qualitätssicherung in der Halbleiterherstellung. *KI*, März 1993.

[Jos94] Nicolai Josuttis. *Objektorientiertes Programmieren in C++*. Addison-Wesley, 1994.

[KHHK94] P. Kohlhepp, H. Haffner, J. Hansemann und J. Koprek. Einsatz von Entfernungssensoren zur Vermessung und Erkennung dreidimensionaler Objekte. KfK-Nachrichten, 26(4/94) S.251-262, 1994.

[KK93] B. R. Kämmerer und W. A. Küpper. Perceptron and Neocognitron: Experimental Studies of Artificial Neural Nets for Pattern Recognition Applications. *International Journal of Computer Vision*, 1993.

[KR82] Les Kitchen und Azriel Rosenfeld. Gray-level corner detection. *Pattern Recognition Letters*, Seiten 95–102, 1982.

[Krö87] B. J. A. Kröse. Local Structure Analyzers as Discriminants of Preattentive Pattern Discrimination. *Biological Cybernetics*, 55:289–298, 1987.

[KWT88] M. Kass, A. Witkin und D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1988.

[Lüb] Ulrich Lübbert. *MiniVISTA*. Fraunhofer-Institut IITB, Fraunhoferstr.1 76135 Karlsruhe.

[Lip90] Stanley B. Lippman. *The C++ Primer*. Addison Wesley, 1990.

[Mas88] R. Massen. Real-Time grey-level and colour image preprocessing for a vision guided bitechnology robot. In *Proc. Int. Conf Robot Vision and Sensory Controls*, Seiten 115–122. IFS (Conferences) LTD and authors, 1988.

[MB93] W. Menz und P. Bley. *Mikrosystemtechnik für Ingenieure.* VCH, Weinheim, New York, Basel, Cambridge, 1993.

[MD92] Anwar K. Muhamad und Farzin Deravi. Cooccurrence based features for automatic texture classification using neural networks. *SPIE*, 1766, 1992.

[MD93] Anwar K. Muhamad und Farzin Deravi. Neural Network Texture Classifiers using direct Input Coocurrence Matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.

[Mey88] Bertrand Meyer. *Object-oriented Software Construction.* Series in Computer Science. Prentice Hall, 1988.

[MH80] D. Marr und E. Hildreth. Theory of edge detection. *Proc. R. Soc. Lond.*, B 207:187 - 217, 1980.

[ML91] Miguel A. Mayorga und Lonnie C. Ludeman. Neural Nets for Determination of Texture and its Orientation. In *IEEE international conference on acoustics, speech, and signal processing*, Seiten 2689–2692, 1991.

[MR90] B. Müller und J. Reinhardt. *Neural Networks, An Introduction.* Physics of Neural Networks. Springer, 1990.

[NB86] Vishvjit S. Nalwa und Thomas O. Binford. On detecting Edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), November 1986.

[Oe93] Shunichiro Oe. Texture Segmentation Method by using Two-Dimensional Model and Kullback Information. *Pattern Recognition*, 26(2):237–244, 1993.

[PBC92] D. T. Pham und E. J. Bayro-Corrochano. Neural Computing for Noise Filtering, Edge Detection and Signature Extraction. *Journal of Systems Engineering*, 2:111–222, 1992.

[Pic70] R. M. Pickett. Visual Analysis of Texture in the Detection and Tecognition of Objects. *Picture Processing and Psychopictorics*, Seiten 289–308, 1970.

[PJJJ84] K. Paler, J.Föglein, J.Illingworth und J.Kittler. Local ordered grey levels as an aid to corner detection. *Pattern Recognition*, 17(5):535 –543, 1984.

[Pra78] William K. Pratt. *Digital Image Processing.* John Wiley, 1978.

[PS91] D. Patel und T. J. Stonham. A Single Layer Neural Network for Texture Discrimination. In *1991 IEEE International Symposium on Circuits and Systems*, Jgg. 5, 1991.

[RK82] A. Rosenfeld und A. C. Kak. *Digital Picture Processing*, Jgg. 1 of *Computer Science and Applied Mathematics*. Academic Press Inc, 2. Auflage, 1982.

[RM88] D. E. Rumelhart und L. McClelland. *Parallel Distributed Processing*, Jgg. 1. MIT Press, 7. Auflage, 1988.

[RM90] Arturo A. Rodriguez und Jon R. Mandeville. Image registration for automated inspection of 2D electronic circuit patterns. *SPIE*, 1384, 1990.

[Roh92] Karl Rohr. Modelling and identification of characteristic intensity variations. *Image and Vision Computing*, 10(2), 1992.

[RS89] Joseph Rosen und Joseph Shamir. Scale invariant pattern recognition with logarithmic radial harmonic filters. *Applied Optics*, 28(2), Januar 1989.

[RS90] J. Rubner und K. Schulten. Development of Feature Detectors by Self-Organization. *Biological Cybernetics*, 62:193–199, 1990.

[RS92] Karl Rohr und Christoph Schnörr. An efficient approach to the identification of characteristic intensity variations. *Image and Vision Computing*, 11(5):273–277, 6 1992.

[SB92] D. Daniel Sheu und Alan H. Bond. A generalized Method for 3D Object Location from single "D Images. *Pattern Recognition*, 25(8):771–786, 1992.

[SBC93] H. C. Shen, C. Y. Bie und D. K. Y. Chiu. A Texture-based Distance Measure for Classification. *Pattern Recognition*, 26(9):1429–1437, 1993.

[SFH92] P. Suetens, P. Fua und A. J. Hanson. Computational Strategies for Object Recognition. *acm computing surveys*, 24(1), März 1992.

[SFK+93] M. Strohrmann, O. Fromhein, W. Keller, K. Lindemann und J. Mohr. LIGA-Sensoren und intelligente Sensorsysteme zur Messung von Beschleunigungen. In *1. Statuskolloquium des Projektes Mikrosystemtechnik*, Seiten 65–70. KfK Bericht 5238, September 1993.

[SRSW84] Hanan Samet, Azriel Rosenfeld, Clifford A. Shaffer und Robert E. Webber. A Geographic Information System using Quadtrees. *Pattern Recognition*, 17(6):647–656, 1984.

[ST92] Yung-Nien Sun und Ching-Tsorng Tsai. A new Model Based Approach for Industrial Visual Inspection. *Pattern Recognition*, 25(11):1237–1336, 1992.

[Sto83] Josef Stoer. *Einführung in die Numerische Mathematik I*, Seiten 183–202. Springer-Verlag, 4. Auflage, 1983.

[Stu94] U. Stucky. Rechnergestützte Vermessung von dreidimensionalen Mikrostrukturen. Dissertation, Kernforschungszentrum Karlsruhe, 5 1994.

[TC77] J. T. Tou und Y. S. Chang. Picture Understanding by Machine via Textural Feature Extraction. In *Conference on Pattern Recognition and Image Processing*, Seiten 392–399, 1977.

[TH86] Qi Tian und Michael N. Huhns. Algorithms for Subpixel Registration. *Computer Vision, Graphics, and Image Processing*, 35, 1986.

[TP89] D. S. Touretzky und D. A. Pomerleau. What's Hidden in the Hidden Layers. *Byte*, August 1989.

[TSK90] Andreas Tirakis, Levon Sukissan und Stefanos Kollias. An Adaptive Technique for Segmentation and Classification of Textured Images. In *International Neural Network Conference*, Jgg. 1, 1990.

[Ull89] Shimon Ullman. Aligning pictorial descriptions: An approach to object recognition. *Cognition*, 32:193–254, 1989.

[Vis91] Ari Visa. Texture Classification and Neural Network Methods. *SPIE Applications of Artificial Neural Networks II*, 1469, 1991.

[WC90] G. A. W. West und T. A. Clarke. A survey and examination of subpixel measurement techniques. *SPIE*, 1395, 1990.

[Wei89] Isaac Weiss. Line Fitting in a Noisy Image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(3):325–329, 1989.

[WR83] Z. Wu und A. Rosenfeld. Filtered projections as an aid in corner detection. *Pattern Recognition*, 16(1):31–38, 1983.

[YY92] Stephan R. Yhann und Tzay Y. Young. Parallel distributed algorithm for texture boundary localization. *SPIE*, 1766, 1992.

[ZH83] Oscar A. Zuniga und Robert M. Haralick. Corner Detection using the Facet Model. *IEEE Transactions on Computer Vision*, 1983.

[Zha94] Jinyou Zhang. *Bereichsbasierte Verfahren zur Straßenerkennung für die autonome Führung von Fahrzeugen.* VDI-Verlag, 1994.

Danksagung

An dieser Stelle möchte ich all jenen danken, die Anteil hatten an der Entwicklung und Fertigstellung dieser Arbeit.

Für die Übernahme des Referats und die strukturellen Hinweise zum Aufbau der Arbeit möchte ich Herrn Prof. Dr. Wolfgang Menz vom Institut für Mikrostrukturtechnik (IMT) des Forschungszentrums Karlsruhe danken. Die Arbeit selbst entstand am Institut für Angewandte Informatik (IAI). Einen Dank dem Institutsleiter Herrn Prof. Dr. Heinz Trauboth für die Übernahme des Korreferates und die gezielten Hinweise zur Präsentation der Arbeit und der darin enthaltenen Ergebnisse. Beiden Doktorvätern zudem ein Lob für das faire Verhalten mir gegenüber während der Betreuung.

Meinem fachlichen Betreuer Helmut Guth möchte ich danken für die Anregungen, die er mir für den Fachbereich Bildverarbeitung gab. Zudem hat mir Helmut bei verschiedenen Gelegenheiten den bürokratischen Tiefgang einer „Großforschungseinrichtung" abgenommen oder zumindest den Umgang damit erleichtert.

Einen Dank an Horst Eggert, der mich immer antrieb, „voran" zu machen nach der Devise, es ist immer später als du denkst. Sein Sinn für passende Zeitpunkte mahnte mich davor, bei Vorträgen und Berichten ins zeitliche Hintertreffen zu geraten. Danke auch für die Vertretung meiner Arbeit gegenüber der Institutsleitung.

Einen besonderen Dank an Klaus Lindemann, dessen subtiler Tiefgang mir erst nach und nach klar wurde. Seine konstruktiven Beiträge und seine tiefen Softwarekenntnisse haben mir oft viel Ärger erspart. Zudem habe ich es in der gesamten Zeit *nie* erlebt, daß Klaus einmal keine Zeit für die Beantwortung einer Frage oder für eine Diskussion hatte. Lieber Klaus, Du warst einer der kritischsten Leser dieser Arbeit, vielen Dank auch dafür!

Ich möchte es als persönlichen Glücksfall bezeichnen, daß ich Zimmer und Schreibtisch mit Peter Wieland teilen durfte, mit dem mich nach einer Eingewöhnungszeit wesentlich mehr verband als die Tagesarbeit. Lieber Peter, auch wenn ich lange nicht alle Anregungen von Deiner Seite angenommen habe, so sind sie ganz

sicher Teil dessen, was ich einfach nur als persönlichen Gewinn bezeichnen möchte, den ich durch Dich erhielt.

Sabine Scheer hat mein angeschlagenes Vertrauen in Sekretärinnen um mindestens 1000 Punkte angehoben. Es gibt sie also noch. Nicht dünkelhaft eingebildet, schnell, hilfsbereit und geduldig trotz jahrelanger Routine. Sabine, Du warst der große Pluspunkt, was Verwaltungskrempel angeht, ich finde es toll, daß Du auch im Streß immer noch Zeit hattest für die vielen Kleinigkeiten eines Doktorranden.

Eine große Hilfe war Clemens Düpmeier. Zwischenzeitlich selbst zum Doktor avanciert, war er auch in meinem Fall der unverzichtbare Mann im Hintergrund. Ohne seine qualifizierte Betreuung des IAI Rechnernetzes, sowie die zahllosen kompetenten Hinweise zu System- und Softwarekomponenten, hätte ich definitiv wesentlich länger für die Arbeit gebraucht. Clemens war es auch, der mich auf die Idee der Betrachtung textureller Merkmale brachte.

Peter Stiller überzeugte mich durch seinen Einblick in die Hierarchien und Entscheidungswege des Forschungszentrums. Er dämpfte zu passenden Zeitpunkten meine Wut auf unabänderliche Fakten und brachte mich wieder auf den berühmten Teppich zurück. Schade eigentlich, Peter, daß Du immer nur im Sommer unsere „Salatbar" mit pfälzischem Gemüse aus Hatzenbühl bereichert hast.

Und dann ist da Uwe Stucky, mit dem ich viele an- und aufregende Diskussionen über Ziel und Zweck eines zu konstruierenden Vermessungssystems hatte. Auch Du, Uwe, hattest immer Zeit und Rat im Fall einer zweifelhaften Entscheidung zum weiteren Vorgehen.

Leider muß wohl jede Liste dieser Art ein Ende finden, obwohl sie eigentlich noch gar nicht wirklich zu Ende ist. Und da sind jene vielen, die hier nicht namentlich auftauchen und dennoch Ihren Beitrag geleistet haben. Ihnen allen möchte ich pauschal einen Dank aussprechen in der Hoffnung, Sie mögen mir die fehlende individuelle Widmung verzeihen.