KfK 4799
Dezember 1990

Heuristisches Mustererkennungsverfahren zur Verifikation zwei-dimensionaler Bildobjekte

H. Andriessen, B. Bürg, H. Guth, A. Hellmann
Institut für Datenverarbeitung in der Technik

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik

KfK 4799

Heuristisches Mustererkennungsverfahren zur Verifikation zwei-dimensionaler Bildobjekte

Henk Andriessen *)

B. Bürg, H. Guth, A. Hellmann

*) Universität Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE GMBH, KARLSRUHE

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Heuristic Pattern Recognition Method for the Verification of Two Dimensional Image Objects

ABSTRACT

This paper deals with the problem of coordination between "theoretical" and "observed" structural marks occurring at image analysing measurements of microstructures with any surface geometry. Several operations-research methods were examined on their applicability. Out of a variety of possible methods four promising ones were chosen and described in detail. One of the results is that all four problems can be reduced to the flow problem.

By further investigation it was shown that none of the four methods meets all the requirements of the image processing system (COSMOS-2D) with regards to the time and memory specification. Therefore a new technique had to be developed to fulfil the requirements.

The basic idea behind the developed procedure is an iterative minimization of the differences in position between a "theoretical" and the corresponding "observed" structural mark, using only "reasonable" related pairs for the next iteration step. To reckon this differences a transformation matrix is calculated on the base of the least square method. The coordination procedure puts the locally obtained pair of marks in a more global geometric interrelationship.

This procedure was implemented on the target machine (Transputer T800, INMOS) and tested with different objects under extreme conditions. The results show that the number of the failures in the related pairs were reduced to a minimum within a few iteration steps.

The procedure meets the needs of the requirements of COSMOS-2D with regards to the time and memory specifications.

Heuristisches Mustererkennungsverfahren zur Verifikation zwei-dimensionaler Bildobjekte

ZUSAMMENFASSUNG

In dieser Arbeit wird das Problem der Zuordnung von Ist- zu Soll-Strukturmerkmalen bei der bildanalytischen Vermessung von Mikrostrukturen mit beliebiger Oberflächengeometrie behandelt. Im Rahmen der Suche nach einem geeigneten Zuordnungsverfahren werden mehrere Operations-Research (OR)-Modelle auf ihre Anwendbarkeit geprüft. Aus der Vielfalt der Modelle werden exemplarisch vier vielversprechende ausgewählt und ausführlich besprochen. Es hat sich gezeigt, daß sich alle vier Modelle auf die eine oder andere Weise auf das sogenannte Flußproblem zurückführen lassen.

Eine genaue Analyse dieser Modelle zeigt, daß keines die Anforderungen des übergeordneten Bildanalysesystems (COSMOS-2D) in Bezug auf Zeit- und Speicherplatzbedarf erfüllen kann, weshalb die Entwicklung eines neuen Algorithmus notwendig wird, der diesen Anforderungen gerecht wird.

Das entwickelte Verfahren basiert auf einer iterativ minimierten Positionsdifferenz zwischen Soll- und Ist-Strukturmerkmalen, wobei jeweils nur "plausible" Zuordnungen in den weiteren Iterationsschritten berücksichtigt werden. Zur Berechnung der Positionsdifferenz wird eine Transformationsmatrix mit Least-Square-Methoden ermittelt. Der Algorithmus stellt somit die vom übergeordneten Bildanalysesystem lokal ermittelten Merkmalszuordnungen in einen globalen geometrischen Zusammenhang.

Das Verfahren, das auf der Zielmaschine (Transputer T800 von INMOS) im Rahmen dieser Arbeit implementiert worden ist, wurde mit unterschiedlichstem Bildmaterial sowie unter schwierigen Startbedingungen getestet. Die Resultate zeigen, daß die fehlerhaften Zuordnungen innerhalb weniger Iterationsschritte gegen einen minimalen Wert konvergieren.

Dieses Verfahren erfüllt die Anforderungen von COSMOS-2D, insbesondere in Bezug auf die Zeit- und Speicherplatzvorgaben.

INHALTSVERZEICHNIS

1 Einleitung

Kernforschungszentrum Karlsruhe (KfK)

Im Jahre 1956 wurde das Kernforschungszentrum Karlsruhe (KfK) mit dem Ziel gegründet, die Nutzung der Kernenergie vor allem unter dem Aspekt der Sicherheit im Umgang mit kerntechnischen Anlagen zu erforschen. Mit der Drosselung der Forschungsaktivitäten auf diesem Gebiet wurde das Kernforschungsprogramm in den 80er Jahren nach und nach bis heute auf etwa ein Drittel zurückgefahren. Es entstanden andere Arbeitsschwerpunkte wie die Umweltforschung, die Handhabungstechnik oder auch die Mikrotechnik.

Mikrotechnik

Gegenstand der Forschung und Entwicklung im Rahmen der Mikrotechnik war immer die Miniaturisierung elektronischer Bauelemente. Während der erste Computer noch mehrere Räume deckenhoch füllte, benutzt man heute, nur wenige Jahrzehnte später, im Arbeitsalltag kleine Tischrechner, die um Dimensionen mehr Leistung erbringen als jener erste Rechner.

Die Entwicklung der Mikrotechnik ist dabei keineswegs am Ende angelangt. Forschungsgegenstand ist einerseits eine weitere Leistungssteigerung der Bauelemente, andererseits die Herstellung von mechanischen Mikrostrukturen, d.h. stabilen Strukturen im mikroskopischen Bereich. Beispiele hierfür sind Filter in Wabenform (Bild 1.1) mit genau definierten Durchlässen im Mikrometerbereich, optische Multiplexer für den Einsatz in der Glasfasertechnik oder Mikrostecker. Grundsätzlich ist jede 2-dimensionale geometrische Oberflächenform denkbar.

Eine interessante Zukunftstechnologie wird die Kombination von elektronischen und mechanischen Bauelementen sein, die Mikrosystemtechnik. Mit ihr lassen sich kleinste Systeme wie Mikromotoren oder Beschleunigungssensoren im Mikrometerbereich bauen.

Für die Fertigung von mechanischen Mikrostrukturen wurden im KfK verschiedene Verfahren entwickelt, wovon das herausragendste das sogenannte **LIGA**-Verfahren (**Li** tographie, **G** alvanik, **A** bformung) [Beck85] ist.

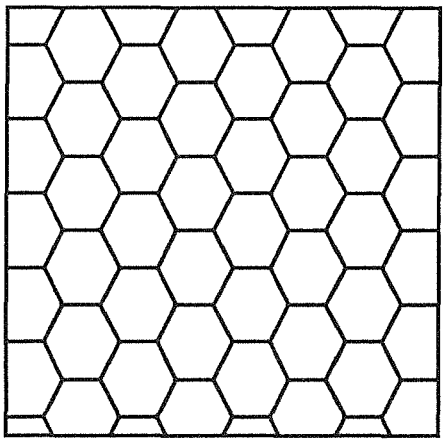

Bild 1.1 Filter (Wabenstruktur)

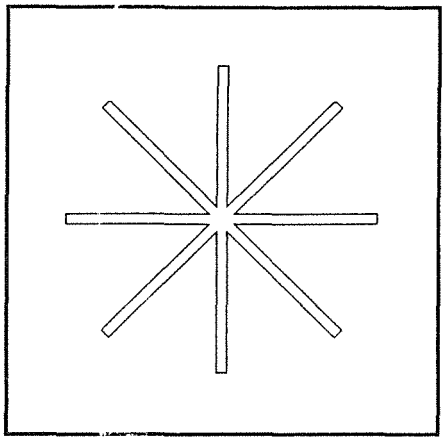

Bild 1.2 Düse

LIGA-Verfahren

Das LIGA-Verfahren ist eine Prozeßtechnik zur Herstellung von mechanischen Mikrostrukturen, die sich auszeichnen durch ihr außerordentlich hohes Aspektverhältnis, das Verhältnis von Strukturwandhöhe zu Strukturwandbreite im mikroskopischen Bereich.

Mit dem LIGA-Verfahren können stabile Strukturen im Mikrometerbereich mit einem Aspektverhältnis bis zu 100 und Abweichungen in der Strukturwanddicke im Submikrometerbereich hergestellt werden. Das Verfahren erlaubt die Fertigung von Mikrostrukturen mit beliebigen Oberflächengeometrien. Einige wenige Beispiele aus einer Vielzahl von möglichen Anwendungen sind hochpräzise Filter oder Düsen (Bild 1.1 und 1.2).

Die zu fertigende Mikrostruktur wird in einem CAD-System entworfen. Der Designer spezifiziert hier die Konstruktion des Produktes und legt gleichzeitig die spätere Qualitätskontrolle fest.

Anhand der CAD-Daten fertigt ein Elektronenstrahlschreiber eine primäre Maske an. Über die aus der primären Maske gefertigten Röntgenmaske wird mit Synchrotronstrahlung ein Resist bestrahlt, dessen nicht resistente Teile weggeätzt werden.

Durch verschiedene Galvanik- und Abformungsschritte entsteht ein "Werkzeug", mit dem durch weitere Abformung die Endprodukte in Serie hergestellt werden.

Der Fertigungsprozeß macht deutlich, daß eine Qualitätskontrolle notwendig ist, und zwar nach jedem Zwischenprodukt in Form von Ganzheitskontrollen und für jedes Endprodukt in Form von parametrischen Tests.

Optisches Qualitätskontroll- und Meßsystem

Für die zerstörungsfreie Oberflächenprüfung der LIGA-Strukturen wurde am Institut für Datenverarbeitung in der Technik (IDT) des KfK der Prototyp eines auf der Basis der digitalen Bildanalyse arbeitendes Qualitätskontroll- und Meßsystem entwickelt.

Dieses System soll bei der Serienherstellung eingesetzt werden und läuft deshalb vollkommen automatisch ab. Diese Eigenschaft aber verbietet ein Einlernen von Referenzstrukturen. Die notwendigen Informationen über die zu vermessenden Bereiche einer Struktur werden vom Entwicklungsingenieur im CAD-System beim Design der Struktur bereits eingetragen. Kritische Bereiche wie Linien, Distanzen, Flächen usw. sind somit im CAD-System auf einer zusätzlichen Informationsebene (Layer) vorhanden.

In einem Datenaufbereitungsprogramm werden die Daten der kritischen Bereiche mit den Original-Strukturdaten verknüpft und in Form von Mustererkennungs- und Auswertungsbefehlen für das Bild-Verarbeitungs (BV) -System formuliert.

Die Informations- und Datenbasis zwischen CAD- und BV-System basiert auf einem abgestuften System von Mustererkennungsbefehlen. Zur untersten Stufe gehört das Erkennen von Ecken bestimmter Ausprägung, der Strukturecken. Die nächste Stufe bilden Befehle zur Verifikation von Linien zwischen erkannten Ecken. Dann folgt die höchste Stufe, die das Messen von Distanzen zwischen Ecken und Linien oder auch Flächenprüfungen ermöglicht.

Bild 1.3 zeigt die Strukturecken, wie sie das BV-System erkennen kann: Eine Ecke als Kombination zweier gerader Linien (X-Typ) oder einer geraden Linie und einem Kreisbogen (Y-Typ) oder zweier Kreisbögen (Z-Typ).

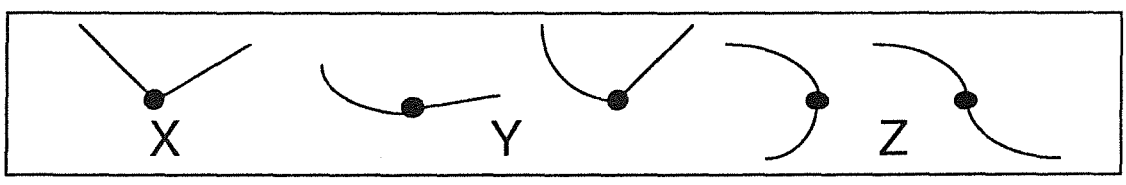

Bild 1.3 Verschiedene Typen von Strukturecken (X, Y und Z- Ecken)

Alle BV-Befehle werden abschnittsweise für ein Bild, dessen Auflösung beim Eintrag der kritischen Bereiche gewählt wurde, in eine Datei, die Soll-Datei, geschrieben. Die Qualitätskontrolle erfolgt dadurch, daß das Meßsystem sukzessiv die Soll-Datei abarbeitet.

Mit der symbolischen Beschreibung der geometrischen Form einer Struktur durch Primitiven, wie Ecken und Linien, kann das BV-System nahezu jede beliebige Struktur anhand ihrer Form positionsinvariant verifizieren und vermessen.

Bild 1.4 zeigt die Einbindung der Qualitätskontrolle im Gesamtablauf, und Bild 1.5 veranschaulicht den Ablauf der Bildanalyse [Bürg87a,Bürg89b]. In Bild 1.6 wird die Rechnerarchitektur des BV-Systems skizziert, wobei auf den parallelen Ansatz hingewiesen sei, der es erlaubt, entsprechend der Anzahl der Transputer, gleichzeitig mehrere Bilder auszuwerten [Bürg89b].

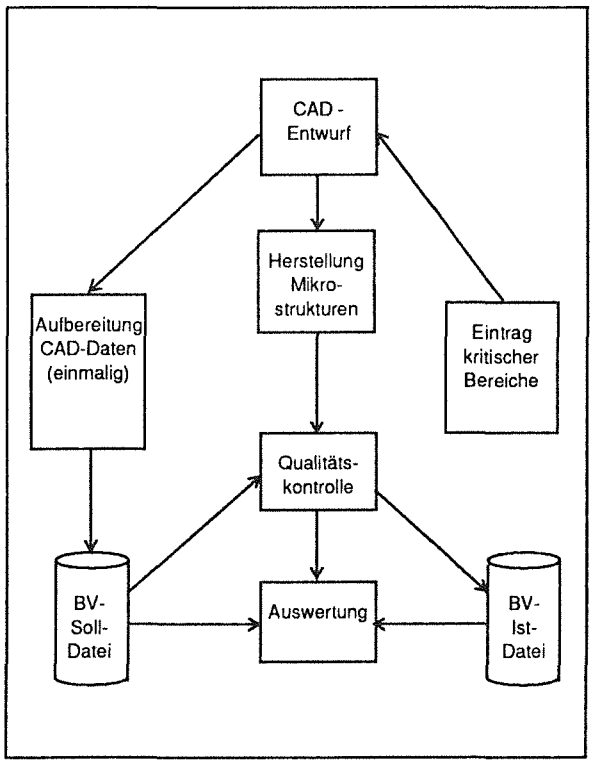

Bild 1.4 Konzept der Qualitätskontrolle aus [Bürg89b]

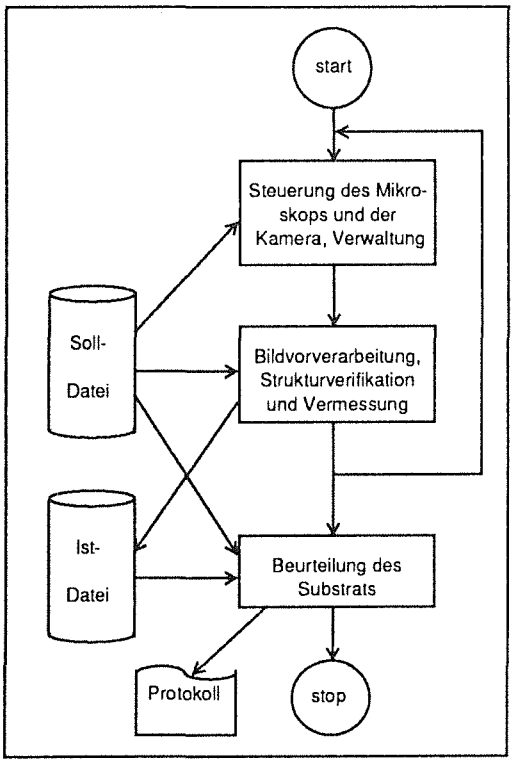

Bild 1.5 Ablauf der Bildanalyse aus [Bürg87a]

Beim Ablauf der Bildanalyse muß unterschieden werden zwischen Funktionen, die der Masterrechner ausführt, und Funktionen, die die Parallel-Prozessoren, also die Transputer, übernehmen.

Die Funktionen des Masterrechners sind:

- Lesen des Datenabschitts für einen Struktur- / Bildbereich aus der Soll-Datei.
- Positionieren des Mikroskoptisches.
- Akquisition eines Grauwertbildes.
- Weitergabe eines Auftrags (Bild + Befehle) an einen BV-Prozessor (Transputer).
- Verwaltung und Auswertung der Ergebnisse der BV-Prozessoren.

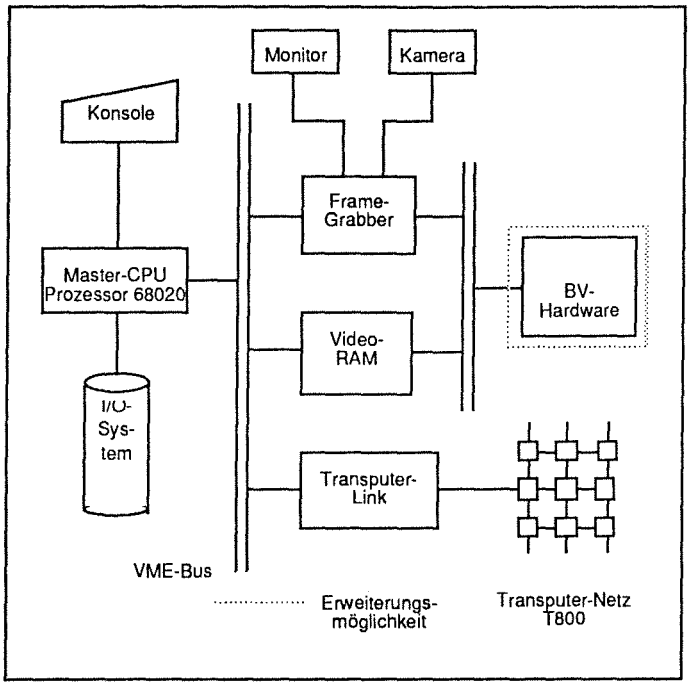

Bild 1.6 Rechnerarchitektur des BV-Systems aus [Bürg89b].

Jeder einzelne BV-Prozessor führt die Analyse eines Bildes folgendermaßen durch:

- Berechnung eines Kantenbildes (Bildvorverarbeitung).
- Verifikation aller durch die Soll-Daten vorgeschriebenen Ecken im Bild.
- Detektion der Linien zwischen erkannten Strukturecken.
- Messen von Standardabweichungen bei Kreisbögen bzgl. des berechneten Radius, bei geraden Linien bzgl. einer an alle Linienpunkte gefitteten Geraden.
- Messen von Distanzen zwischen Ecken und Strukturkanten.

Die Art der Strukturecken-Detektion läßt in der Regel eine eindeutige Verifikation des Bildinhaltes zu. Bei Strukturen allerdings, die mehrere Ecken gleicher Ausprägung, also mit gleichen Schenkelrichtungen, auf engstem Raum aufweisen (Bild 1.7), sind fehlerhafte Zuordnungen von erkannten Ist-Ecken zu Soll-Ecken möglich. Für solche Fälle soll ein Algorithmus entwickelt werden, der eine sichere und schnelle Zuordnung gewährleistet.

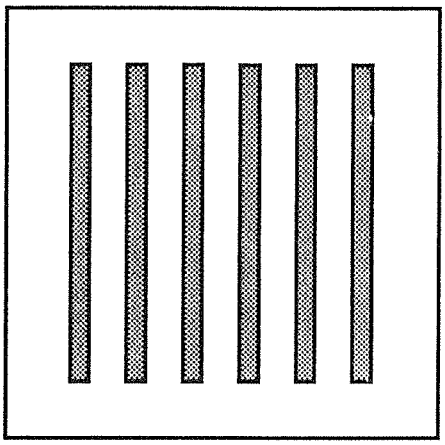

Bild 1.7 Balkenfigur

Problemstellung

Die zeitliche Restriktion bei der Bildanalyse verhindert eine Mustererkennung, die die globale Strukturgeometrie miteinbezieht. Die Strukturverifikation stützt sich vielmehr punktuell auf Strukturecken, die wohl in Form von Schenkelwinkeln gegeben sind, nicht jedoch auf die topologische Abhängigkeit der Ecken untereinander. Die Position der Ecke geht nur insofern in die Suche ein, als sie als Ausgangspunkt für ein quadratisches Suchfenster (Window) dient.

Die Größe des Windows ist ein "Ablaufparameter" und kann vom Benutzer für jedes (automatisch) einstellbare Objektiv am Mikroskop definiert werden. Sie hängt unmittelbar von der Auflösung ab, die der Entwurfsingenieur beim Eintrag der kritischen Bereiche in das CAD-System für jeden Bildausschnitt vorgeben kann. Bei hoher Auflösung wird beispielsweise ein großes Objektiv und damit eine starke Vergrößerung gewählt. Der Bildausschnitt wird kleiner und eine Positionsverschiebung macht sich stärker bemerkbar. Bei einer fest vorgegebenen Tischungenauigkeit oder einer von den Prozeßfachleuten erklärten möglichen Positionsvarianz der Strukturen muß das Suchfenster entsprechend angepaßt, in diesem Fall also vergrößert werden.

Bild 1.8 zeigt eine Balkenstruktur. Die Soll-Strukturen, wie sie die Soll-Datei liefert, sind gestrichelt eingezeichnet.
Die Ist-Position der Balken sei mit durchgezogenen Linien angedeutet. Vom mittleren Balken soll die rechte obere Ecke gesucht werden. Ausgehend von dem Soll-Punkt dieser Ecke werden in einem quadratischen Window (im Bild schraffiert) alle Ist-Linien auf Paßgenauigkeit mit der gesuchten Ecke durchsucht.

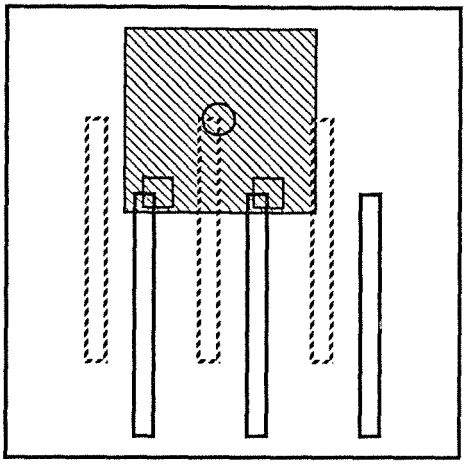

Bild 1.8 Balkenfigur (Kantenbild)

Zu diesen Linien gehören allerdings **zwei** Ist-Ecken, die die gleichen Parameter der gesuchten Ecke aufweisen (kleine Quadrate). Die Wahl des Suchalgorithmus fällt auf die Ecke, die eine bessere Eck-Geometrie aufweist, d.h. den jeweiligen Eck-Parametern besser entspricht.

Damit besteht die Möglichkeit, daß die falsche Ecke gewählt wird. Somit ist das Ergebnis der Mustererkennung bei solchen Strukturen nicht zwingend eindeutig. Nur in einer globalen geometrischen Betrachtung kann eine falsche Zuordnung von Ist-Ecken zu Soll-Ecken korrigiert werden.

Im Rahmen der vorliegenden Arbeit soll nun ein Algorithmus entwickelt werden, der eine sichere und schnelle Zuordnung unter den genannten Rahmenbedingungen gewährleistet. Dieser Algorithmus soll implementiert und in das beschriebene Qualitätskontroll- und Meßsystem integriert werden.

Vorgehensweise

Nachdem die Aufgaben- und Problemstellung diskutiert worden ist, stellt sich die Frage nach der optimalen Zielerfüllung.

Im Vordergrund steht zunächst die Ermittlung der systembedingten Anforderungen, denn das Lösungsverfahren soll implementiert und in das vorhandene Bildverarbeitungssystem integriert werden. Die Vorgaben durch das System sind damit eindeutig festgelegt und müssen bei jedem weiteren Vorgehen berücksichtigt werden.

Die in dem Abschnitt "Problemstellung" angedeuteten möglichen Fehler müssen eingegrenzt und klassifiziert werden, so daß sowohl eine qualitative als auch eine quantitative Beschreibung möglich ist. Erst dann lassen sich die Anforderungen an das Lösungsverfahren exakt formulieren.

Bei der Betrachtung der Aufgabenstellung liegt die Vermutung nahe, daß es sich um ein Optimierungsproblem handelt. Ein Wissenschaftszweig, der sich überwiegend mit Optimierungsproblemen beschäftigt, und auch exakte oder heuristische Algorithmen zur Verfügung stellt, ist das Operations Research (OR). Möglicherweise existiert ein verwandtes OR-Problem und ein entsprechendes Lösungsverfahren, das im vorliegenden Fall Verwendung finden könnte. In einer Literaturrecherche soll dies ermittelt werden. Mehrere Verfahren sollen auf ihre Tauglichkeit untersucht und unter dem Aspekt einer möglichen Anwendung im vorliegenden Fall ausführlich diskutiert werden.

Dabei wird die Notwendigkeit ersichtlich, ein neues Verfahren zu entwickeln. Der Algorithmus soll zunächst programmunabhängig formuliert werden, wobei die Datenschnittstelle zum System detailliert beschrieben werden soll.

Dieser Algorithmus soll schließlich in der Programmiersprache "C" implementiert und in das System eingebunden werden. Detaillierte Tests und die Beschreibung und Diskussion ihrer Ergebnisse sollen die Tauglichkeit des Verfahrens unter Beweis stellen.

Schließlich soll das Verfahren einer allgemeinen Bewertung unterzogen werden und in einem Ausblick zukünftige Entwicklungen auf diesem Gebiet angesprochen werden.

Gliederung

Kapitel 2 gibt eine Beschreibung der Anforderungen an das Lösungsverfahren.

Die Vorstellung des Problems erfolgt dann in Kapitel 3. Hier werden mögliche Schwierigkeiten aufgezeigt und die Untersuchung von Operations-Research-Verfahren in bezug auf das gegebene Problem dargestellt.

Kapitel 4 beschreibt ein eigenes Lösungsverfahren.

Die Beschreibung der Implementierung des Verfahrens auf dem Zielrechner (Transputer) wird in Kapitel 5 aufgezeigt. Nach der Implementierung erfolgt die Erläuterung der Tests und eine Bewertung des Verfahrens.

Den Schluß bildet in Kapitel 6 ein Ausblick auf weitere Einsatzgebiete.

2 Randbedingungen des Systems

Auf die Entwicklung des Zuordnungsverfahrens folgt sein Einbau in das Qualitätskontroll- und Meßsystem. Es muß deshalb untersucht werden, welche Anforderungen die Programmumgebung in bezug auf Ablaufgeschwindigkeit, Speicherplatz, Programmiersprache und nicht zuletzt auf Software-Qualitätsrichtlinien stellt.

Ablaufgeschwindigkeit

Das LIGA-Verfahren wurde letztlich für die Serienherstellung von Mikrostrukturen in einer Produktionsstraße entwickelt. Die Qualitätskontrolle bezüglich der Endprodukte wird damit einem Fertigungstakt von einigen Minuten (nach heutigem Wissensstand) unterworfen. Ein Fertigungsnutzen kann dabei die Kontrolle mehrerer tausend Bilder unterschiedlichsten Inhalts erfordern, wenn man von einer Ganzheitsüberprüfung ausginge. Eine detaillierte Messung, wie sie das Qualitätskontrollsystem durchführen kann, ist für diese Anzahl Bilder in so kurzer Zeit nicht möglich. Es ist deshalb beabsichtigt, sogenannte parametrische optische Tests einzuführen, bei denen Hilfsstrukturen, die auf dem Substrat an repräsentativen Positionen zusätzlich aufgebracht sind, ausgemessen werden, und von den ermittelten Ergebnissen auf die Qualität der übrigen Strukturen geschlossen wird. Der Aufwand läßt sich damit auf die Vermessung einiger hundert Bilder reduzieren.

Mit einem Durchsatz des Meßsystems von einem Bild pro Sekunde könnte dann der Produktionstakt eingehalten werden.

Alle Programme für die bildanalytische Vorverarbeitung eines Bildes, die Eckenbestimmung und die Vermessung laufen auf allen Transputern vollständig ab. Ein Wabenbild (Bild 1.1) - die Vermessung eines Wabenbildes repräsentiert den Worst-Case Fall - erfordert die Detektion von mehr als 200 Ecken, die Verifikation und Vermessung von ebensoviel Geradenstücken, wobei Unterbrechungen oder Unregelmäßigkeiten festgestellt, in die Position der Linienpunkte eine Gerade mathematisch gefittet und die Standardabweichung aller Linienpunkte zu diesen Geraden und deren maximale Abweichung berechnet werden. Zudem werden die Abstände der Ecken zueinander in jeder Wabe berechnet und eine Vollständigkeitsprüfung durchgeführt. Dafür benötigt ein Transputer insgesamt 27 Sekunden, wobei für die Vorverarbeitung 18 Sekunden, für die Eckenbestimmung 3 Sekunden und für die Messung 6 Sekunden verbraucht werden.

Um die Eckenbestimmung insgesamt nicht zu sehr auszudehnen, ist es verhältnismäßig, eine zusätzliche Sekunde für die globale Verifikation der Eckenzuordnung aufzuwenden. Die erste Restriktion für das Zuordnungsprogramm legt also für den Ablauf 1 Sekunde als Obergrenze fest.

Speicherplatz

Bei der Konfiguration der Rechnerarchitektur fiel die Wahl bei der Festsetzung der Slave-Prozessoren auf den Transputer T800 von Inmos. Die umfangreiche und detaillierte Vermessung der Strukturen erfordert ein komplexes Programmsystem, das aus etwa 60 Modulen besteht, und zu Beginn einer Kontrolle in jeden Transputer geladen wird. Die Größe des Hauptspeichers jedes Transputers wurde mit 4 MByte so festgelegt, daß neben der speicherplatzintensiven Verarbeitung von Long-Bildern (1 Long = 4 Bytes und ein Bild besteht aus 512 x 512 Long-Werte $(512 \times 512 \times 4\text{Byte} = 1\ \text{MByte})$)Erweiterungen der Kontroll- und Meßaufgaben nicht an mangelndem Speicherplatz scheitern. Um eine solche Erweiterbarkeit zu erhalten, wurde die Zurverfügungstellung von 1/4 MByte als angemessen und verhältnismäßig angesehen. Somit begrenzt die zweite Restriktion den für das Zuordnungsprogramm zur Verfügung stehenden Speicherplatz auf 1/4 MByte.

Programmiersprachen

Aus Gründen der Portabilität, der Möglichkeit des strukturierten Programmierens und der nahezu optimalen Übersetzung des Programmcodes in Maschinenbefehle wurde die Programmiersprache "C" gewählt. Die dritte Restriktion schreibt also die Programmiersprache "C" für die Implementierung des Zuordnungsalgorithmus vor.

Software- Qualitätsrichtlinien

Am IDT, Abteilung Intelligente Analyse- und Steuerungssysteme, wurden Richtlinien [Düpm87] erstellt, die die Qualität des Codes, die Handhabbarkeit der Code-Dateien und die Inline-Dokumentation von Quelltexten festlegen. Alle Programmmodule des Meßsystems wurden unter Beachtung dieser Richtlinien erstellt. Zur "nahtlosen" Einfügung der zu implementierenden Algorithmen müssen, und dies ist die vierte Vorgabe, diese Richtlinien befolgt werden.

Zusammenfassung der Vorgaben

Auf einen Blick lassen sich die Vorgaben für das Zuordnungsprogramm folgendermaßen zusammenfassen:

- CPU-Zeit : < 1 Sekunde.
- Speicherplatz : < 250 kByte für Programmcode und interne Daten.
- Programmiersprache : "C"
- Beachtung der Software-Qualitätsrichtlinien des IDT, Abt. Intelligente Analyse- und Steuerungssysteme.

3 Lösungsansatz

Die systembedingten Restriktionen stellen nur einen kleinen Teil der Anforderungen an den gesuchten Zuordnungsalgorithmus dar. In einer Grundlagendiskussion soll zunächst ein Überblick darüber erstellt werden, welche Fehlerquellen zu einer Falschzuordnung von Ist-Ecken zu Soll-Ecken führen können und welche Fehlertypen dabei unterschieden werden. Dies führt schließlich zu einer Spezifikation der Mächtigkeit des gesuchten Verfahrens.

In einer Literaturrecherche werden mehrere OR-Verfahren, die ähnliche Probleme lösen, auf ihre Anwendbarkeit für die vorliegende Problemstellung untersucht und bezüglich der einzelnen Spezifikationsmerkmale beurteilt.

3.1 Grundlagendiskussion

Um die systembedingte Robustheit der Eckenzuordnung umfassend einschätzen zu können ist es hilfreich, den Ecken-Suchalgorithmus zu kennen. Der Ablauf soll am Beispiel der Detektion einer X-Ecke (siehe Bild 1.3), also einer Ecke als Kombination zweier Geraden, gezeigt werden.

Eingabe-Daten sind drei (x / y) -Koordinatenpaare, die durch eine bereits erfolgte Transformationsrechnung einem Translations- und Rotationsanteil des Substratkoordinatensystems zum Tischkoordinatensystem Rechnung tragen, und die, losgelöst von der globalen Strukturgeometrie, die Ecke explizit beschreiben. In einer Parameterberechnung wird aus diesen drei Punkten die Soll-Drehlage der beiden Eckschenkel in Bezug zum Tischkoordinatensystem berechnet. Ausgehend vom Soll-Eckpunkt wird nach Umrechnung der Tisch-Eckpunktskoordinaten in Bild-Eckpunktskoordinaten ein von der Bildauflösung größenabhängiges Suchfenster im zugrundeliegenden Linienbild definiert und alle sich in diesem Fenster befindlichen Linienobjekte pixelweise und objektorientiert in einer Pixel-Liste zusammengefaßt. Eine wichtige Information ist die Gradientenrichtung eines jeden Pixels, die in der Liste neben jedem Pixel eingetragen ist. Die Soll-Gradientenrichtungen der beiden Schenkel der gesuchten Ecke werden nun mit den Gradientenrichtungen aller Pixel, nach Objekten getrennt, verglichen und dort der Ist-Eckpunkt verifiziert, wo die kleinste Abweichung besteht.

Diese Vorgehensweise garantiert die exakte Detektion von Eckpunkten in Objekten, deren Ecken entgegen den Sollvorgaben mehr oder weniger abgerundet sind und liefert gleichzeitig einen Gütewert, der die Exaktheit der Ist-Ecke beurteilt. Die Unabhängigkeit von der globalen Strukturgeometrie läßt die positionsinvariante Detektion auch von kleinsten Teilstrukturen zu, was eine eminent wichtige Anforderung an das Qualitätskontroll- und Meßsystem war.

Die Unabhängigkeit von der globalen Strukturgeometrie läßt dann allerdings die Eindeutigkeit der Eckenzuordnung verlieren, wenn sich mehrere Ecken mit gleichen Parametern auf engstem Raum, d.h. im gleichen Suchfenster, befinden. Im folgenden soll deshalb untersucht werden, welche Zuordnungsfehler welche Ursachen haben, und zwischen welchen Fehlertypen unterschieden werden kann.

3.1.1 Fehlerquellen

Wenn die Strukturgeometrie eine eindeutige Ecken-Zuordnung zuläßt, arbeitet der Ecken-Detektions-Algorithmus "einwandfrei", in dem Sinne, daß beschädigte, nicht exakt ausgebildete oder nicht vorhandene Ecken als solche verifiziert und klassifiziert werden, so daß die weitere bildanalytische Untersuchung auf korrekter Datengrundlage erfolgen und der Strukturfehler als solcher in der detaillierten Auswertung benannt werden kann. Durch die positionsinvariante Ecken-Detektion können Positionierfehler durch Tischungenauigkeit oder einen evtl. Verzug des Substrats erkannt und in Form von Verschiebungsvektoren berechnet werden.

Im anderen Fall der nicht eindeutigen Ecken-Zuordnung treten neben Strukturanomalien oder Substratpositionsverschiebungen auch Fehlerursachen auf, die auf den die globale Strukturgeometrie vernachlässigenden Algorithmus zurückzuführen sind. Das Auftreten beliebiger Kombinationen der genannten Ursachen machen es erforderlich, die Unregelmäßigkeiten in einzelne Fehlertypen zu unterscheiden.

3.1.2 Fehlertypen

a) Typ 1

Eine Ecke wurde gefunden, jedoch aufgrund des schlechten Gütewerts für ungültig erklärt. Dabei können sowohl eine falsche Zuordnung durch den Ecken-Algorithmus als auch eine Positionsverschiebung vorliegen. Bei diesem Sachverhalt ist ein sog. Gültigkeitsflag der Ecke negativ belegt. Der Gütewert hat einen positiven, von Null verschiedenen Wert.

Die Eckdaten werden in diesem Falle bei der weiteren Verifikation aufgrund ihrer Unsicherheit nicht mehr berücksichtigt. Es muß später jedoch noch einmal eine Ecken-Detektion unter neuen Gesichtspunkten (bekannter Substratverschiebungsparameter usw.) durchgeführt werden.

b) Typ 2

Eine Ecke wurde nicht gefunden, der Gütewert beträgt Null. Die Ecke war im Suchfenster offensichtlich nicht vorhanden. Als Ursache kommt sowohl eine Strukturanomalie als auch ein zu kleines Suchfenster in Frage. Wie beim Fehlertyp 1 werden diese Eckdaten für weitere Zuordnungsberechnungen nicht mehr berücksichtigt. Eine spätere Überprüfung muß also nochmals erfolgen.

c) Typ 3

Die Position einer Ist-Ecke tritt in der Eckenliste **n**-fach auf (n = 2,3,...). Das Gültigkeitsflag ist positiv belegt. Der Gütewert ist "plausibel". Die Eckdaten dürfen nicht weiter verwendet werden, da mindestens **n-1** Zuordnungen falsch sind. Ob überhaupt eine Zuordnung richtig ist, läßt sich erst in einer späteren Überprüfung nachvollziehen. Auf jeden Fall müssen alle **n** Sollecken nochmals verifiziert werden.

d) Typ 4

Zwei oder mehrere **m** Ecken wurden jeweils einfach erkannt, aber in der Zuordnung "verwechselt". Da das Gültigkeitsflag positiv belegt ist, der Gütewert plausibel erscheint und keine Mehrfachzuordnung erkennbar ist, handelt es sich hierbei um den am schwierigsten handhabbaren Fehlertyp. Er geht in jede weitere Zuordnungsberechnung ein und stört durch falsche Positionsdifferenzen erheblich den Vorgang der "Wahrheitsfindung". Je mehr Ecken verwechselt wurden, um so unsicherer und langwieriger ist die Geometrieverifikation.

Dieser Fehler kann allerdings nicht auftreten, wenn in einem Suchfenster mehrere Ecken der **gleichen** Ausprägung zu liegen kommen, da der Eckenalgorithmus bei gleichen Voraussetzungen immer das gleiche Resultat (Typ 3) liefert. Erst wenn sich viele Ecken ähnlicher Ausprägung im Suchfenster befinden, wobei die Ähnlichkeit zwischen den Eckparametern sehr groß sein muß, und der Ist-Zustand einer verwechselten Ecke dem ihres Partners gleicht und umgekehrt, kann eine solche schwierige Ausgangslage entstehen. Es muß allerdings darauf hingewiesen werden, daß die Wahrscheinlichkeit solcher Konstellationen äußerst gering ist.

Damit lassen sich direkte Anforderungen an den gesuchten Eckenzuordnungsalgorithmus ableiten, die im folgenden zusammengestellt werden.

3.1.3 Mächtigkeit des gesuchten Verfahrens

In einem ersten Check der Eckenliste muß das Verfahren die Ecken in "gute", d.h. tatsächlich oder vermeintlich (siehe Fehlertyp 4) richtig erkannte und zugeordnete Ecken und in "schlechte", d.h. die durch die Fehlertypen 1, 2 oder 3 verursachten Fehlerdetektionen, klassifizieren können. Nur die "guten" Ecken dürfen Grundlage für das weitere Vorgehen bilden.

Das Verfahren muß ein geeignetes Kriterium benutzen, um aus den "guten" Ecken globale Struktur- und Positionsmerkmale berechnen zu können.

Die "schlechten" Ecken müssen anhand dieser Merkmale und vor dem Hintergrund angepaßter Suchfenstergrößen und vorläufig berechneter Positionsdifferenzen im Bild neu verifiziert werden können.

Durch iteratives Vorgehen soll eine Annäherung an die richtige Zuordnung möglich sein.

Die Forderung nach einer Unempfindlichkeit des Verfahrens gegenüber einer ungleichen Anzahl von Soll- und Ist-Ecken, dem Auflösen von mehrdeutigen Zuordnungen und der Robustheit gegenüber Kombinationen mehrerer Fehlertypen ergibt sich direkt aus obiger Fehlerklassifizierung.

3.2 Literaturrecherche

Wie schon erwähnt beschäftigt sich der Wissenschaftszweig Operations Research hauptsächlich mit der Lösung von Optimierungsproblemen. Charakteristisch für eine OR-Problemstellung sind die Zielfunktion (1) und eine oder mehrere Nebenbedingungen (vgl. [Neum75a] und Anhang B).

$$\Phi() \rightarrow \text{optimieren} \qquad (1)$$

$$\text{unter Nebenbedingungen}$$

Die zu optimierende Zielfunktion Φ ist von Variablen, die die Größe des Problems, z.B. die Anzahl der zu betrachtenden Elemente, beschreiben, und in der Regel von einer Kostenmatrix C, die eine quantitative Beschreibung des Zusammenhangs der einzelnen Elemente untereinander liefert,

abhängig, sowie von der boolschen Variablen $x_{ij} \in \{0,1\}$, welche angibt, ob eine Verbindung zwischen den Punkten **i** und **j** besteht.

Im vorliegenden Fall gibt es zwei Elementgruppen, nämlich **m** Ist-Ecken und **n** Soll-Ecken. Das Element $\mathbf{c_{ij}}$, i = 1,.., m; j = 1,..,n der Kostenmatrix C enthält z.B. die Ortsdifferenz (im Bildraster) zwischen der Ist-Ecke i und der Soll-Ecke j. Als Zielfunktion $\Phi(m,n)$ könnte sich zum Beispiel die Summe aller Ortsdifferenzen im Quadrat, also $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^2$ eignen, wobei bei der Lösungssuche die Kosten minimiert werden müßten. Nebenbedingungen sind die Eindeutigkeit und $n \geq m$.

Somit lautet das Problem

$$\min \ \Phi(m,n) = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} (c_{ij})^2 \qquad (2)$$

u.d.N.

$$\sum_{j=1}^{n} x_{ij} \leq 1, \ i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \ j = 1, \ldots, n$$

$$x_{ij} \in \{0,1\}$$

Damit liegt entsprechend des OR ein sogenanntes schweres Problem vor, das nach heutigem Kenntnisstand exakt nur mit Verfahren zu lösen ist, die eine exponentielle Zeitkomplexität $O(k)=X^k$ besitzen (siehe [Doms81]und Anhang B). Die Zeitkomplexität O beschreibt den Aufwand eines Algorithmus, der in Abhängigkeit der Problemgröße k zu treiben ist. Mit optimierten Verfahren und leistungsstarken Computern lassen sich heute Probleme mit etwa $k \leq 50$ exakt lösen [Doms81]. Bei 200 Ecken, wie sie ein Worst-Case-Bild im vorliegenden Fall besitzt, würde die Berechnung einer Lösung mit den heutigen Maschinen die astronomische Zeit von Millionen von Jahren dauern.

Für schwere Probleme wurden im Bereich des OR deshalb vielfältige Heuristiken entwickelt, die wohl nur an das Optimum angenäherte und nicht exakte Lösungen liefern, die sich allerdings im

realen Zeitbereich bewegen, da sie in der Regel eine polynomiale Zeitkomplexität $O(k) = x \cdot k^y$ aufweisen, wobei y oft $\in \{1,2,3,4\}$ und nicht problemgrößenabhängig ist [siehe Doms81]. Dieser Kompromiß zwischen der Güte der Lösung und dem Rechenaufwand ist faktisch zwingend, da die einzige Alternative (die exakte Lösung) aus Zeitgründen eben nicht realisierbar ist.

Das Augenmerk muß sich im folgenden auf die Suche nach Heuristiken richten. Deshalb sollten sämtliche Teilgebiete des OR überprüft werden. Nach Tomas Gal ([Gal87a]) und Klaus Neumann ([Neum75a]) wird in folgende Gebiete unterteilt:

- Mathematische Optimierungsmodelle
- Transportmodelle (Zuordnungsproblem)
- Graphentheorie (Flußproblem)
- Netzplantechnik
- Netzflußmodelle
- Spieltheorie
- Standortplanung
- Lagerhaltungsmodelle
- Warteschlangenmodelle
- Instandhaltungsmodelle
- Reihenfolgemodelle (Rundreise, Travelling-Salesman)
- Simulation
- Unscharfe Entscheidungsmodelle

Es würde zu weit führen, alle diese Verfahren hier zu beschreiben und zu beurteilen. Deshalb soll anhand der vier erfolgversprechendsten Modelle das Problem angegangen werden, zumal viele der Gebiete nur theoretische Ansätze bieten (wie beispielsweise die mathematischen Optimierungsmodelle) oder nicht auf das Problem anwendbar sind (wie z.B. die unscharfen Entscheidungsmodelle).

Insbesondere wurden folgende Problematiken und Lösungsalgorithmen ausführlich auf die Anwendbarkeit im vorliegenden Fall untersucht: Die Standortplanung, das Travelling-Salesman-Problem, das Zuordnungsproblem und das Flußproblem.

3.2.1 Standortproblem

Problembeschreibung:

Auf einer begrenzten, homogenen Fläche (Ebene) existieren **m** Kunden an Orten mit den Koordinaten (u_j, v_j) für j = 1,.., m. Die Nachfrage des Kunden **j** beträgt $\mathbf{b_j}$ Mengen-Einheiten (ME) pro Periode. Die Transportkosten zwischen allen Punkten der Ebene sind proportional zur transportierten Menge und zur zurückgelegten Entfernung. Die Transportkosten betragen **c** Geld-Einheiten (GE) pro ME und Längen-Einheit (LE).

Das Unternehmen möchte ein Auslieferungslager an einem zu bestimmenden Ort mit den Koordinaten (x,y) so lokalisieren, daß die Gesamtkosten für den Transport der Produkte vom Lager zu den Kunden minimiert werden.

$$\min\ S(x, y) = \sum_{j=1}^{m} b_j \sqrt{(x - u_j)^2 + (y - v_j)^2} \qquad (3)$$

Idee:

Ersetzt man die Kunden durch die Soll-Ecken und die Standorte durch die Ist-Ecken, so könnte man vielleicht, herausfinden, welcher Standort zu welchen Kunden am besten paßt, und zwar so, daß insgesamt die beste (also eine globale) Lösung gefunden wird.
Bild 3.1 zeigt ein Standortproblem der einfachsten Art, vier Kunden mit dem Lager als Mittelpunkt.

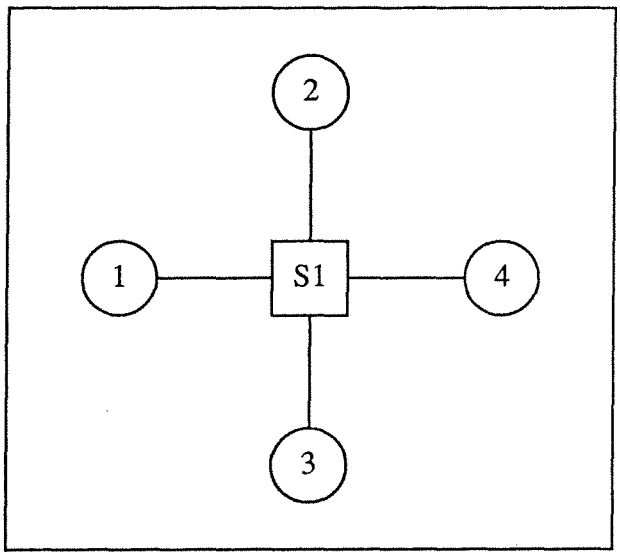

Bild 3.1 Standortproblem

Eine erste Modifikation des Problems ist folgende: Erneut seien m Kunden gegeben, jedoch möchte das Unternehmen **n** Auslieferungslager an zu bestimmenden Punkten mit den Koordinaten (x_i, y_i) für i = 1,.., n so lokalisieren, daß die Gesamtkosten für den Transport von Produkten zwischen Lager und Kunden minimiert werden.

Eine zweite Modifikation führt zu dem Standort-Einzugsbereich-Problem. Die Verallgemeinerung besteht darin, daß nicht von gegebenen Transportmengen ausgegangen wird, vielmehr werden neben Standorten simultan optimale Werte für die Transportvariablen w_{ij} gesucht.

Wenn man für die Entfernung die euklidische Entfernungsmessung d_{ij} zugrunde legt und a_i die Kapazität beschreibt, kann man das Problem wie folgt formulieren:

$$\min\ S'(x,y,w) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot w_{ij} \cdot d_{ij} \quad (4)$$

u.d.N.

$$\sum_{i=1}^{m} w_{ij} = b_j,\ j = 1, \ldots, n$$

$$\sum_{j}^{n} w_{ij} \leq a_i,\ i = 1, \ldots, m$$

$$w_{ij} \geq 0 \quad \forall\ i,j$$

Analogie zum Ecken-Zuordnungsproblem

Beim vorliegenden Ecken-Zuordnungsproblem sind die Standorte fixiert durch die Ist-Punkt-Koordinaten. Die Transportkosten errechnen sich nur aus der Entfernung d_{ij} zwischen Soll- und Ist-Punkten. Deshalb besteht die Kostenmatrix nur aus Einsen, d.h. $c_{ij} = 1, \forall\ i,j$. Daraus ergibt sich dann die neue Zielfunktion:

$$\min S''(x,y,w) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \cdot w_{ij} \qquad (5)$$

u.d.N.

$$\sum_{i=1}^{m} w_{ij} = b_j, \; j = 1,\ldots,n$$

$$\sum_{j=1}^{n} w_{ij} \leq a_i \; , \; i = 1, \ldots, m$$

$$w_{ij} \geq 0 \quad \forall \; i,j$$

Das Modell ist in der hier beschriebenen Form anwendbar auf das Ecken-Zuordnungsproblem. Es existieren auch Verfahren wie zum Beispiel das Verfahren von Cooper (1972) [Doms84]; jedoch ist dieses Verfahren für praktische Probleme zu zeitaufwendig.

Außerdem reduziert sich das Problem durch die Fixierung der Standorte zu einem klassischen Transportproblem. Dieses Problem wird später noch ausführlicher als Zuordnungsproblem in 3.2.3 behandelt.

3.2.2 Travelling-Salesman-Problem (Rundreise-Problem)

Problembeschreibung:

Auf einer begrenzten Fläche existieren **m** Kunden an verschiedenen Orten. Die Transportkosten betragen für die Verbindung zwischen Kunde **i** und Kunde **j** c_{ij} GE.

Das Unternehmen möchte **einen** Vertreter zu **allen n** Kunden schicken. Gesucht wird eine optimale (kostengünstigste) Reiseroute von einem Startort zu den Kunden und zurück. Die so entstehende Tour wird auch Rundreise genannt [Neum75b].

$$\min\ T(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \qquad (6)$$

u.d.N.

$$\sum_{i=1}^{m} x_{ij} = 1,\ j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1, \ldots, m$$

Bed. zur Verhinderung von Kurzzyklen[1]

$$x_{ij} \in \{0,1\}$$

Idee:

Das vorliegende Problem kann folgendermaßen interpretiert werden: Gesucht wird für alle **n** Ist-Knoten **I** mit Startknoten (Ix, Iy) eine Tour zu genau einem End-Knoten **E** mit Koordinaten (Ex, Ey).

Das beschriebene Problem entspricht noch nicht dem gewünschten Problem. Deshalb müssen, wie es im Operations Research üblich ist, Modifikationen durchgeführt werden.

Durch diese Abwandlungen des Standardproblems wird versucht, das eigene Problem zu lösen.

1. Diese sind recht aufwendig in der Beschreibung und werden im nächsten Schritt überflüssig. Deshalb werden sie der Vollständigkeit wegen nur angedeutet.

Die erste Modifikation des Problems betrifft die Vertreter. Erneut seien **m** Kunden gegeben; jedoch möchte das Unternehmen für **n** Vertreter die Rundreise so planen, daß **jeder** Kunde **mindestens einmal** von einem Vertreter aufgesucht wird.

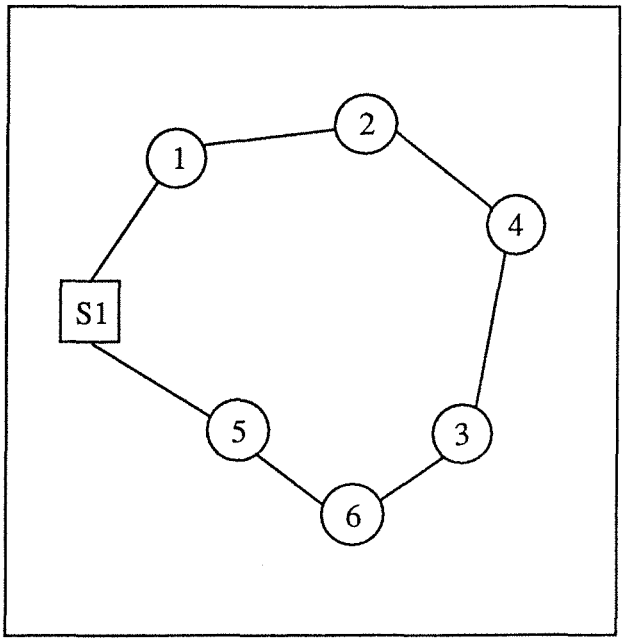

Bild 3.2 Rundreise

Zweitens muß der Standort modifiziert werden. Anstatt **eines** Standorts für alle **n** Vertreter, soll **jeder** Vertreter **einen** eigenen Standort haben (Filialen).

Eine dritte Modifikation betrifft die Touren. Durch Beschränkung einer Rundreise auf **einen** einzigen Kunden entsteht ein modifiziertes Problem, das dem ursprünglichen Problem der Ecken-Zuordnung entspricht.

Bild 3.2 zeigt eine einfache Rundreise (S1,1,2,4,3,6,5,S1) vom Standort (S1) des Vertreters zu den einzelnen Kunden (1, 2, 3, 4, 5 und 6) und zurück.

Das Problem sieht dann folgendermaßen aus:

$$\min \; T'(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \qquad (7)$$

u.d.N.

$$\sum_{i=1}^{m} x_{ij} = 1, \; j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1, \; i = 1, \ldots, m$$

$$x_{ij} \in \{0,1\}$$

Die beiden Gleichungen (6) und (7) sind bis auf die Bedingungen zur Verhinderung von Kurzzyklen identisch. Diese Bedingungen können in Gleichung (7) entfallen weil die drei Modifikationen des Modells sie überflüssig machen.

Analogie zum Ecken-Zuordnungsproblem:

Setzt man in dem vorliegenden Problem die c_{ij} gleich dem euklidischen Abstand zwischen den Punkten **i** und **j**, und setzt man als Startpunkte für die verschiedenen Vertreter die jeweiligen Koordinaten der Soll-Ecken ein, so ergibt sich das gesuchte Problem.

Beurteilung:

Diese Problemstellung ist zwar auf das Problem anwendbar, aber durch die Modifikationen ist hieraus das Zuordnungsproblem entstanden, das in 3.2.3 behandelt und beurteilt wird.

3.2.3 Das Zuordnungsproblem

Problembeschreibung:

Ein Unternehmen erstellt ein Produkt. Dafür stehen ihm **n** Maschinen zur Verfügung, an denen **m** Tätigkeiten verrichtet werden müssen. Jede Maschine ist aber nur für manche Tätigkeiten einsetzbar oder nur sehr ineffizient verwendbar, d.h. die Kosten c_{ij} geben an, wie "teuer" die Ausführung von Tätigkeit **i** auf Maschine **j** ist. Geht man nun weiter davon aus, daß eine Maschine jeweils nur eine Tätigkeit ausführen kann (z.B. wegen hoher Umrüstzeiten), so heißt das, daß **m = n** sein muß.

Ziel für das Unternehmen ist nun die Ermittlung der Verteilung der Tätigkeiten auf die Maschinen, also die Beantwortung der Frage, welche aller möglichen Zuordnungen die kostengünstigste ist.

Das Problem läßt sich dann wie folgt darstellen[1]:

$$\min Z(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \qquad (8)$$

u.d.N.

$$\sum_{i=1}^{m} x_{ij} = 1, \; j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1, \; i = 1, \ldots, m$$

$$x_{ij} \in \{0,1\}$$

1. Identisch zu Gleichungssystem 7. Gleichung 6 wurde so vereinfacht, daß gerade das Zuordnungsproblem, gegeben durch Gleichung 8, entsteht.

Die Problemdarstellung des Zuordnungsproblems entspricht genau dem des gesuchten Problems. Bild 3.3 zeigt zwei Tätigkeiten (1,2) im gepunkteten Rechteck, sowie die Zuordnung zu den Maschinen (3,4) im abgerundeten Rechteck. Die Beschriftung (**a** / **b**) an den Pfeilen bedeutet: die Zuordnung einer Einheit kostet **a** GE und die Kapazität beträgt **b** ME.

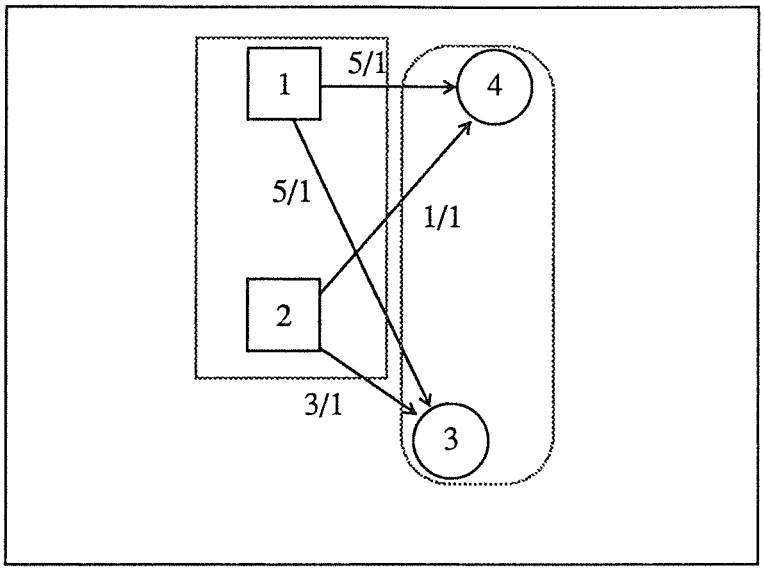

Bild 3.3 Zuordnungsproblem

Analogie zum Ecken-Zuordnungsproblem:

Ersetzt man Tätigkeiten und Maschinen durch Soll- bzw. Ist-Ecken, so bedarf es lediglich noch einer guten Bestimmung der Kosten.

Eine Möglichkeit für die Kostenbestimmung wäre, den euklidischen Abstand $\left(d(\mathrm{I,S}) = \sqrt{(I_x - S_x)^2 + (I_y - S_y)^2}\right)$ zwischen den beiden Eckpunkten zu nehmen. Noch besser allerdings ist die Verwendung des quadratischen Abstandes $(d^2(\mathrm{I,S}))$, denn hierdurch wird nicht nur das Rechnen einfacher und schneller, sondern auch die Unterscheidung der Kosten deutlicher, eine hohe Differenz (also Fehlzuordnung) geht jetzt mit wesentlich höheren Kosten einher.

Beurteilung:

Das lineare Zuordnungsproblem ist im Operations Research hinreichend bekannt und es existieren mehrere Lösungsansätze für das Problem, sowohl exakte Verfahren als auch schnellere Heuristiken. Alle diese Verfahren lassen sich in zwei Kategorien einteilen: die eine Gruppe beruht auf dem Simplexverfahren und die andere Gruppe benutzt den graphentheoretischen Ansatz.

Bei den Lösungen, die auf dem Simplexverfahren basieren, ist anzumerken (siehe [Neum75a] und [Neum77]), daß als Zwischenlösung für das Simplexproblem nur entartete Ecken (siehe Anhang B) auftreten, wodurch die Anzahl der Verfahrensschritte **erheblich** vergrößert wird.
Der graphentheoretische Ansatz zeigt einen besseren (weniger aufwendigen) Weg auf.

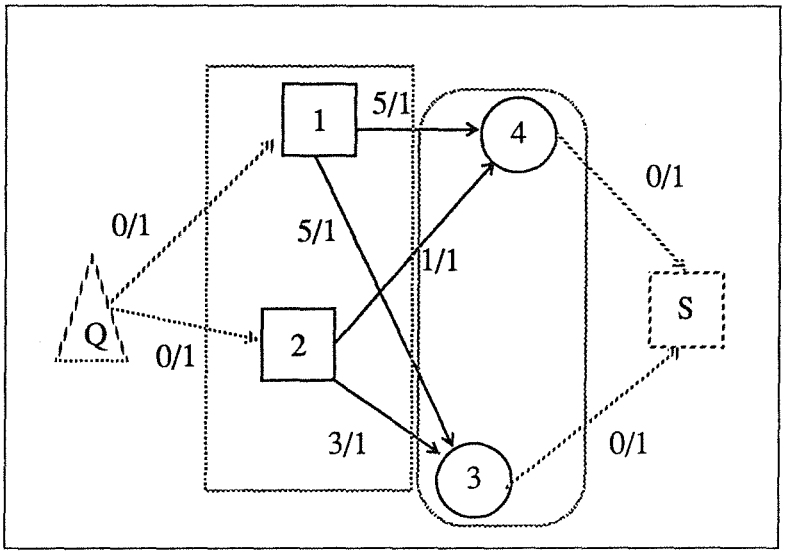

Bild 3.4 Umwandlung des Zuordnungsproblem zum Flußproblem.

Es empfiehlt sich daher, das Problem in ein Flußproblem [Neum75b] umzuwandeln. Dies geschieht durch die Einführung einer sogenannten "Super-Quelle", in Bild 3.4 als Dreieck dargestellt, und einer "Super-Senke", als gestricheltes Rechteck dargestellt. Ansonsten sind die Bezeichnungen wie in Bild 3.3. Die Super-Quelle hat eine kostenlose Verbindung zu allen Ist-Knoten und gibt 1ME an diese weiter. Analoges gilt für die Super-Senke, die über eine kostenlose Verbindung zu den Soll-Knoten 1 ME abnimmt. Die Knoten im gepunkteten Rechteck entsprechen den Soll-Ecken und jene Knoten im abgerundeten Rechteck entsprechen den Ist-Ecken des ursprünglichen Problems.

Dies stellt nun ein neues Problem dar, das im folgenden Unterabschnitt 3.2.4 behandelt wird.

3.2.4 Das Flußproblem

Problembeschreibung:

Man hat einen Anbieter und einen Abnehmer sowie verschiedene Zwischenstationen (Umladestationen). Die Nachfrage des Abnehmers beträgt ω ME und die Transportkosten pro Teilstrecke $\mathbf{c_{ij}}$ GE für die Strecke (<i, j >) von Knoten **i** nach Knoten **j.** Der Transport der Waren vom Anbieter zum Abnehmer erfolgt über die Umladestationen und die Verbindungen zwischen ihnen. Die Menge, die auf einem solchen Weg transportiert werden kann, ist begrenzt (Kapazitäten) und mit Kosten versehen [Neum75b].

Die Frage lautet nun, wie kann der Anbieter alle seine Güter zum Abnehmer transportieren, so daß die Gesamtkosten für die Beförderung unter Berücksichtigung der Kapazitäten der Transportverbindungen minimal sind (kostenminimaler Fluß mit Stärke ω)?

$$\min\ F(\hat{x}) = \sum_{ij \in E} c_{ij} \cdot \hat{x}_{ij} \qquad (9)$$

u.d.N

$$\sum_{j \in S(i)} \hat{x}_{ij} - \sum_{k \in P(i)} \hat{x}_{kj} = \begin{cases} \omega & \text{für } i = r \\ 0 & \text{für } i \in V \setminus \{r,s\} \\ -\omega & \text{für } i = s \end{cases}$$

$$\left.\begin{array}{c} \lambda_{ij} \le \hat{x}_{ij} \le \kappa_{ij} \\ \hat{x}_{ij} \ge 0 \end{array}\right\} \ (<i,j> \in E)$$

E:	Kantenmenge;	V:	Knotenmenge;
r:	Quelle;	s:	Senke;
λ_{ij}:	Minimalkapazität von <i, j>;	κ_{ij}:	Maximalkapazität von <i, j>;
S(i):	Menge der Nachfolger von Knoten i;	P(i):	Menge der Vorgänger des Knoten i;

Idee:

Wählt man die Umladestationen, die von der Quelle bedient werden, als Ist-Ecken und die, welche an die Senke abgeben, als Soll-Ecken, so braucht man nur noch die Stärke ω des Flusses auf **n** (Anzahl der Ecken) und die Kapazitäten richtig zu setzen, um die gewünschte Problemstellung zu erhalten.

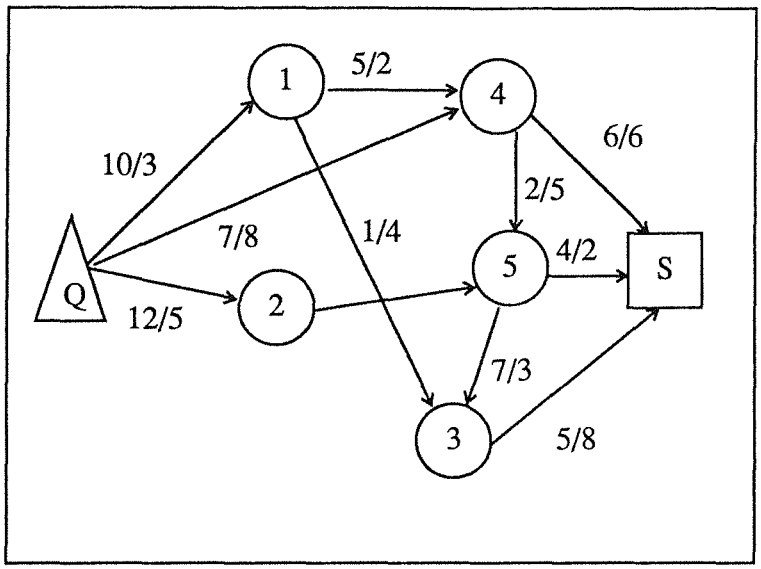

Bild 3.5 Flußproblem.

Das Bild 3.5 zeigt das Flußproblem ähnlich wie Bild 3.4.

Die Modifikation des Problems ist folgende: die Minimalkapazitäten werden gleich 0 und die Maximalkapazitäten auf 1 gesetzt. Da immer nur ganze Güter verschickt werden können, bedeutet dies eine Binärisierung der Variablen $\hat{x}_{ij}$. Daraus folgt[1]

$$\min \; F'(x) \; = \sum_{ij \in E} c_{ij} \cdot x_{ij} \qquad (10)$$

u.d.N

$$\sum_{j \in S(i)} x_{ij} - \sum_{k \in P(i)} x_{kj} = \begin{cases} \omega & \text{für } i = r \\ 0 & \text{für } i \in V \setminus \{r,s\} \\ -\omega & \text{für } i = s \end{cases}$$

$$x_{ij} \in \{0,1\}$$

Analogie zum Ecken-Zuordnungsproblem:

Ersetzt man jene Knoten aus der Menge **I**, der Nachfolger der Quelle, mit den Ist-Ecken und den entsprechenden Knoten aus der Menge **S**, der Vorgänger der Senke, mit den Soll-Ecken, so braucht man nur die Kantenmenge zwischen diesen beiden Knotenmengen **I** und **S** so zu erweitern, daß von jedem Knoten aus **I** eine Kante zu jedem Knoten aus **S** geht.

---

1. Gleichung 10 ähnlich Gleichung 9, aber x_{ij} nur noch 0 oder 1.

Als Wert für die Transportkosten c_{ij} eignet sich wohl am besten der quadratische Abstand zwischen beiden Knoten.

Beurteilung:
Diese Vorgehensweise ist durchaus auf das Problem anwendbar.

Im allgemeinen ist eines der schnellsten Verfahren zur Lösung eines kostenminimalen Flußproblem der Stärke ω der Out-Of-Kilter-Algorithmus mit einem Aufwand von $O(n^2\omega)$. Bei Anwendung auf das Zuordnungsproblem verändert sich die Zeitkomplexität damit auf $O(n^3)$ [Gal87b].

Weitaus günstiger sind spezielle Abwandlungen des Out-Of-Kilter-Verfahrens, wie z.B. die Ungarische Methode [Gal87b, Doms81, Neum75a]. Der Nachteil bei dieser Methode ist ein Speicheraufwand von $O(n^2)$, d.h. bei 200 Ecken ein Mindestspeicherplatz von 160 kByte nur für die Kostenmatrix.

Domschke [Doms81] erwähnt hier noch ein schnelleres Verfahren, nämlich das Verfahren von Tomizawa (1971) speziell für das Zuordnungsproblem in einer verbesserten Ausführung von Dorhout (1973). Implementiert in Fortran (1980) benötigt es jedoch einen Speicherplatz von $n^2 + 8n$.

3.2.5 Schlußfolgerungen

Damit erfüllen alle diese Verfahren nicht die Randbedingungen des Systems. Wie aus den vorherigen Betrachtungen der Aufgabenstellung deutlich wird, ist das Problem zwar lösbar, aber man kann den gestellten Anforderungen nicht gerecht werden, insbesondere den Anforderungen nach Zeit- oder Speicheraufwand. Deshalb muß für die vorliegende Problemstellung ein anderer Weg gegangen werden. Eine solche Möglichkeit wird im nächsten Kapitel beschrieben.

4 Formulierung des Algorithmus

Wie in Kapitel 3 gezeigt, lassen sich keine der vorgefundenen Lösungsmöglichkeiten verwenden, um das gegebene Problem zu lösen. Deshalb ist es notwendig, einen eigenen Weg zu finden.

Woran kann man dabei anknüpfen?

Erstens gilt es zu beachten, daß das Verfahren zur Eckenbestimmung ein sehr zuverlässiges Verfahren ist, das wenig Fehleranfälligkeit aufweist. Dies bedeutet, daß eine Art Korrektur-Verfahren genügt, um die Fehler zu erkennen und zu beheben.

Zweitens sind die Fehler, die auftreten können, auf relativ einfache Art und Weise zu erkennen, so daß keine aufwendigen Routinen hierfür benötigt werden.

Drittens liefern die vielen "guten" Zuordnungen ein recht eindeutiges Bild von der Lage und somit der Translation (mit der Methode der kleinesten Quadrate).

Mit diesen drei Erkenntnissen kann man zu einem heuristischen Verfahren gelangen, das mittels sukzessiverVerbesserungen zum Ziel führt und das folgendermaßen aussieht:

Schritt 1:	Erkennen der fehlerhaften Zuordnungen.
Schritt 2:	Aus den übrigen Zuordnungen Informationen gewinnen bezüglich der geometrischen Situation.
Schritt3:	Unter Ausnutzung der in Schritt 2 gewonnenen Informationen werden die als fehlerhaft erkannten Zuordnungen nochmals bestimmt.
Schritt 4:	Wiederholung von Schritt 1-2-3, bis keine Korrektur mehr möglich.

Wie sich bei umfangreichen Vortests herausstellte, war es nötig, den Schritt 3 zu verfeinern und solche Zuordnungen als fehlerhaft anzusehen, deren Abstand (zwischen den entsprechenden Ecken) größer ist als die der Mehrheit.

Diese Trennung konnte allerdings nicht so einfach vorgenommen werden wie gedacht: es stellte sich heraus, daß eine Zwei-Klassen Trennung nötig ist. Als ein gutes "Trenn-Verfahren" erwies sich der Algorithmus von Otsu [Otsu79].

Wichtig ist jedoch noch ein zweiter Algorithmus, der die geometrische Lage der beiden Strukturen (Soll und Ist) zueinander auswertet. Hierfür wurde die Routine **cacoe2del** verwendet [Guth89].

Verfahren

Das gesamte Verfahren zur Zuordnung der Ist- zu den Soll-Ecken läßt sich dann wie folgt beschreiben (siehe auch Bild 4.1):

Schritt 1: Im Hauptprogramm werden die Bilder soweit vorbereitet und verarbeitet, bis eine erste Zuordnung durch die Eckenbestimmungs-Routine erfolgt ist.

Schritt 2: Dann erfolgt die Überprüfung der Ergebnisse dieser Zuordnung, d.h. direkte Kontrolle der Ergebnisse auf nicht-gefundene oder mehrdeutige Eckenzuordnungen.

Schritt 3: Die restlichen Eckenpaare werden zur Bestimmung der geometrischen Lage ausgewertet.

Schritt 4: Unter Verwendung der in Schritt 3 gewonnenen Daten erfolgt mittels einer "Diskriminanzanalyse" die Bestimmung einer Trennschwelle, um die restlichen Eckenzuordnungen in die zwei Gruppen "gute" und "schlechte" aufzuteilen.

Schritt 5: Trennung der "guten" und "schlechten" Zuordnungen, d.h. die schlechten werden auch als fehlerhafte Zuordnungen angesehen.

Schritt 6: Neubestimmung der fehlerbehafteten Zuordnungen unter Berücksichtigung der in Schritt 3 gewonnenen Kenntnisse über die Lage der Ecken.

Schritt 7: Wiederholung der Schritte 2 bis 6, bis keine Fehlerkorrektur mehr möglich.

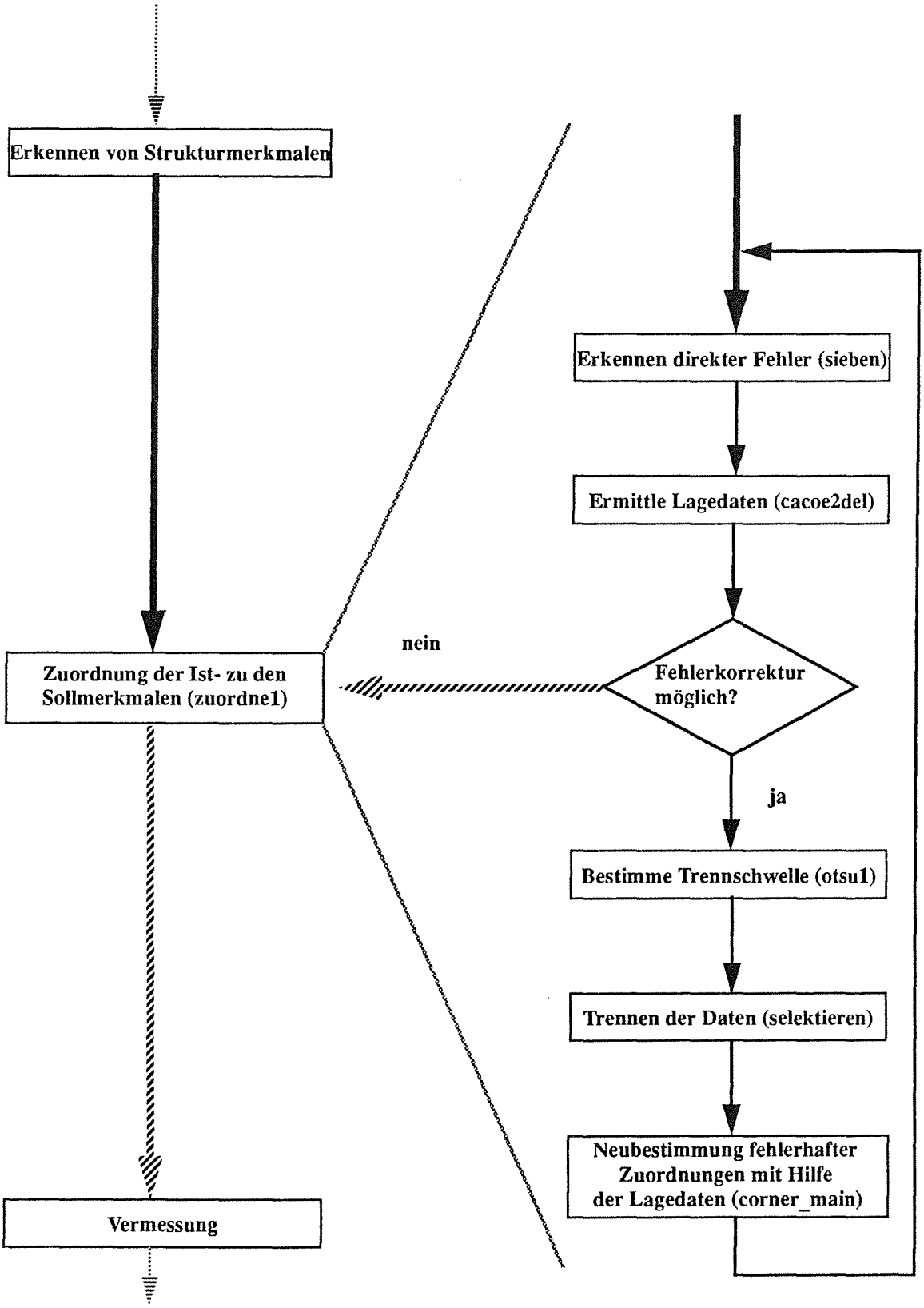

Bild 4.1: Überblick über den Ablauf des Verfahrens zur korrekten Zuordnung.

otsu1

Die Routine **otsu1**[Otsu79] ist ein statistischer Algorithmus, der eine Trennung in zwei Güteklassen vornimmt. Sie wird auch bei der Schwellenbestimmung aus Grau-Wert-Histogrammen in der digitalen Bildverarbeitung benutzt und ist nach Nobuyuki Otsu benannt.

Das Modell benutzt die Diskriminanzanalyse, um eine Trennung in Klassen durchzuführen.

Weitere Informationen des Verfahrens sind:

- Wahrscheinlichkeiten für die Klassen.
- Durchschnittswerte für jede Klasse.
- Qualitätsmerkmal für die Trennbarkeit.

Das Verfahren von Otsu wurde aus zwei Gründen verwendet:

- Die Methode arbeitet auch dann sehr erfolgreich, wenn aus dem Histogramm (der Abweichungen) keine offensichtlichen Klassenzuordnungen möglich erscheinen.
- Die Methode ist bereits erfolgreich bei anderen Verfahren eingesetzt worden und ihr Verhalten ist hinlänglich bekannt.

cacoe2del

Die Routine **cacoe2del** [Guth89] berechnet die Koeffizienten einer 2-dimensionalen Transformations-Matrix (Rotation und Translation). Mit Hilfe der Punktepaare co[n][2] und co_t[n][2] (x-y Koordinaten von Punkten) werden die Koeffizienten pa[i] (i = 0..5) der Transformationsmatrix berechnet, die die Punkte co in die Punkte co_t transformiert.

Außerdem werden die Quadrate der Abstände der beobachteten und berechneten Transformationspunkte berechnet.

Die Koeffizienten pa[i] werden mit Hilfe der Methode der kleinsten Quadrate gefittet, wobei die Abstände der entsprechenden transformierten co-Punkte und übergebenen co_t-Punkte minimiert werden.

Für die Transformation gilt:

$$\begin{bmatrix} co\_t\text{-}x \\ co\_t\text{-}y \end{bmatrix} = \begin{bmatrix} pa[0] & pa[1] \\ pa[3] & pa[4] \end{bmatrix} \times \begin{bmatrix} co\text{-}x \\ co\text{-}y \end{bmatrix} + \begin{bmatrix} pa[2] \\ pa[5] \end{bmatrix}$$

4.1 Vorbemerkungen

Anfangszustand:

Beim Aufruf von **corner_aufruf ()** ist schon ein Großteil der Arbeit des übergeordneten Systems erfolgt. Die Vorarbeit zum Detektieren von Ecken ist erledigt, und in den Übergabeparametern stehen die bis dahin gewonnenen Daten.

Bevor nun die einzelnen Routinen in Abschnitt 4.2 beschrieben werden, erfolgt eine Übersicht über den Aufbau der Beschreibung. Die Beschreibung einer Routine ist unterteilt in folgende Abschnitte:

Zweck:	Erläuterung des Zwecks der Routine.
Aufwand:	Angabe von Zeit- und Speicherplatzkomplexität.
Datenschnittstelle:	Beschreibung benötigter Informationen (I = Input, O = Output, L = Lokal)
Algorithmus:	Beschreibung des Verfahrens entsprechend der Syntax aus Anhang A.

4.2 Ablauf

4.2.1 Die aufrufende Routine aus dem Hauptprogramm (corner_aufruf)

Zweck :

Die Routine corner_aufruf ermittelt selektiv für jede Soll-Ecke eine Ist-Ecke. Die Zuordnung wird überprüft und evtl. korrigiert.

Datenschnittstelle :

Anzahl der Soll_dateneinträge:	ns_d	I

Algorithmus :

corner_aufruf ();

Für alle i *von* 1 *bis* ns_d *führe aus:*

falls Kennung vom Typ Ecke *führe aus:*

Bestimme Ecke (**corner_main ()**);

Kontrolliere und korrigiere fehlerhafte Zuordnung (**zuordne1()**).

4.2.2 Die Eckenbestimmungsroutine des Hauptprogramms (corner_main)

Zweck :

Die Routine **corner_main** berechnet Suchfenster, erstellt Pixel-Liste, ermittelt für eine gegebene Soll-Ecke eine Ist-Ecke, und besorgt Umrechnung der Koordinatensysteme.

Algorithmus :

corner_main ();

Berechne die Suchfenster-Koordinaten (berechne_window());

Ermittle die Objekte im Suchfenster (ketten_listen());

Suche Ecke des entsprechenden Typs (corner());

Führe Umrechnung Bild- in Tischkoordinaten durch (bild_tisch_koord());

4.2.3 Die Hauptroutine für die korrekte Zuordnung (zuordne1)

Zweck :

Zuordne1 (Bild 4.1) überprüft unter Umständen in mehreren Durchgängen anhand globaler Geometriebetrachtungen die Zuordnung zwischen Soll- und Ist-Ecken. Fehlerhafte Zuordnungen werden nochmals einer Zuordnung zugeführt, wobei jedoch gewonnene Erkenntnisse über die tatsächliche Lage einfließen. Aufgrund dieser Daten ist es möglich, die Größe des Suchfensters zu verringern und das Fenster besser zu positionieren. Dadurch reduzieren sich die Fehlermöglichkeiten enorm.

Aufwand :

Zeitkomplexität: $O(k * n^2)$

Speicherkomplexität: $O(l * n)$

Datenschnittstelle :

Maximale Anzahl der Iterationsschritte: MaxIteration L

Algorithmus :

zuordne1 ();
- Sortiere Mehrfachzuordnungen und ungültige Ecken aus (**sieben()**);
- Berechne Transformationsmatrix (Rotation + Translation) durch globale Überprüfung (**cacoe2del()**);
- *Solange* (Fehleranzahl verbessert) und (Iteration < MaxIteration) *tue:*
  - Suche Trennschwelle (**otsu1()**);
  - Trenne zusätzlich anhand der Trennschwelle auch die vermeintlich richtig zugeordneten Ecken. (**selektieren()**);
  - *Für alle* fehlerhaft zugeordneten Ecken *führe aus:*
    - Nochmalige Eckenbestimmung mit reduzierter Fenstergröße und berechnetem Verzug. (**corner_main()**);
  - Sortiere Mehrfachzuordnungen und ungültige Ecken aus (**sieben()**);
  - Berechne Verzug und Fehlermatrix durch globale Überprüfung (**cacoe2del()**);
  - Inkrementiere Iteration.

4.2.4 Die Routine zum Aussortieren direkter Fehler (sieben)

Zweck :

Die Routine **sieben** filtert die Soll-Ecken aus, die keine **eindeutige** Zuordnung besitzen und jene Ecken, die als nicht gefunden markiert sind.

Sie überprüft in einem ersten Schritt, ob die Ecke gefunden wurde. Dies geschieht durch Abprüfen einer in eda gesetzter Marke. Wurde eine Ecke als nicht gefunden markiert, so wird sie an die Liste lb angehängt. Eine als gefunden markierte Ecke wird mittels der Routine **inserth** in eine Hilfsmenge lh eingetragen, eine Menge deshalb um Mehrfachzuordnungen abfangen und als fehlerhafte Zuordnungen behandeln zu können. Nach der Fehlererkennung werden die Koordinatenpaare der zugehörigen "guten" Soll- und Ist-Ecken in ls bzw. li zur weiteren Verarbeitung in den übergeordneten Routinen eingetragen.

Aufwand :

Zeitkomplexität:	O ($n^{2)}$
Speicherkomplexität:	O (k)

Datenschnittstelle :

Feld für Ist-Ecken-Einträge:	eda	I
Anzahl der Einträge in eda:	eda_numb	I
Zeiger auf Liste mit Soll-Koordinaten:	ls	O
Zeiger auf Liste mit Ist-Koordinaten:	li	O
Liste mit Indizes der fehlerhaften Eda-Elemente:	lb	O
Anzahl der fehlerhaft zugeordneten Ecken:	nlb	O
Zeiger auf Hilfsspeicher, um Mehrdeutigkeiten zu erkennen:	lh	L

Algorithmus :

sieben();

Für alle eda-Elemente *führe aus:*

falls Eintrag = ungültig *führe aus:*

Ergänze lb um eda-Index und erhöhe nlb;

sonst

Trage eda-Index ein in lh (**inserth()**);

Für alle Elemente von lh *führe aus:*

falls mehrfach verwendet *führe aus:*

Ergänze lb um eda-Index und erhöhe nlb;

sonst

Schreibe entsprechende Koordinaten in ls und li.

4.2.5 Die Routine zur Bestimmung der geometrischen Lage (cacoe2del)

Zweck :

Die Routine **cacoe2del** berechnet die Koeffizienten einer 2-dimensionalen Transformationsmatrix pa (mit Translation), welche Koordinatenpaare co in Koordinatenpaare co_t überführt und erstellt zusätzlich eine Liste, in die die quadratischen Differenzen zwischen den transformierten Soll-Ecken und den Ist-Ecken eingetragen werden. Diese Differenzliste del wird in einer nachfolgenden Routine als Merkmals-Liste für die Diskriminanzanalyse benützt.

Aufwand :

Zeitkomplexität:	O (n)
Speicherkomplexität:	O (n)

Datenschnittstelle :

Zeiger auf X/Y -Koordinatenpaaren (Ist-Ecken):	co	I
Zeiger auf transformierte Koord. (Soll-Ecken):	co_t	I
Anzahl der Wertepaare:	n	I
Liste von Abstandsquadraten zwischen beobachteten und transformierten Eckpunkten:	del	O
gefittete Koeffizienten (Transformationsmatrix):	pa	O

Algorithmus :

cacoe2del ();

für i *von* 0 *bis* n-1 *führe aus:*

Ergänze Summenspeichern mit den Ist- und

Soll-Koordinatenpaaren;

Berechne die inverse Matrix (MATINVC_D(), MATMULC_D());

Errechne daraus pa-Matrix-Elemente;

für i *von* 0 *bis* n-1 *führe aus:*

Berechne die quadratische Abweichung zwischen den transformierten

Ist- und den Soll-Koordinaten und schreibe in del.

4.2.6 Die Routine für die Diskriminanzanalyse (otsu1)

Zweck :

Die Routine **otsu1** führt eine Diskriminanzanalyse durch, welche die Liste von Abstandsquadraten del in zwei Klassen aufteilt. Sie verwendet hierzu das Verfahren von Otsu und gibt die Trennschwelle schwelle zurück.

Aufwand :

Zeitkomplexität:	O (k * n)
Speicherkomplexität:	O (l)

Datenschnittstelle :

Liste von Abstandsquadraten zwischen beobachteten und transformierten Eckpunkten:	del	I
Anzahl der Elemente in del:	n	I
Wert der Trennschwelle:	schwelle	O
Arbeitsbereich für Histogramm:	lhist	L

Algorithmus :

otsu1();

Initialisiere Ihist;

Bestimme Maximum der Liste (**maximum()**);

Setze Skalierungsfaktor;

Skaliere Abweichungsliste in Histogramm;

Ermittle Wahrscheinlichkeiten für Klassen und

optimiere Schwelle nach Otsu.

4.2.7 Die Routine zur Ermittlung eines größten Wertes einer Liste (maximum)

Zweck :

Die Routine **maximum** liefert den maximalen Wert der Liste, sie geht dabei sequentiell durch die Liste del.

Aufwand :

Zeitkomplexität:	O (n)
Speicherkomplexität:	O (k)

Datenschnittstelle :

Zeiger auf Liste der Abweichungen:	del	I
Anzahl der Elemente in del:	n	I
maximaler Wert in Liste:	max	L

Algorithmus :

maximum();

Setze max = Wert vom ersten Element;

für i *von* 1 *bis* n *führe aus:*

falls max < Wert von Element i *führe aus:*

Setze max = Wert von Element i;

Gebe max zurück;

4.2.8 Die Routine zur Überprüfung von Mehrfachzuordnungen (inserth)

Zweck :

Die Routine **inserth** dient der Erkennung von Mehrfachzuordnungen. Sie überprüft sequentiell, ob eine Ist-Ecke schon in der Hilfsmenge lh eingefügt ist. Falls ja, wird das gefundene Element als fehlerhaft markiert (nicht gefunden) und der Repräsentant in der Hilfsmenge markiert. Falls nein, wird die Ecke in die Menge aufgenommen und die Anzahl der Elemente erhöht.

Aufwand :

Zeitkomplexität:	O (n)
Speicherkomplexität:	O (k)

Datenschnittstelle :

Zeiger auf Hilfsspeicher, um Mehrdeutigkeiten zu erkennen:	lh	I
Zeiger auf nächste freie Stelle von lh:	lh_zg	O
Anzahl Elemente in lh:	lh_n1	O
Anzahl eindeutiger Elemente in lh:	lh_n2	O
Zeiger auf Fehlerliste:	lb	O
Anzahl der Fehler in Fehlerliste:	nlb	O

Algorithmus :

inserth ();

Solange nicht gefunden und noch nicht alle Elemente kontrolliert *tue:*

falls x-Koordinate und y-Koordinate wie Element in Liste *führe aus:*

Setze gefunden auf Wahr;

falls noch nicht markiert *führe aus:*

Erniedrige lh_n2;

Markiere Element;

Trage Eda-Index in Fehlerliste lb ein;

Erhöhe nlb;

Markiere Eda-Element als nicht gefunden;

sonst

Inkrementiere Zeiger des Listenelements;

falls nicht gefunden *führe aus:*

Trage Eda_Index ein in Liste und erhöhe lh_n1 und lh_n2.

4.2.9 Die Routine zur Trennung in zwei Klassen (selektieren)

Zweck :

Die Routine **selektieren** sortiert sequentiell die Zuordnungen aus, bei denen die Abweichungen del zwischen den transformierten Soll-Ecken und den Ist-Ecken zu sehr abweichen von der Mehrzahl der Zuordnungen, also die Abweichungen größer sind als die Trennschwelle schwelle.

Datenschnittstelle :

Zeiger auf Eda-Liste:	eda	I
Anzahl der Elemente in Eda:	eda_numb	I
Zeiger auf Abweichungen:	del	I
Wert der Trennschwelle:	schwelle	I
Zeiger auf Fehlerliste:	lb	O
Anzahl der Fehler:	nlb	O

Aufwand :

Zeitkomplexität:	O (n)
Speicherkomplexität:	O (k)

Algorithmus :

selektieren();

Für alle i *von* 0 *bis* Anzahl-1 gute Ecken *führe aus:*

falls Abweichung größer Schwelle *führe aus:*

Schreibe Eda-Index in lb;

Erhöhe nlb;

Nächstes Eda-Element.

5 Implementierung

Bei der Implementierung von Algorithmen muß neben den Software-Richtlinien auch die Rechner-Architektur beachtet werden. Dies gilt um so mehr, wenn die Programme sehr schnell sein müssen und nur wenig Speicherplatz belegen dürfen.

Das Programm, in dem die Routinen eingebunden werden, besteht grob aus zwei Teilen, einem Verwaltungsteil (auf dem Master-Rechner) und dem eigentlichen Verarbeitungsteil (auf den Slaves).

Die Routinen, die eingebaut werden müssen, gehören zur zweiten Kategorie. Sie werden also auf den Transputern zum Einsatz kommen und folglich deren Anforderungen erfüllen müssen.

5.1 Transputer

Es handelt sich bei den Transputern um T800 von INMOS, mit einem Hauptspeicher von je 4 MByte. Es wird der C-Compiler von 3L Ltd, ausgelegt für Parallele Programmierung [3L88], benutzt.

Datentypen

Aufgrund der Zugriffsstruktur der Transputer sind die Datentypen vom Standard abweichend implementiert:

char	Byte	(unsigned)
int	word	
short	word	
long	word	
float	word	(IEEE single-precision format)
double	double	(IEEE double-precision format)
pointer	word.	

Wobei auf dem T800 1 **word** = 4 **Byte** sind [3L88].

Diese Einschränkung ist insoweit von Belang, als durch die Benutzung des **short**-Typs für Laufvariablen und zur Speicherung der Listen-Indizes kein Speicherplatz gegenüber **long** gespart wird. Dadurch werden aufwendigere Speichermethoden nötig (z.B. 2 **Int**-Zahlen in 1 **long**-

Variable speichern), oder es muß auf **long** (4 Byte) ausgewichen werden. Zudem werden Gleitkomma-Zahlen im Transputer immer mit doppelter Genauigkeit gerechnet, so daß bei Verwendung von **float** rechnerintern eine Umwandlung des Formats erfolgen muß.

In dieser Arbeit wurde zugunsten der Arbeitsgeschwindigkeit auf die platzsparendere Form der Speicherbelegung verzichtet und überall statt **float double** und statt **int long** verwendet.

Speicher

Für die optimale Ausnutzung der Rechengeschwindigkeit des Transputers ist die Beachtung der Speicherbelegung ausschlaggebend. Wie in Bild 5.1 zu erkennen, ist der Speicher in mehrere Bereiche aufgeteilt. Insbesondere ist der untere Abschnitt von Interesse, der 'on board' realisiert ist. Dieser Bereich, der 4 kByte umfaßt und in den der Stack gelegt wird, wird vom Transputer besonders schnell verarbeitet. Bei der Belegung dieses Bereichs muß mit Bedacht vorgegangen werden. Nur oft verwendete Variablen der Routine sollten in diesem Bereich gespeichert werden.

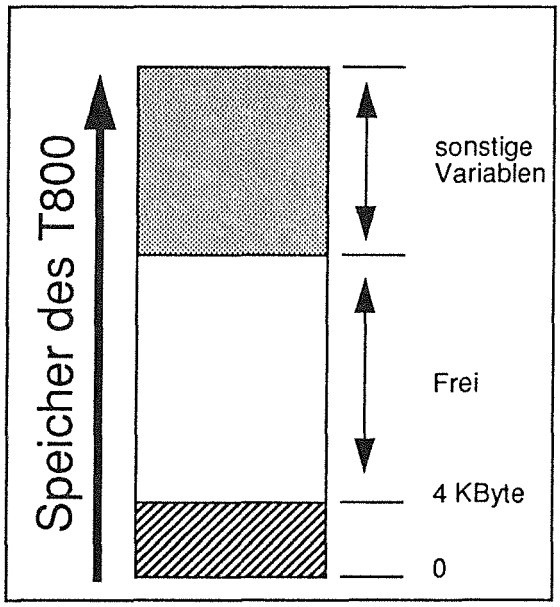

Bild 5.1 Speicherbelegung des T800

Pointer

Die Programmiersprache 'C' bietet die Möglichkeit, Feldzugriffe mittels Pointerbenutzung zu beschleunigen. Beim T800 bringt die jedoch kein Vorteil, hier sind beide Arten gleich schnell und deshalb wurde (ohne Verlust von Geschwindigkeit) die lesbarere Version, nämlich die mit der Indizierung verwendet.

Kommunikation

Wie bereits erwähnt, besteht das Gesamtsystem aus zwei Programmteilen, die nur eingeschränkt miteinander kommunizieren können. Wie sieht nun ein ganz normaler Arbeitsvorgang aus?

In einem ersten Schritt werden vom Master verschiedene Variablen und Felder initialisiert. Danach werden die Transputer gebootet und mit dem Verarbeitungsteil geladen. Anschließend werden vom System die gewünschten Bilder eingelesen, mit den jeweiligen Bildverarbeitungsbefehlen verbunden und an den ersten freien Transputer übermittelt. Dies geschieht im Takt von ca. 0.7 Sekunden.

Ist ein Transputer innerhalb der vorgeschriebenen Zeit fertig, liefert er das Resultat an den Master-Rechner zurück. Die von den Transputern errechneten Resultate werden dann vom Master archiviert. Dies stellt die ganze Kommunikation, die zwischen Master und Slave abläuft, dar.

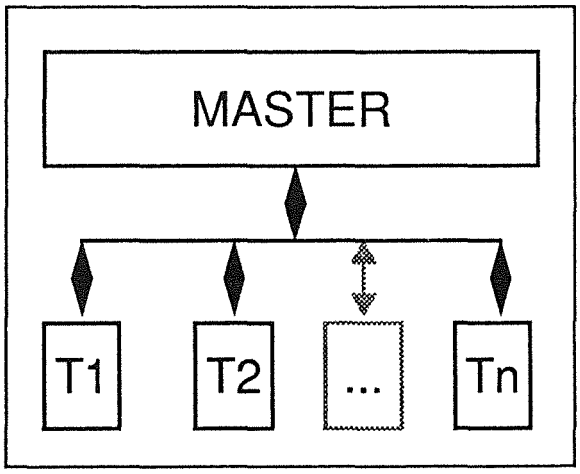

Bild 5.2 vereinfachte Darstellung der Datenübergabe zwischen Master und Slaves.

Vor allem sind keine weiteren Verbindungen zur Peripherie vorgesehen. Diese Tatsache bedeutet für die Tests ein erhebliches Manko. Deshalb wurde vom Systemersteller eine einfache Ausgabe-Routine geschrieben, mit der man über den VME-Bus einfache Ausgaben auf dem Monitor des Masters machen kann. Eingeschränkt ist die Ausgabe sowohl im Umfang als auch in den Datentypen, die zur Verfügung stehen, nämlich nur **char**, **string** und **long** in Printf-Format. Der große Nachteil hiervon ist die Asynchronität zwischen Ausgabe und wirklichem Zustand des Systems. Modernere Tools wie Debugger standen leider auf dem Zielsystem nicht zur Verfügung.

5.2 Test

In diesem Kapitel sollen die Tests bezüglich der Funktionalität der Routinen beschrieben werden. Diese Tests bestehen aus zwei Teilen. Erstens ein Vortest (der im Vorfeld geschah) zur Überprüfung von **cacoe2del**, um zu erkennen, ob dieser Ansatz überhaupt verwendbar ist und zum Schluß eine Überprüfung des gesamten Verfahrens.

5.2.1 Vortest

Nach dem umfangreichen Studium der Operations-Research-Modelle zur Problemstellung blieb nichts anderes übrig, als einen eigenen Weg zur Lösung zu suchen. Bei der Ausarbeitung des Algorithmus wurde es ratsam, im Vorfeld der Implementierungen eine wichtige Routine auf ihre Anwendbarkeit zu überprüfen. Die Routine heißt **cacoe2del** und berechnet die Koeffizienten der Transformationsmatrix, die die Koordinaten der Soll-Punkte in die der Ist-Punkte überführt. Es wurde ein Bild mit dem ursprünglichen System aufgenommen und so weit vorverarbeitet bis zur jetzigen Schnittstelle in **corner_aufruf**. Dort wurden die benötigten Daten mittels der Ausgabe-Routine auf dem Master-Monitor ausgegeben und von dort in eine Datei umgelenkt. Die Daten wurden anschließend auf eine Sun-Workstation portiert, wo aufgrund besserer Hilfen Tests durchgeführt wurden. Es wurde eine Testumgebung geschaffen und mit den Daten experimentiert, um die Routine **cacoe2del** zu testen.

Vorgehensweise:

Da die Daten von einem fehlerfreien Bild stammten, wurden drei Änderungen angebracht.

Erstens wurde die Datei in verschiedene Größen unterteilt; getestet wurde die Abhängigkeit der Routine von der Anzahl der Elemente, die eingingen. Da maximal 216 Eck-Daten zur Verfügung standen, andererseits die Routine mindestens drei Werte benötigt, wurden folgende Aufteilungen durchgeführt: 10, 20, 50, 100 und alle Elemente.

Die zweite Änderung bestand in der bewußten Abänderung (durch Übernehmen des Nachbarwertes) der Eck-Daten. Dies wurde mit folgender Anzahl von Fehlern getestet: 0,1,2,3,4,5,10,20.

Schließlich wurden beide vorangehenden Änderungen kombiniert, sofern dies möglich war. Durch diese Kombination entstand eine gute Beurteilungsbasis für die Routine **cacoe2del**.

Resultate

- Je mehr Elemente zur Beurteilung herangezogen wurden, um so genauer und besser war das Auffinden von abweichenden Elementen. Insbesondere durch den **quadratischen** Abstand waren die Abweichungen, die auftraten, auffällig. War bei den "guten" Ecken die signifikante Abweichung 4-5-stellig, so war sie bei den fehlerhaften 8-stellig. Unter Abweichung versteht man hier die quadratische Differenz zwischen den transformierten Soll-Koordinaten und den Ist-Koordinaten, die in **cacoe2del** berechnet wird.
- Je mehr Fehler, um so schlechter die Aussagemöglichkeit der Routine. Insbesondere wenn die Anzahl der Fehler sich der Anzahl der guten Ecken-Zuordnungen näherte. Hier muß dazu gesagt werden, daß der angenommene Fehler, eine Vertauschung oder Mehrfachzuordnung, die am häufigsten vorkommende Fehlerquelle ist und die Tests also eine hohe Aussagekraft besitzen.

Schlußfolgerungen:

Aus diesen Vortests konnte man drei wichtige Erkenntnisse gewinnen:

- Diese Tests waren im Hinblick auf die Funktionalität der Routine **cacoe2del** sehr zufriedenstellend.
- Sortierte man die mehrdeutigen Ecken aus und gab sie folglich nicht an **cacoe2del** weiter, fand eine weitere (wenn auch geringe) Verbesserung des Ergebnisses statt, da der Einfluß der falschen Daten auf die Transformationsmatrix reduziert wurde.
- **Cacoe2del** arbeitet solange zufriedenstellend, wie die Anzahl der "guten" die Anzahl der "schlechten" Zuordnungen übertrifft.

5.2.2 Haupttest

Nach der Implementierung und dem Austesten der Routine, konnte man mit der Überprüfung der Güte des Algorithmus in der Praxis beginnen.

Vorgehensweise:

Als erstes wurde getestet, um wieviel Ecken die um die neuen Teile erweiterte Routine besser ist, als die frühere Version ohne dieses Korrekturverfahren. Nicht förderlich für die Tests war die Tatsache, daß die vorhandenen Daten so gut waren, daß nur in Ausnahmefällen eine Verbesserung möglich war.

Um extreme Fehlersituationen zu erzeugen, war es notwendig, die Ablaufparameter des Gesamtsystems zu manipulieren.

Dies geschah auf zweierlei Art:

- Einmal durch Vergrößerung des Suchfensters (wodurch die Anzahl der Mehrdeutigkeiten bei den Ecken-Zuordnungen erhöht werden konnte).
- Zum zweiten durch manuelle Verschiebung des Substrats vor der Aufnahme mit der Kamera (also eine zwangsweise Erhöhung der Translation).

Resultate

Erläuterungen für die Interpretation der Testdaten:

1. Zeile : *Halbwert:..* *; Hilfhalbwert:..*; Halbwert ist dabei die halbe Kantenlänge des quadratischen Suchfensters in Pixel; Hilfhalbwert ist die halbe Kantenlänge des verkleinerten Suchfensters.

2. Zeile : Vergrößerung:.. ; ux:.. ;uy:..; Unter Vergrößerung ist die Objektiv-Vergrößerung zu verstehen, welche noch mit dem Faktor 10 (Okular) multipliziert werden muß, um die Gesamtvergrößerung des Mikroskops zu erhalten. Ux und Uy stehen für die Koordinaten der Mitte des Bildes, das bearbeitet wird, nicht die Mitte des Suchfenster.

Außer diesen beiden Zeilen gibt es noch vier weitere Zeilentypen:

Zeile : Schleifenvariable | Prozedur: ..Zeit in Millisekunden

Zeile : Schleifenvariable | Angabe über die Ecken nach Typ und Art des Fehler aufgelistet (siehe Abschnitt 1.4 und Bild 1.3).

Zeile : Schleifenvariable | sonstige interessante Variablen.

Zeile : Loop[Schleifenvariable] time = Gesamtzeit für einen Schleifendurchgang

Die Schleifenvariable zeigt an, in welchem Durchlauf die Routine sich befindet. Zu den interessanten Variablen gehören die ermittelte Translation in x und y Richtung (in negierter Schreibweise), sowie die neuen Mittelpunkte für das Bild: u2x und u2y. Die Angaben sind in 'Hektoångström' (10^{-8}m).

Es würde zuviel Platz in Anspruch nehmen, sämtliche Testvariationen an den Testfiguren zu zeigen. Deshalb wurde hier ein Beispiel ausgewählt, an dem besonders gut deutlich wird, welche

Vor- und Nachteile die Ergänzung besitzt. Im Anhang C stehen jedoch zusätzlich zu einigen wichtigen Testfiguren die unkommentierten Testdaten.

Testfigur: Eine Spirale mit der Objektiv-Vergrößerung 20 und einer leichten Verschiebung in x-Richtung. Bei dieser Serie wurde nur die **Suchfenstergröße** variiert und die restlichen Parameter wurden beibehalten.

```
Halbwert: 100;Hilfhalbwert : 50
    Vergroesserung :   20 ;ux :  9386402 ;uy :  5067984
 0 | sieben       : Zeit =    14
 0 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 1/ 9
 0 | cacoe2del   : Zeit =    12  ; Verzug x =   1240   ; Verzug y =   -128
 1 | otsu1        : Zeit =    34   ; Schwelle/1000 =     4101
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 0/ 1
 1 | u2x =  9385162 ; u2y =  5068112
 1 | corner_main : Zeit =   941
 1 | sieben       : Zeit =    16
 1 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 2
 1 | cacoe2del   : Zeit =    13  ; Verzug x =   1154   ; Verzug y =   -127
loop[ 1]-time    :  1012
 2 | otsu1        : Zeit =    34   ; Schwelle/1000 =     3766
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 0/ 1
 2 | u2x =  9385248 ; u2y =  5068111
 2 | corner_main : Zeit =   282
 2 | sieben       : Zeit =    17
 2 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 2
 2 | cacoe2del   : Zeit =    13  ; Verzug x =   1147   ; Verzug y =   -113
loop[ 2]-time    :   353
Zuordnel: Ende   ; Zeit = 1599
```

```
Halbwert :  90;Hilfhalbwert :  45
    Vergroesserung :   20 ;ux :  9386402 ;uy :  5067984
 0 | sieben       : Zeit =    14
 0 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 1/ 8
 0 | cacoe2del   : Zeit =    12  ; Verzug x =   1170   ; Verzug y =   -129
 1 | otsu1        : Zeit =    33   ; Schwelle/1000 =     7551
```

```
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  0/ 0/ 0
 1 | u2x =  9385232 ; u2y =  5068113
 1 | corner_main : Zeit =   640
 1 | sieben      : Zeit =    17
 1 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 2
 1 | cacoe2del   : Zeit =    13  ; Verzug x =   1079   ; Verzug y =   -111
loop[ 1]-time    :   710
 2 | otsu1       : Zeit =    34   ; Schwelle/1000 =     6373
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 0/ 1
 2 | u2x =  9385323 ; u2y =  5068095
 2 | corner_main : Zeit =   218
 2 | sieben      : Zeit =    16
 2 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 2
 2 | cacoe2del   : Zeit =    13  ; Verzug x =   1095   ; Verzug y =   -111
loop[ 2]-time    :   288
Zuordne1: Ende   ; Zeit = 1074
```

```
Halbwert :  80;Hilfhalbwert :  40
    Vergroesserung :   20 ;ux :  9386402 ;uy :  5067984
 0 | sieben      : Zeit =    16
 0 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 7
 0 | cacoe2del   : Zeit =    12  ; Verzug x =   1173   ; Verzug y =   -144
 1 | otsu1       : Zeit =    34   ; Schwelle/1000 =     1960
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  0/ 0/ 1
 1 | u2x =  9385229 ; u2y =  5068128
 1 | corner_main : Zeit =   426
 1 | sieben      : Zeit =    16
 1 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 4
 1 | cacoe2del   : Zeit =    12  ; Verzug x =   1171   ; Verzug y =   -168
loop[ 1]-time    :   497
 2 | otsu1       : Zeit =    34   ; Schwelle/1000 =     2024
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 0/ 0
 2 | u2x =  9385231 ; u2y =  5068152
 2 | corner_main : Zeit =   218
 2 | sieben      : Zeit =    16
 2 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 4
```

```
 2 | cacoe2del   : Zeit =    13  ; Verzug x =   1171   ; Verzug y =   -168
loop[ 2]-time    :   288
Zuordnel: Ende   ; Zeit = 980
```

```
Halbwert :  70;Hilfhalbwert :  35
    Vergroesserung :   20 ;ux :  9386402 ;uy :  5067984
 0 | sieben       : Zeit =    16
 0 | x/y/z nicht gefunden =  0/ 1/ 1 ; mehrdeutig =  0/ 0/ 6
 0 | cacoe2del   : Zeit =    11  ; Verzug x =   1107   ; Verzug y =   -125
 1 | otsu1        : Zeit =    34   ; Schwelle/1000 =     5732
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  0/ 1/ 0
 1 | u2x =  9385295 ; u2y =  5068109
 1 | corner_main : Zeit =   343
 1 | sieben       : Zeit =    16
 1 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 2
 1 | cacoe2del   : Zeit =    13  ; Verzug x =   1073   ; Verzug y =   -134
loop[ 1]-time    :   415
 2 | otsu1        : Zeit =    34   ; Schwelle/1000 =     3483
 2 | selektieren : Zeit =     1, aussortierte x/y/z =  0/ 1/ 0
 2 | u2x =  9385329 ; u2y =  5068118
 2 | corner_main : Zeit =   135
 2 | sieben       : Zeit =    17
 2 | x/y/z nicht gefunden =  0/ 1/ 0 ; mehrdeutig =  0/ 0/ 2
 2 | cacoe2del   : Zeit =    13  ; Verzug x =   1091   ; Verzug y =   -151
loop[ 2]-time    :   206
Zuordnel: Ende   ; Zeit = 697
```

```
Halbwert :  60;Hilfhalbwert :  30
    Vergroesserung :   20 ;ux :  9386402 ;uy :  5067984
 0 | sieben       : Zeit =    16
 0 | x/y/z nicht gefunden =  0/ 1/ 1 ; mehrdeutig =  0/ 0/ 4
 0 | cacoe2del   : Zeit =    12  ; Verzug x =   1115   ; Verzug y =    -97
 1 | otsu1        : Zeit =    34   ; Schwelle/1000 =     5663
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  0/ 1/ 0
 1 | u2x =  9385287 ; u2y =  5068081
```

```
 1 | corner_main : Zeit =   191
 1 | sieben      : Zeit =    16
 1 | x/y/z nicht gefunden =  0/ 1/ 1 ; mehrdeutig =  0/ 0/ 4
 1 | cacoe2del   : Zeit =   13  ; Verzug x =   1125   ; Verzug y =   -110
loop[ 1]-time    :   261
Zuordnel: Ende   ; Zeit = 316
```

```
Halbwert :  50;Hilfhalbwert :  25
    Vergroesserung :   20 ;ux :  9386402 ;uy :  5067984
 0 | sieben      : Zeit =     9
 0 | x/y/z nicht gefunden =  0/ 1/19 ; mehrdeutig =  0/ 0/ 8
 0 | cacoe2del   : Zeit =    9  ; Verzug x =    779   ; Verzug y =   -235
 1 | otsu1       : Zeit =    34   ; Schwelle/1000 =     1047
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  0/ 1/ 0
 1 | u2x =  9385623 ; u2y =  5068219
 1 | corner_main : Zeit =   596
 1 | sieben      : Zeit =    14
 1 | x/y/z nicht gefunden =  0/ 1/ 5 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del   : Zeit =   12  ; Verzug x =    697   ; Verzug y =   -158
loop[ 1]-time    :   664
 2 | otsu1       : Zeit =    34   ; Schwelle/1000 =     1237
 2 | selektieren : Zeit =     1, aussortierte x/y/z =  0/ 0/ 1
 2 | u2x =  9385705 ; u2y =  5068142
 2 | corner_main : Zeit =   138
 2 | sieben      : Zeit =    15
 2 | x/y/z nicht gefunden =  0/ 1/ 5 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del   : Zeit =   12  ; Verzug x =    686   ; Verzug y =   -141
loop[ 2]-time    :   207
Zuordnel: Ende   ; Zeit = 1060
```

Durch die Größe des Suchfensters wird auch die Suchzeit für eine Ecke bestimmt. Dies ist einleuchtend, wenn man bedenkt, daß in einem doppelt so großen Fenster viermal so viele zu untersuchende Punkte liegen. Nicht nur der Zeitverbrauch nimmt zu, sondern auch die Gefahr von Mehrdeutigkeiten. Dies veranschaulicht auch die folgende Tabelle in Bild 5.3. Aufgelistet sind hier nach Fenstergröße die mehrdeutig zugeordneten Ecken in den jeweiligen Durchläufen, sowie die

in den jeweiligen Durchläufen nicht gefundenen Ecken. Erkennbar wird hier der Verlauf der Korrektur sowie der Einfluß der Fenstergröße auf die mehrdeutigen Ecken, sowie auf die nicht gefundenen Ecken. Desweiteren zeigt sich, daß die mehrdeutigen Ecken sich in wenigen Schritten verbessern lassen. Der Schleifendurchgang wird in Bild 5.3 durch Anhängen der Schleifenzahl an die Bezeichnungen dargestellt.

halbe Fenstergröße	50	60	70	80	90	100
Mehrdeutig_0	8	4	6	7	9	10
Mehrdeutig_1	0	4	2	4	2	2
Mehrdeutig_2	0	-	2	0	2	2
nicht gef. 0	20	2	2	1	1	1
nicht gef. 1	6	2	1	1	1	1
nicht gef. 2	6	-	1	1	1	1

Bild 5.3 Tabelle der Fehlerhäufigkeiten in Abhängigkeit der Fenstergröße

Schlußfolgerungen:

Aus diesen Tests kann man verschiedene Erkenntnisse gewinnen:

1. Je größer das Suchfenster, um so mehr mehrdeutige Ecken werden gefunden.
2. Je größer das Suchfenster, um so besser (bis zum Umkipppunkt) wird eine Translation aufgefangen. Der Umkipppunkt bezeichnet dabei die Suchfenstergröße, ab der sich die Ergebnisse der Routine **corner_main** drastisch verschlechtern.
3. Mehrdeutigkeiten infolge von großen Suchfenstern werden sehr schnell, meist in einem Durchlauf, aufgelöst.
4. Nicht gefundene Ecken, die zwar vorhanden, jedoch nicht im normalen Suchfenster auffindbar sind, werden meist in wenigen Durchläufen gefunden.
5. Ecken, die in der Routine **selektieren** aussortiert werden, weil sie weiter abweichen, als die anderen und vermutlich falsch zugeordnet sind, werden in der Regel bei der nachfolgenden Neuzuordnung gefunden.

6. Treten viele Fehler auf und ist hohe Symmetrie auf dem Bild vorhanden, so kann eine Verwechslung der Spalten / Reihen auftreten, welche sich in einigen nicht gefundenen Ecken manifestiert, sowie in einer Translation, die zu groß / klein ist. Dieses Problem ist mit den vorhandenen Ausschnittsbildern nicht lösbar.
7. Die mit Abstand zeitintensivste Routine ist die vorgegebene Routine **corner_main** zur Bestimmung der Ecken.

5.3 Bewertung

Zeitaufwand

Wie die vielfältigen Tests zeigten, ist der Zeitverbrauch der Routine Zuordne1 gering. Nach Abzug des Zeitbedarfs für die externe Routine **corner_main,** die iterativ zur Eckenbestimmung aufgerufen wird, liegt der Zeitaufwand im Bereich von wenigen hundert Millisekunden. Ausnahmen entstehen, wenn die Routine **corner_main** oft aufgerufen werden muß, da sie sehr zeitintensiv ist. Je größer das Suchfenster ist, umso länger benötigt diese Routine zum Finden der gesuchten Ecke im Fenster. Als weitere zeitintensive Routine trat die Routine **sieben** hervor. Der eigentliche Verbraucher dieser Zeit ist die **inserth** Routine, welche die gefundenen Ecken auf Mehrdeutigkeit überprüft. Je mehr "gute" Ecken vorhanden sind, um so länger braucht sie.

Speicherplatzaufwand

Der Speicherplatzaufwand ist relativ gering; er liegt bei 3n **long**- und 5n **double**-Felder (für die von der Eckenanzahl n abhängigen Speicherallokationen), 1*1000 **long**-Feld für das Histogrammfeld und ein kleiner Rest für sonstige lokale Variablen. Der benutzte Speicherplatz wird mit dem Verlassen der Routine **zuordne1** wieder freigegeben, so daß nachfolgende Programmteile wieder den gesamten Speicherraum für sich beanspruchen können.

Bei 'Worst-Case'-Bildern treten Werte für n in der Größenordnung von 200 auf. Unter Berücksichtigung der Platzbelegung für **long**-(4Byte) und **double**-Werte (8Byte) wird weniger als 20 KB Speicherplatz für die Variablen benötigt.

Der Speicherplatzbedarf für den Programm-Code der einzelnen Module sieht folgendermaßen aus:

zuordne1	8702
sieben	2807
inserth	1897
cacoe2del	3063
histogra	2928
select	1849
corner_aufruf	2928
---	---
Insgesamt	24174

Der Speicherplatzbedarf für Daten und Programm liegt somit unter 50 kByte und damit weit unter dem geforderten Limit von 1/4 MByte.

Fazit

Die Routine **zuordne1** liegt, sowohl was den Zeit- als auch den Speicheraufwand angeht, in den vorgegeben Grenzen. Das System leistet gute Arbeit bei der Korrektur von Bildern mit Translation oder sonstigen Fehlern und benötigt für die vielen Bilder ohne Fehler oder Verzug kaum Zeit.
Das Verfahren hat sich somit als sehr schnell und effektiv gezeigt.

6 Ausblick

6.1 Verbesserungen

Die Routine inserth.

Die Routine **inserth** kann mit einer verbesserten Speicherstruktur für die Koordinatenpaare wesentlich beschleunigt werden. Es wäre z.B. denkbar, hier eine verbesserte Speicherstruktur, ähnlich der bei geometrischen Datenbanken, zu benutzen. Die Realisierung einer solchen Speicherstruktur mitsamt ihren zugehörigen Funktionen übersteigt allerdings jeden vernünftigen Aufwand, zumal der Zeitgewinn im Verhältnis zum Zeitbedarf der Ecken-Bestimmungsroutine **corner_main**, verschwindend gering ist.

Das Abbruchkriterium der Schleife in der Routine zuordne1.

Das Abbruchkriterium reagiert im Moment nur auf Veränderungen in der Gesamtfehleranzahl. Wird diese in einem Schritt nicht mehr verbessert, so bricht die Schleife ab. Trotzdem könnte es noch mehrdeutige Ecken geben, die man auf jeden Fall noch korrigieren könnte.

Folglich sollte man das Abbruchkriterium so abändern, daß die Routine nur dann abbricht, wenn zusätzlich noch die Bedingung gilt, daß keine Mehrdeutigkeiten mehr vorhanden sind. Am besten geschieht dies in Verbindung mit einem immer kleiner werdenden Suchfenster. Die Suchfenstergröße sollte dabei abhängen von der Iterationszahl, vielleicht errechnet nach folgender Formel:

$$\text{Neue\_Suchfenster\_Größe} = \frac{\text{Standard\_Suchfenster\_Größe}}{\text{Iterations\_Zahl} + 1}.$$

Zu beachten wäre hierbei allerdings eine minimale Suchfenstergröße, die zum einwandfreien Bestimmen der Ecken benötigt wird.

Die Weitergabe zusätzlicher statistischer Daten.

Die Routine stellt nebenbei noch einige zusätzliche Daten bereit, die dem übergeordneten System zur Verfügung gestellt werden könnten. Beispielweise Informationen über den errechneten Verzug im bearbeiteten Bild oder Daten, die Auskunft geben über die Häufigkeit, mit der die verschiedenen Ecktypen (siehe 1.4) als nicht gefunden markiert wurden. Als Zusatz könnte die Häufigkeit noch nach der Fehlerursache (nicht vorhanden, mehrdeutig oder ausselektiert aufgrund einer zu großen Abweichung vom Gesamtbild) aufgeschlüsselt werden.

6.2 Perspektiven

Bei dem vorliegenden Ecken-Zuordnungs-Problem handelt es sich um ein allgemeines Geometrieverifikationsproblem. Es taucht überall dort auf, wo in der Bildverarbeitung und Mustererkennung mit schnellen und speicherplatzsparenden heuristischen Methoden sichere Zuordnungen von Strukturmerkmalen zwischen Soll- und Ist-Bildern unter globalen geometrischen Gesichtspunkten hergeleitet werden sollen.

Aufgrund einer ständig wachsenden Nachfrage der Industrie nach Bildanalysesystemen, die unter Einsatz moderner Hard- und Software in der Lage sind, Rationalisierungsbestrebungen vor allem in der Qualitätskontrolle zu unterstützen, wird das hier vorgestellte Verfahren sicher auch außerhalb der vorliegenden Anwendung seine Bedeutung haben.

Anhang A : Syntaktische Beschreibung

Das Verfahren ist der Verständlichkeit wegen in natürlicher Sprache beschrieben, aber an höhere Programmiersprachen wie Pascal oder C angelehnt.

While-Schleife:

Solange Bedingung *tue:* Befehl1; Befehl2; Befehl3;	Wenn die Bedingung erfüllt ist, führe Befehl1 und Befehl2 aus, bis die Bedingung falsch wird und die Bearbeitung mit Befehl3 weitergeht.

For-Schleife:

Für alle A *von* B *bis* C *führe aus:* Befehl1; Befehl2; Befehl3;	Setze A = B ; Solange "A < C" tue: Befehl1; Befehl2; erhöhe A um 1; Befehl3;

If-Konstruktion:

falls Bedingung *führe aus:* Befehl1; Befehl2; *sonst* Befehl3; Befehl4; Befehl5;	Wenn die Bedingung wahr ist, dann führe Befehl1 und Befehl2 aus, wenn die Bedingung falsch ist, führe Befehl3 und Befehl4 aus. Anschließend weiter mit Befehl5;

Jeder Befehl kann wieder ein Konstrukt oder eine Beschreibung der Tätigkeit sein, die ausgeführt wird. Hinter der Beschreibung steht in Klammern der Name der zuständigen Routine. Ist der Name in **Fettdruck** geschrieben, so wird auch diese Routine vollständig in gleicher Weise erläutert. Ist der Name unterstrichen, so handelt es sich um eine dem übergeordneten System zugehörige Bibliotheksfunktion, die nicht weiter erklärt wird.

Anhang B : Operations Research

Allgemeine Vorgehensweise bei der Problemlösung.

Im Operations Research wird für ein zu lösendes Problem zuerst nach ähnlichen, aber bereits gelösten Problemen gesucht. Wurde ein solches gefunden, wird das Problem, eventuell in mehreren Schritten (Modifikationen), so angepaßt, daß es in die ursprüngliche Problemstellung überführt wird [Gal87a].

Entartete Ecken

Im $\Re^n$ versteht man unter einer entarteten Ecke eine Ecke, die durch mehr als n Hyperebenen bestimmt ist, also überbestimmt ist [Gal87a]. Dies bedeutet für das zugehörige Lineare Optimierungssystem, daß noch eine Basis existiert, zu der die Werte der Basisvariablen und der Wert der Zielfunktion gleich sind.

Schwierige Probleme und vermutlich schwierige Probleme (schwere Probleme).

Unter der Menge der vermutlich schwierigen Probleme versteht man die Klasse der NP-Vollständigen Probleme [Doms81]. Bekannte Vertreter dieser Gruppe sind das Lineare-Umlade-Problem und das Travelling-Salesman-Problem. Es sind Probleme, für die kein Algorithmus bekannt ist, der garantieren kann, daß das Problem immer in polynomialer Zeit gelöst werden kann.

Im Gegensatz hierzu, ist bei den schwierigen Problemen bekannt, daß kein polynomialer Algorithmus existiert.

Komplexität von Algorithmen

Bei der Komplexität von Algorithmen gibt es zwei Arten: die **Zeitkomplexität**, die den maximalen Rechenaufwand in Abhängigkeit von der Größe des Problems angibt, und die **Speicherplatzkomplexität**, die den maximalen Verbrauch an Speicherplatz in Abhängigkeit von der Größe des Problems angibt. Die Schreibweise und die Berechnung ist für beide Begriffe analog, so daß an dieser Stelle nur noch die Zeitkomplexität erläutert werden soll.

Die Schreibweise: O (f(n)) bedeutet, daß die Zeitkomplexität die Größenordnung f(n) annimmt, wenn sie asymptotisch proportional zu f(n) ist [Doms81].

Anhang C : Test-Reihen

Untersuchtes Objekt: Wabe

```
Halbwert :  80;Hilfhalbwert :  40
    Vergroesserung :    5 ;ux :  5316754 ;uy :  4872823
 0 | sieben       : Zeit =    56
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 164/ 0/ 0
 0 | cacoe2del    : Zeit =     9  ; Verzug x =   -957   ; Verzug y =    577
 1 | otsu1        : Zeit =    34   ; Schwelle/1000 =   154493
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  5317711 ; u2y =  4872246
 1 | corner_main : Zeit =  8294
 1 | sieben       : Zeit =   125
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =    37  ; Verzug x =    666   ; Verzug y =   -475
loop[ 1]-time    :  8498
 2 | otsu1        : Zeit =    36   ; Schwelle/1000 =   131881
 2 | selektieren : Zeit =     2, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  5316088 ; u2y =  4873298
 2 | corner_main : Zeit =    50
 2 | sieben       : Zeit =   124
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del    : Zeit =    37  ; Verzug x =    692   ; Verzug y =   -422
loop[ 2]-time    :   255
Zuordne1: Ende   ; Zeit = 8879

Halbwert :  70;Hilfhalbwert :  35
    Vergroesserung :    5 ;ux :  5316754 ;uy :  4872823
 0 | sieben       : Zeit =    52
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 174/ 0/ 0
 0 | cacoe2del    : Zeit =     8  ; Verzug x =   -855   ; Verzug y =    436
 1 | otsu1        : Zeit =    34   ; Schwelle/1000 =   130950
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  5317609 ; u2y =  4872387
 1 | corner_main : Zeit =  6766
 1 | sieben       : Zeit =   124
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =    36  ; Verzug x =    814   ; Verzug y =   -372
loop[ 1]-time    :  6968
 2 | otsu1        : Zeit =    36   ; Schwelle/1000 =   100775
 2 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  5315940 ; u2y =  4873195
 2 | corner_main : Zeit =    39
 2 | sieben       : Zeit =   124
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del    : Zeit =    36  ; Verzug x =    833   ; Verzug y =   -384
loop[ 2]-time    :   244
Zuordne1: Ende   ; Zeit = 7349

Halbwert :  60;Hilfhalbwert :  30
    Vergroesserung :    5 ;ux :  5316754 ;uy :  4872823
 0 | sieben       : Zeit =    68
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 147/ 0/ 0
 0 | cacoe2del    : Zeit =    13  ; Verzug x =  -2623   ; Verzug y =   -171
 1 | otsu1        : Zeit =    34   ; Schwelle/1000 =   125951
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  5319377 ; u2y =  4872994
 1 | corner_main : Zeit =  4327
 1 | sieben       : Zeit =   123
 1 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =    37  ; Verzug x =   1199   ; Verzug y =    -60
loop[ 1]-time    :  4529
```

```
 2 | otsu1       : Zeit =    36   ; Schwelle/1000 =   102957
 2 | selektieren : Zeit =     2, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  5315555 ; u2y =  4872883
 2 | corner_main : Zeit =    59
 2 | sieben      : Zeit =   122
 2 | x/y/z nicht gefunden =  2/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del   : Zeit =    37  ; Verzug x =   1252   ; Verzug y =    -89
loop[ 2]-time    :   262
Zuordnel: Ende   ; Zeit = 4926

Halbwert :  50;Hilfhalbwert :  25
    Vergroesserung :    5 ;ux :  5316754 ;uy :  4872823
 0 | sieben      : Zeit =   123
 0 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  2/ 0/ 0
 0 | cacoe2del   : Zeit =    36  ; Verzug x =    233   ; Verzug y =   -124
 1 | otsu1       : Zeit =    36   ; Schwelle/1000 =      100
 1 | selektieren : Zeit =     2, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  5316521 ; u2y =  4872947
 1 | corner_main : Zeit =    84
 1 | sieben      : Zeit =   123
 1 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del   : Zeit =    37  ; Verzug x =    231   ; Verzug y =   -123
loop[ 1]-time    :   289
 2 | otsu1       : Zeit =    37   ; Schwelle/1000 =      190
 2 | selektieren : Zeit =     1, aussortierte x/y/z =  0/ 0/ 0
 2 | u2x =  5316523 ; u2y =  4872946
 2 | corner_main : Zeit =    21
 2 | sieben      : Zeit =   123
 2 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del   : Zeit =    36  ; Verzug x =    231   ; Verzug y =   -123
loop[ 2]-time    :   226
Zuordnel: Ende   ; Zeit = 728

Halbwert :  40;Hilfhalbwert :  20
    Vergroesserung :    5 ;ux :  5316754 ;uy :  4872823
 0 | sieben      : Zeit =   122
 0 | x/y/z nicht gefunden =  2/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del   : Zeit =    36  ; Verzug x =    155   ; Verzug y =   -198
 1 | otsu1       : Zeit =    35   ; Schwelle/1000 =    37650
 1 | selektieren : Zeit =     2, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  5316599 ; u2y =  4873021
 1 | corner_main : Zeit =    44
 1 | sieben      : Zeit =   123
 1 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del   : Zeit =    37  ; Verzug x =    181   ; Verzug y =   -170
loop[ 1]-time    :   248
Zuordnel: Ende   ; Zeit = 459

Halbwert :  30;Hilfhalbwert :  15
    Vergroesserung :    5 ;ux :  5316754 ;uy :  4872823
 0 | sieben      : Zeit =   121
 0 | x/y/z nicht gefunden =  3/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del   : Zeit =    36  ; Verzug x =    229   ; Verzug y =   -133
 1 | otsu1       : Zeit =    36   ; Schwelle/1000 =      109
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  5316525 ; u2y =  4872956
 1 | corner_main : Zeit =    38
 1 | sieben      : Zeit =   123
 1 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del   : Zeit =    37  ; Verzug x =    237   ; Verzug y =   -132
loop[ 1]-time    :   273
 2 | otsu1       : Zeit =    36   ; Schwelle/1000 =      175
 2 | selektieren : Zeit =     2, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  5316517 ; u2y =  4872955
 2 | corner_main : Zeit =    19
```

```
 2 | sieben       : Zeit =   123
 2 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del    : Zeit =    36  ; Verzug x =    237   ; Verzug y =   -132
loop[ 2]-time     :   224
Zuordnel: Ende    ; Zeit = 709

Halbwert :  25;Hilfhalbwert :  12
    Vergroesserung :    5 ;ux :  5316754 ;uy :  4872823
 0 | sieben       : Zeit =   122
 0 | x/y/z nicht gefunden =  2/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del    : Zeit =    36  ; Verzug x =    229   ; Verzug y =   -134
 1 | otsul        : Zeit =    35   ; Schwelle/1000 =      512
 1 | selektieren : Zeit =     2, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  5316525 ; u2y =  4872957
 1 | corner_main : Zeit =    19
 1 | sieben       : Zeit =   122
 1 | x/y/z nicht gefunden =  2/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =    37  ; Verzug x =    241   ; Verzug y =   -134
loop[ 1]-time     :   222
Zuordnel: Ende    ; Zeit = 437

Halbwert :  20;Hilfhalbwert :  10
    Vergroesserung :    5 ;ux :  5316754 ;uy :  4872823
 0 | sieben       : Zeit =   121
 0 | x/y/z nicht gefunden =  3/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del    : Zeit =    36  ; Verzug x =    228   ; Verzug y =   -137
 1 | otsul        : Zeit =    36   ; Schwelle/1000 =      203
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  5316526 ; u2y =  4872960
 1 | corner_main : Zeit =    17
 1 | sieben       : Zeit =   121
 1 | x/y/z nicht gefunden =  3/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =    37  ; Verzug x =    239   ; Verzug y =   -136
loop[ 1]-time     :   248
Zuordnel: Ende    ; Zeit = 459
```

Untersuchtes Objekt : Kreisbögen

```
Halbwert :  70;Hilfhalbwert :  35
    Vergroesserung :   10 ;ux :  9681679 ;uy :  4993002
 0 | sieben       : Zeit =     1
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  1/ 1/ 0
 0 | cacoe2del    : Zeit =     3  ; Verzug x =   -149   ; Verzug y =     55
 1 | otsul        : Zeit =    33   ; Schwelle/1000 =       61
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 1/ 0
 1 | u2x =  9681828 ; u2y =  4992947
 1 | corner_main : Zeit =    77
 1 | sieben       : Zeit =     1
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =     3  ; Verzug x =   -174   ; Verzug y =     -1
loop[ 1]-time     :   122
 2 | otsul        : Zeit =    33   ; Schwelle/1000 =      690
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 1/ 0
 2 | u2x =  9681853 ; u2y =  4993003
 2 | corner_main : Zeit =    26
 2 | sieben       : Zeit =     1
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del    : Zeit =     3  ; Verzug x =   -173   ; Verzug y =      2
loop[ 2]-time     :    72
Zuordnel: Ende    ; Zeit = 376

Halbwert :  60;Hilfhalbwert :  30
    Vergroesserung :   10 ;ux :  9681679 ;uy :  4993002
```

```
 0 | sieben      : Zeit =     1
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  2/ 2/ 0
 0 | cacoe2del   : Zeit =     2  ; Verzug x =   -101   ; Verzug y =     47
 1 | otsu1       : Zeit =    33   ; Schwelle/1000 =        4
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 0/ 1
 1 | u2x =  9681780 ; u2y =  4992955
 1 | corner_main : Zeit =   103
 1 | sieben      : Zeit =     1
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del   : Zeit =     4  ; Verzug x =   -153   ; Verzug y =    -18
loop[ 1]-time    :   150
 2 | otsu1       : Zeit =    33   ; Schwelle/1000 =      610
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 1/ 0
 2 | u2x =  9681832 ; u2y =  4993020
 2 | corner_main : Zeit =    20
 2 | sieben      : Zeit =     1
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del   : Zeit =     4  ; Verzug x =   -150   ; Verzug y =    -14
loop[ 2]-time    :    68
Zuordnel: Ende   ; Zeit = 291

Halbwert :  50;Hilfhalbwert :  25
    Vergroesserung :   10 ;ux :  9681679 ;uy :  4993002
 0 | sieben      : Zeit =     2
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del   : Zeit =     3  ; Verzug x =   -352   ; Verzug y =     19
Zuordnel: Ende   ; Zeit = 22
```

Untersuchtes Objekt : Balken

```
Halbwert :  70;Hilfhalbwert :  35
    Vergroesserung :   10 ;ux :  9540373 ;uy :  5113793
 0 | sieben      : Zeit =     2
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 26/ 0/ 0
 0 | cacoe2del   : Zeit =     2  ; Verzug x =   -628   ; Verzug y =     18
 1 | otsu1       : Zeit =    33   ; Schwelle/1000 =     1238
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9541001 ; u2y =  5113775
 1 | corner_main : Zeit =   800
 1 | sieben      : Zeit =     4
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 10/ 0/ 0
 1 | cacoe2del   : Zeit =     5  ; Verzug x =    324   ; Verzug y =     11
loop[ 1]-time    :   851
 2 | otsu1       : Zeit =    33   ; Schwelle/1000 =     5463
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  9540049 ; u2y =  5113782
 2 | corner_main : Zeit =   292
 2 | sieben      : Zeit =     5
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del   : Zeit =     7  ; Verzug x =    255   ; Verzug y =    -11
loop[ 2]-time    :   343
 3 | otsu1       : Zeit =    34   ; Schwelle/1000 =     4763
 3 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 3 | u2x =  9540118 ; u2y =  5113804
 3 | corner_main : Zeit =    29
 3 | sieben      : Zeit =     5
 3 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 3 | cacoe2del   : Zeit =     7  ; Verzug x =    327   ; Verzug y =    -10
loop[ 3]-time    :    82
Zuordnel: Ende   ; Zeit = 1454

Halbwert :  60;Hilfhalbwert :  30
    Vergroesserung :   10 ;ux :  9540373 ;uy :  5113793
```

```
 0 | sieben       : Zeit =     3
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 21/ 0/ 0
 0 | cacoe2del    : Zeit =     3  ; Verzug x =   -113   ; Verzug y =     -1
 1 | otsu1        : Zeit =    33   ; Schwelle/1000 =     8563
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9540486 ; u2y =  5113794
 1 | corner_main : Zeit =   446
 1 | sieben       : Zeit =     5
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =     7  ; Verzug x =      5   ; Verzug y =      0
loop[ 1]-time     :   497
 2 | otsu1        : Zeit =    33   ; Schwelle/1000 =     5804
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  9540368 ; u2y =  5113793
 2 | corner_main : Zeit =    21
 2 | sieben       : Zeit =     4
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del    : Zeit =     7  ; Verzug x =    -71   ; Verzug y =      0
loop[ 2]-time     :    73
Zuordnel: Ende    ; Zeit = 638

Halbwert :  50;Hilfhalbwert :  25
    Vergroesserung :   10 ;ux :  9540373 ;uy :  5113793
 0 | sieben       : Zeit =     3
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 20/ 0/ 0
 0 | cacoe2del    : Zeit =     3  ; Verzug x =    -49   ; Verzug y =      0
 1 | otsu1        : Zeit =    33   ; Schwelle/1000 =     7708
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9540422 ; u2y =  5113793
 1 | corner_main : Zeit =   315
 1 | sieben       : Zeit =     5
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =     7  ; Verzug x =    -80   ; Verzug y =     -6
loop[ 1]-time     :   367
 2 | otsu1        : Zeit =    33   ; Schwelle/1000 =     6459
 2 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  9540453 ; u2y =  5113799
 2 | corner_main : Zeit =    16
 2 | sieben       : Zeit =     4
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del    : Zeit =     6  ; Verzug x =   -148   ; Verzug y =     -8
loop[ 2]-time     :    67
Zuordnel: Ende    ; Zeit = 507

Halbwert :  40;Hilfhalbwert :  20
    Vergroesserung :   10 ;ux :  9540373 ;uy :  5113793
 0 | sieben       : Zeit =     4
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del    : Zeit =     7  ; Verzug x =    -41   ; Verzug y =     -6
Zuordnel: Ende    ; Zeit = 28

Halbwert :  32;Hilfhalbwert :  16
    Vergroesserung :   10 ;ux :  9540373 ;uy :  5113793
 0 | sieben       : Zeit =     5
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del    : Zeit =     6  ; Verzug x =    -38   ; Verzug y =    -12
Zuordnel: Ende    ; Zeit = 32
```

Untersuchtes Objekt : Balken 2

```
Halbwert :  70;Hilfhalbwert :  35
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben       : Zeit =     3
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 21/ 0/ 0
 0 | cacoe2del    : Zeit =     3  ; Verzug x =  -1460   ; Verzug y =   -117
 1 | otsu1        : Zeit =    33   ; Schwelle/1000 =     3923
 1 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9497694 ; u2y =  5066258
 1 | corner_main : Zeit =   742
 1 | sieben       : Zeit =     3
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 14/ 0/ 0
 1 | cacoe2del    : Zeit =     4  ; Verzug x =   -886   ; Verzug y =    -47
loop[ 1]-time    :   790
 2 | otsu1        : Zeit =    33   ; Schwelle/1000 =     4151
 2 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  9497120 ; u2y =  5066188
 2 | corner_main : Zeit =   507
 2 | sieben       : Zeit =     4
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  6/ 0/ 0
 2 | cacoe2del    : Zeit =     6  ; Verzug x =   -345   ; Verzug y =    -35
loop[ 2]-time    :   557
 3 | otsu1        : Zeit =    34   ; Schwelle/1000 =     4674
 3 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 3 | u2x =  9496579 ; u2y =  5066176
 3 | corner_main : Zeit =   216
 3 | sieben       : Zeit =     4
 3 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 3 | cacoe2del    : Zeit =     6  ; Verzug x =    -38   ; Verzug y =    -45
loop[ 3]-time    :   267
 4 | otsu1        : Zeit =    34   ; Schwelle/1000 =     4661
 4 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 4 | u2x =  9496272 ; u2y =  5066186
 4 | corner_main : Zeit =    30
 4 | sieben       : Zeit =     4
 4 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 4 | cacoe2del    : Zeit =     7  ; Verzug x =     72   ; Verzug y =    -41
loop[ 4]-time    :    86
Zuordnel: Ende   ; Zeit = 1767

Halbwert :  60;Hilfhalbwert :  30
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben       : Zeit =     2
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 25/ 0/ 0
 0 | cacoe2del    : Zeit =     2  ; Verzug x =  -2085   ; Verzug y =    485
 1 | otsu1        : Zeit =    33   ; Schwelle/1000 =      490
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9498319 ; u2y =  5065656
 1 | corner_main : Zeit =   645
 1 | sieben       : Zeit =     5
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =     6  ; Verzug x =   -839   ; Verzug y =   -447
loop[ 1]-time    :   694
 2 | otsu1        : Zeit =    34   ; Schwelle/1000 =     3160
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  9497073 ; u2y =  5066588
 2 | corner_main : Zeit =    23
 2 | sieben       : Zeit =     4
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del    : Zeit =     7  ; Verzug x =   -784   ; Verzug y =   -484
loop[ 2]-time    :    75
Zuordnel: Ende   ; Zeit = 818
```

```
Halbwert :  50;Hilfhalbwert :  25
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben       : Zeit =     3
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 20/ 0/ 0
 0 | cacoe2del    : Zeit =     3  ; Verzug x =   -919   ; Verzug y =    -68
 1 | otsu1        : Zeit =    33   ; Schwelle/1000 =     9116
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9497153 ; u2y =  5066209
 1 | corner_main : Zeit =   371
 1 | sieben       : Zeit =     4
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del    : Zeit =     6  ; Verzug x =    186   ; Verzug y =     -3
loop[ 1]-time     :   422
 2 | otsu1        : Zeit =    33   ; Schwelle/1000 =     8330
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  9496048 ; u2y =  5066144
 2 | corner_main : Zeit =    17
 2 | sieben       : Zeit =     4
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del    : Zeit =     6  ; Verzug x =    287   ; Verzug y =     -3
loop[ 2]-time     :    68
Zuordnel: Ende    ; Zeit = 554

Halbwert :  32;Hilfhalbwert :  16
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben       : Zeit =     4
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del    : Zeit =     6  ; Verzug x =    -10   ; Verzug y =    -35
Zuordnel: Ende    ; Zeit = 25
```

Untersuchtes Objekt : Balken 2 (nach unten und nach links verschoben)

```
Halbwert :  80;Hilfhalbwert :  40
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben       : Zeit =     2
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 23/ 0/ 0
 0 | cacoe2del    : Zeit =     3  ; Verzug x =  -1085   ; Verzug y =   -676
 1 | otsu1        : Zeit =    33   ; Schwelle/1000 =     2545
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9497319 ; u2y =  5066817
 1 | corner_main : Zeit =  1041
 1 | sieben       : Zeit =     3
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 16/ 0/ 0
 1 | cacoe2del    : Zeit =     4  ; Verzug x =   -412   ; Verzug y =   -224
loop[ 1]-time     :  1090
 2 | otsu1        : Zeit =    34   ; Schwelle/1000 =     6206
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 0/ 0
 2 | u2x =  9496646 ; u2y =  5066365
 2 | corner_main : Zeit =   670
 2 | sieben       : Zeit =     4
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  8/ 0/ 0
 2 | cacoe2del    : Zeit =     5  ; Verzug x =   -134   ; Verzug y =   -198
loop[ 2]-time     :   720
 3 | otsu1        : Zeit =    34   ; Schwelle/1000 =     5590
 3 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 3 | u2x =  9496368 ; u2y =  5066339
 3 | corner_main : Zeit =   374
 3 | sieben       : Zeit =     4
 3 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 3 | cacoe2del    : Zeit =     7  ; Verzug x =     77   ; Verzug y =   -162
loop[ 3]-time     :   427
 4 | otsu1        : Zeit =    33   ; Schwelle/1000 =     4738
 4 | selektieren : Zeit =     1, aussortierte x/y/z =  1/ 0/ 0
 4 | u2x =  9496157 ; u2y =  5066303
```

```
 4 | corner_main : Zeit =    43
 4 | sieben      : Zeit =     4
 4 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 4 | cacoe2del   : Zeit =     7  ; Verzug x =    174   ; Verzug y =   -102
loop[ 4]-time    :    98
Zuordnel: Ende   ; Zeit = 2397

Halbwert :  70;Hilfhalbwert :  35
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben      : Zeit =     2
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 25/ 0/ 0
 0 | cacoe2del   : Zeit =     2  ; Verzug x =   -739   ; Verzug y =   -614
 1 | otsu1       : Zeit =    33   ; Schwelle/1000 =     1290
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9496973 ; u2y =  5066755
 1 | corner_main : Zeit =   863
 1 | sieben      : Zeit =     4
 1 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig = 10/ 0/ 0
 1 | cacoe2del   : Zeit =     4  ; Verzug x =   -143   ; Verzug y =     18
loop[ 1]-time    :   911
 2 | otsu1       : Zeit =    33   ; Schwelle/1000 =     2914
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  9496377 ; u2y =  5066123
 2 | corner_main : Zeit =   393
 2 | sieben      : Zeit =     4
 2 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del   : Zeit =     6  ; Verzug x =    -25   ; Verzug y =   -140
loop[ 2]-time    :   444
 3 | otsu1       : Zeit =    34   ; Schwelle/1000 =     3367
 3 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 3 | u2x =  9496259 ; u2y =  5066281
 3 | corner_main : Zeit =    62
 3 | sieben      : Zeit =     4
 3 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 3 | cacoe2del   : Zeit =     6  ; Verzug x =     70   ; Verzug y =   -131
loop[ 3]-time    :   115
Zuordnel: Ende   ; Zeit = 1523

Halbwert :  60;Hilfhalbwert :  30
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben      : Zeit =     2
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 23/ 0/ 0
 0 | cacoe2del   : Zeit =     3  ; Verzug x =    296   ; Verzug y =   -362
 1 | otsu1       : Zeit =    33   ; Schwelle/1000 =     6921
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9495938 ; u2y =  5066503
 1 | corner_main : Zeit =   593
 1 | sieben      : Zeit =     4
 1 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del   : Zeit =     6  ; Verzug x =    -39   ; Verzug y =    -58
loop[ 1]-time    :   644
 2 | otsu1       : Zeit =    33   ; Schwelle/1000 =     6252
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 2 | u2x =  9496273 ; u2y =  5066199
 2 | corner_main : Zeit =    45
 2 | sieben      : Zeit =     4
 2 | x/y/z nicht gefunden =  1/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del   : Zeit =     6  ; Verzug x =     28   ; Verzug y =    -54
loop[ 2]-time    :    95
Zuordnel: Ende   ; Zeit = 794

Halbwert :  50;Hilfhalbwert :  25
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben      : Zeit =     3
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig = 17/ 0/ 0
```

```
 0 | cacoe2del   : Zeit =     3  ; Verzug x =    307   ; Verzug y =   -327
 1 | otsu1       : Zeit =    33   ; Schwelle/1000 =     8278
 1 | selektieren : Zeit =     0, aussortierte x/y/z =  1/ 0/ 0
 1 | u2x =  9495927 ; u2y =  5066468
 1 | corner_main : Zeit =   311
 1 | sieben      : Zeit =     4
 1 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 1 | cacoe2del   : Zeit =     6  ; Verzug x =    -69   ; Verzug y =   -148
loop[ 1]-time    :   362
 2 | otsu1       : Zeit =    33   ; Schwelle/1000 =     7238
 2 | selektieren : Zeit =     0, aussortierte x/y/z =  0/ 0/ 0
 2 | u2x =  9496303 ; u2y =  5066289
 2 | corner_main : Zeit =     0
 2 | sieben      : Zeit =     5
 2 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 2 | cacoe2del   : Zeit =     7  ; Verzug x =    -69   ; Verzug y =   -148
loop[ 2]-time    :    52
Zuordnel: Ende   ; Zeit = 470

Halbwert :  32;Hilfhalbwert :  16
    Vergroesserung :   10 ;ux :  9496234 ;uy :  5066141
 0 | sieben      : Zeit =     4
 0 | x/y/z nicht gefunden =  0/ 0/ 0 ; mehrdeutig =  0/ 0/ 0
 0 | cacoe2del   : Zeit =     6  ; Verzug x =   -150   ; Verzug y =   -313
Zuordnel: Ende   ; Zeit = 24
```

LITERATUR

[3L88] 3L Ltd:"Parallel C User Guide"; 3L Ltd, Peel House, Ladywell, Livingston EH54 6AG, Scotland;1988.

[Beck85] E. W. Becker, W. Ehrfeld, P. Hagmann, A. Maner, D. Münchmeyer: "Herstellung von Mikrostrukturen mit großem Aspektverhältnis und großer Strukturhöhe durch Röntgentiefenlithographie mit Synchrotronstrahlung, Galvanoformung und Kunstoffabformung (LIGA-Verfahren)." KfK-Bericht 3995, Kernforschungszentrum Karlsruhe 1985.

[Bürg87a] B. Bürg, B. Deiß, H. Guth, U. Klein, D. Schmidt. Unveröffentlichter Bericht, Kernforschungszentrum Karlsruhe (1987).

[Bürg87b] B. Bürg, H.Guth, A. Hellmann. Unveröffentlichter Bericht, Kernforschungszentrum Karlsruhe (1987).

[Bürg89a] B. Bürg, H. Guth, A. Hellmann. Unveröffentlichter Bericht, Kernforschungszentrum Karlsruhe (1989).

[Bürg89b] Bernhard Bürg, Helmut Guth, Andreas Hellmann: "Bildanalytische Qualitätskontrolle in der Mikrofertigung: Ein vollautomatisches, hochflexibles und schnelles Strukturmeßsystem."; *Mustererkennung 1989: 11. DAGM-Symposium Hamburg, Oktober 1989, Proceedings* Seite 168-172. Springer-Verlag Berlin Heidelberg; 1989; ISBN 3_540_51748_0.

[Doms81] Wolfgang Domschke: "Logistik: Transport" Band 1; R. Oldenbourg Verlag Gmbh, München; 1981; ISBN 3_486_24791_3.

[Doms82] Wolfgang Domschke: "Logistik: Rundreisen und Touren" Band 2; R. Oldenbourg Verlag Gmbh, München; 1982; ISBN 3_486_24991_6.

[Doms84] Wolfgang Domschke: "Logistik: Standorte" Band 1; R. Oldenbourg Verlag Gmbh, München; 1981; ISBN 3_486_25651_3.

[Düpm87] C.Düpmeier, H. Guth. Unveröffentlichter Bericht, Kernforschungszentrum Karlsruhe (1987).

[Gal87a] Tomas Gal, Hrsg.; "Grundlagen des Operations Research" Band 1; Springer Verlag Berlin Heidelberg; 1987; ISBN 3_540_17276_9.

[Gal87b] Tomas Gal, Hrsg.; "Grundlagen des Operations Research" Band 2; Springer Verlag Berlin Heidelberg; 1987; ISBN 3_540_17275_0.

[Gal87c] Tomas Gal, Hrsg.; "Grundlagen des Operations Research" Band 3; Springer Verlag Berlin Heidelberg; 1987; ISBN 3_540_17530_X.

[Guth89] Helmut Guth: "CACOE2DEL: Eine Routine zur Berechnung der Koeffizienten einer 2-Dimensionen Transformationsmatrix"; private Mitteilung 1989.

[Neum75a] Klaus Neumann: "Operations Research Verfahren" Band 1; Carl Hanser Verlag, München / Wien; 1975; ISBN 3_446_11958_2.

[Neum75b] Klaus Neumann: "Operations Research Verfahren" Band 3; Carl Hanser Verlag, München / Wien; 1975; ISBN 3_446_11992_2.

[Neum77] Klaus Neumann: "Operations Research Verfahren" Band 2; Carl Hanser Verlag, München / Wien; 1977; ISBN 3_446_11991_4.

[Otsu79] Nobuyuki Otsu: "A Treshold Selection Method from Gray-Level Histograms"; *IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-9,* No 1, January 1979;