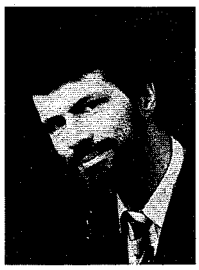
Fehlertoleranz in intelligenten Benutzerschnittstellen

Fault Tolerance in Intelligent User Interfaces

Rüdiger Brause, Johann-Wolfgang-Goethe-Universität Frankfurt a. M.



Dr. Rüdiger Brause studierte Physik und Mathematik in Saarbrücken und Tübingen und beendete sein Studium mit der Promotion in Tübingen 1983 über verteilte, fehlertolerante Systeme. Seit 1985 ist er als Akad. Rat an der Universität Frankfurt tätig und beschäftigt sich mit parallelen, fehlertoleranten Multi-Mikroprozessorsystemen und Neuronalen Netzen. Dr. Brause ist Mitglied der ACM, GI und INNS.

Dieser Beitrag führt Schwierigkeiten im Umgang mit Computern auf mangelnde Fehlertoleranz der Benutzerschnittstelle zurück. Es wird gezeigt, daß diese Probleme in den Bereichen Bilderkennung, Spracherkennung und Robotik der Künstlichen Intelligenz ebenfalls vorhanden sind und sich auf drei grundsätzliche Problemkreise zurückführen lassen. Gute Lösungsansätze existieren auf dem Gebiet der Neuronalen Netze, die allerdings noch nicht konsistent und vollständig sind.

This paper views problems of computer users as lacking fault tolerance in the user interface. It is shown, that these problems are very similar to those encountered in artificial intelligence, especially in computer vision, speech recognition and robotics. The difficulties can be reduced to three different kinds of basic problems. The research of Neural Networks proposes interesting solutions, but they are neither consistent nor complete until now.

1. Einleitung und Problemstellung

Die rasante Entwicklung des Preis-Leistungsverhältnisses der Mikroprozessoren in den letzten Jahren machen Anwendungen möglich, die vorher für Computer undenkbar oder unrentabel erschienen.

Dabei zeigte sich aber auch gleichzeitig, daß bestimmte Anwendungen (Robotik, visuelle Qualitätskontrolle, automatische Sprachübersetzungen, etc.) mit den gängigen von-Neumann Maschinen und den traditionellen Algorithmen sehr schwierig zu bearbeiten waren. Erst der massive Einsatz paralleler Informationsverarbeitung, realisiert durch parallel arbeitende Algorithmen auf parallelen Maschinen, verspricht in naher Zukunft bessere Ergebnisse. Trotz der Erfolge, die die anvisierten wissensbasierten, parallelen Systeme bringen werden, sind einige Probleme weitgehend ungelöst.

Eingabe-Fehlertoleranz und menschengerechte Benutzerschnittstellen

Die Integration der Computer in der Verwaltung, in Fabriken und Banken ist weniger ein Problem von leistungsfähigen Algorithmen, Datenbanken oder Maschinen, sondern das Problem, Menschen ohne „Computerverständnis" den Umgang mit den automatischen Daten- und Fabrikationssystemen zu ermöglichen. Das Grundproblem besteht darin, wie man „den Computer menschengerecht" machen kann, wie also die Benutzerschnittstelle der automatischen Systeme den Fähigkeiten und Gewohnheiten der Menschen angepaßt werden kann.

Betrachten wir die gewohnten menschlichen Informationskanäle, wie Bilder (z. B. bei Gesten) und Schall (Sprache, Laute), so bemerken wir, daß das reiche Spektrum menschlicher Ausdrucksfähigkeit geprägt ist durch Ergänzungen im Kontext. Sei es, daß wir durch nonverbale Gestik oder durch verbale Kurzsätze unseren Willen kundtun; stets muß der Zuhörer Informationen aus dem sprachlichen oder inhaltlichen Kontext ergänzen sowie falsche und unpassende Formen korrigieren, um eine sinnvolle Aussage oder eine sinnvolle Handlung zu erkennen. Diese menschliche Fähigkeit beim Zuhörer (Empfänger), fehlertolerant Informationen zu abstrahieren, ermöglicht es wiederum dem Sprecher (Sender), sich mehrdeutig, fehlerhaft und unvollständig zu artikulieren und trotzdem noch verstanden zu werden.

So menschengerecht und problemlos diese Kommunikationsgewohnheiten bei Menschen sind, so problematisch ist dies bei Computersystemen. Aus frustrierenden Erfahrungen mit falschen oder fehlenden Computeraktionen heraus werden heutzutage deshalb nur eindeutige, logisch konsistente Befehle und Anfragen verarbeitet. Dazu muß der Benutzer eine spezielle, manchmal „benutzerfreundlich" genannte Sprache lernen, in der der mit dem System kommunizieren darf. Im Prinzip paßt sich damit wieder der Benutzer im Denken dem Computer an, nicht umgekehrt.

Benötigt wird also eine fehlertolerante bzw. fehlervermeidende Benutzerschnittstelle, die nicht nur formale (z. B. syntaktische) Fehler erkennt und korrigiert, sondern auch inhaltlich aus dem Kontext erkennt, was der Benutzer meint.

Fehlende Programmierung

Die Forderung nach einer fehlertoleranten Benutzerschnittstelle hat allerdings ein Problem: jeder Benutzer hat einen anderen Kontext, beispielsweise die Erfahrungen von mehreren Konsultationen des Computersystems. Eine gute Benutzerschnittstelle müßte also die Eigenheiten des Benutzers und seine Erfahrungen speichern, allerdings auf der abstrakten Ebene von Gewohnheiten. Beim Benutzen wird dabei die Schnittstelle klüger, sie *lernt* aus den Daten. Da für gute Benutzerschnittstellen bereits Expertensysteme eingesetzt werden, zielen die Forderungen nach Lernen dabei auf ein Grundproblem aller existierenden Expertensysteme. Im Unterschied zum menschlichen Experten, der beim Erstellen seiner Expertisen auch lernt (z. B. aus seinen Fehlern), müssen Expertensysteme immer wieder aktualisiert und von menschlichen Experten an das menschliche Wissen angepaßt werden. Normale menschliche Erfahrungen, aus denen menschliche Experten im Zweifelsfall ihre Analogien herleiten, sind ihnen prinzipiell verschlossen. Betrachten wir den Aufbau und die Pflege der Wissensbank eines Expertensystems als eine der effektivsten Formen der Programmierung heutzutage, so läßt sich das derzeitige Problem von fehlenden Programmierern in Industrie und Wirtschaft als typisch für die heutige Rechnerarchitektur betrachten: Sie ist nicht selbstprogrammierend bzw. selbstlernend. Gesucht ist eine Soft- und Hardwarearchitektur, die komplexe Fähigkeiten selbst lernen kann.

Hardware-Fehlertoleranz

Bei dem massiven parallelen Einsatz von Hardware steigt auch die Ausfallwahrscheinlichkeit des Gesamtsystems. Dabei stellen sich folgende Fragen:

Wie läßt sich erkennen, daß eine Einheit ausgefallen ist?

Wie wird Test, Diagnose und Reparatur (Rekonfiguration) durchgeführt?

Wie werden die durch Hardwaredefekte verfälschten Daten wieder restauriert?

Besonders die letzte Frage ist ziemlich heikel. Verwendet man ein Mehrheitsvotum von Einheiten (maskierende Fehlerkorrektur), um umständliches Sichern von Zwischenergebnissen zu vermeiden (Rollback), so ist der Ausfall einzelner, votierender Einheiten beim Ergebnis nicht festzustellen. Dies kompliziert die Diagnosesituation. Auch zusätzlich eingebaute Testsignalpfade sagen nur etwas über Defekte in diesen Pfaden aus, nicht aber über den maskierten Datenpfad. Verwendet man andererseits keine Fehlererkennung, so ist es nur eine Frage der Zeit, bis die Mehrheit der votierenden Einheiten defekt ist.

Im Gegensatz dazu bereitet es uns Menschen keinerlei Probleme, daß in unserem relativ langsam und asynchron arbeitenden Gehirn (Schaltfrequenzen um den Faktor 10^4 geringer als beim Mikroprozessor) laufend informationsverarbeitende Nervenzellen bei der Arbeit wegsterben, ohne ersetzt zu werden. Da diese Eigenschaften ohne die Parallelität der Aktivität der Nervenzellen undenkbar wäre, sind die Gehirntheoretiker – im Unterschied zu den Informatikern – schon seit Jahrzehnten gezwungen, Funktionsmodelle paralleler, fehlertoleranter Systeme zu entwickeln.

Bei den parallelen Funktionsmodellen lassen sich zwei verschiedene Ansätze unterscheiden: der „lokalistische" und der „distributionistische" Ansatz.

Die dedizierten oder *lokalistischen* Modelle erklären die Gesamtfunktion des Systems aus der Verbindung von Einheiten; jeder Verbindung korrespondiert dabei eine Einzelfunktion [FELD].

Die Verbindung von genau definierten Einzelfunktionen zu einer Gesamtfunktion liegt im wesentlichen den schnellen Maschinen der AI [HILL] und den verteilten Algorithmen [SHA] zu Grunde.

Allerdings sind die Modelle dieses Ansatzes nicht besonders fehlertolerant: Fällt eine Einheit oder Verbindung aus, so ist damit auch eine Relation oder ein Objekt „vergessen".

Diese Art von Netzwerken ist zwar geeignet, bestimmte Probleme schnell zu lösen, erlaubt aber Fehlertoleranz auf Hardware- und Softwareebene nur mit zusätzlichem, nicht unerheblichem Aufwand.

In nicht-zentralisierten, verteilten Systemen ist ein Overhead zur Verwaltung der Redundanz (Systemzustandstafeln), Fehlererkennung, -diagnose, und Rekonfiguration nötig [KUHL]. Allein das Problem der Erkennung fehlerhafter Einheiten (Diagnose) ist bei funktions-verfälschenden Einheiten nicht einfach zu lösen [PEAR]. Das lokalistische konnektionistische Modell entspricht damit dem heutigen Ansatz, möglichst viele von-Neumann Maschinen zur parallelen Informationsverarbeitung in einem Netz zusammen zu schließen.

Leistungsverluste durch Hard- und Softwarezusätze für den Netzbetrieb (Netzwerkprotokolle) werden dabei in Kauf genommen und versucht, sie durch günstige funktionelle Partitionierung des Problems zu minimieren.

Anders dagegen der Ansatz der *distributiven* Modelle. Sie repräsentieren die Aktivitätsmuster nicht lokal in einer Verbindung, sondern als gemeinsame Aktivität aller Verbindungen. Die Gesamtheit aller Verbindungen läßt sich mittels einer Verbindungsmatrix beschreiben, so daß die Gesamtfunktion als Wechselwirkung aller Neuronen untereinander mit einer gewichteten Verbindungs-

matrix modelliert wird [KOH1]. Aus der Holografie ist bekannt, daß die Beschädigung von Teilen der Bilderfolien bei der Reproduktion des Bildes nur die Qualität des Gesamtbildes herabsetzt, nicht aber Einzelbereiche verschwinden läßt. Dies ergibt sich aus der Tatsache, daß die Bildinformationen nicht lokalisiert, sondern prinzipiell an jedem Punkt der Bildfolie vorhanden sind. Da die Matrix-Modelle ähnliche Proportionen aufweisen, wurden sie in Analogie zum physikalischen Pendant anfangs „Hololologische Modelle" genannt [LONG], [WIL].

2. Fehlertolerante Computer und die Probleme der Künstlichen Intelligenz

Wieweit lassen sich nun mit den geschilderten konnektivistischen Ansätzen **(Neuro-Computer)** die erwähnten Probleme lösen?

Zweifelsohne kann ein neuer Ansatz nicht sofort alle Anwendungsprobleme lösen, vielmehr muß man auch bei einem erfolgversprechenden Ansatz noch viel Entwicklungsarbeit leisten, um alle neuen Probleme zu lösen, die bei diesem speziellen Ansatz auftreten. Beispielsweise bedeutet ein Schritt zu menschenähnlicheren Computern ja auch, die Fehler zuzulassen, die Menschen normalerweise machen.

Trotzdem gibt es interessante Ansätze konnektivistische Modelle in den Gebieten Bilderkennung, Sprachverarbeitung und Robotik der künstlichen Intelligenz einzusetzen. Dabei ist die Eigenschaft *Fehlertoleranz* untrennbar verbunden mit den grundlegenden Funktionen dieser Modelle; *Datenkorrektur, Abstraktion, Mustererkennung* und *Kategorisierung* sind inhärente Fehlertoleranzmechanismen. Damit überdeckt sich die Untersuchung der Fehlertoleranzeigenschaften der konnektivistischen Modelle weitgehend mit der Untersuchung ihrer Grundfunktionen.

Betrachten wir nun die Probleme der drei genannten Gebiete genauer.

Bildverarbeitung

Abgesehen von der Forderung, möglichst alle Punkte eines Bildes (Pixel) gleichzeitig zu bearbeiten, ist die Bildverarbeitung üblicherweise in verschiedene Stufen oder Schichten aufgeteilt; siehe Bild 1.

Lassen sich Bilderfassung und -verbesserung noch mit klaren, nachrichtentechnischen Algorithmen durchführen, so gibt schon die Erkennung eines Umrisses **(Segmentierung)** Probleme auf. Viele Umrisse sind durch Störungen und Verdekkungen nicht vollständig. Eine der besten Techniken, um kleine Lücken zu schließen, besteht in der Ergänzung dieser Stellen mittels der Information von Nachbarpixeln **(Relaxation).** Ist dagegen eine

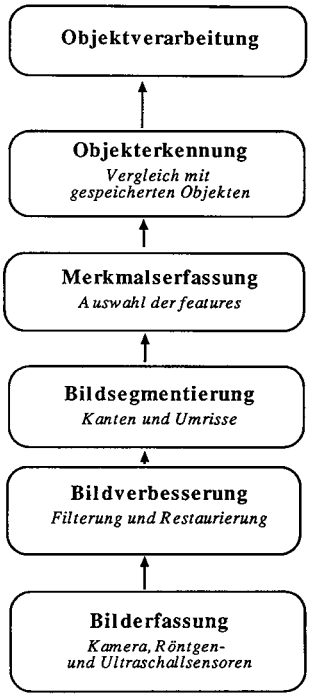


Bild 1. Schichten der Bildverarbeitung.

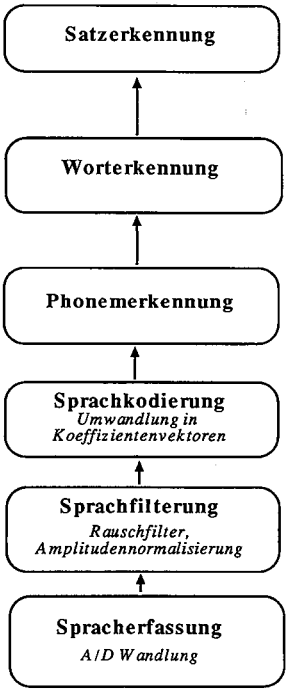


Bild 2. Schichten der Sprachverarbeitung.

starke Oberflächentextur vorhanden, so versagen auch diese Methoden und es müssen, am besten mittels **stochastischer Mustererkennung**sverfahren, die Flächen mit gleicher Textur erkannt und abgegrenzt werden [BALL].

Auch beim Erkennen von Gesamtobjekten und Situationen, die im allgemeinen Fall mit wissensbasierten Methoden (semantischen Netzen u.ä.) behandelt werden, kämpft die Objekterkennung bei falscher Identifizierung von Teilobjekten (Grund: Störungen auf unterer Ebene) und unvollständigen Objekten mit dem gleichen Problem: Bildmuster, die den gespeicherten Referenzmustern ähnlich, aber nicht gleich sind, müssen dem ähnlichsten Referenzmuster zugeordnet werden.

Dies gilt übrigens auch bei der Erkennung von zeitlichen Sequenzen von Bildern (Bildfolgen): Menschen und Gegenstände, die sich bei der Bewegung leicht ändern, müssen trotz Variation und Verschiebung auf jedem Bild der Folge wiedererkannt werden.

Spracherkennung

Auch in der Spracherkennung haben sich die mehrstufigen Verfahren und Modelle ebenso wie in der Experimentalpsychologie durchgesetzt. Ein solches Schichtenmodell ist in Bild 2 gezeigt.

Reichte die Methode, eine Lautcharakteristik (zeitliche, endliche Folge von Amplituden- und Frequenzwerten) mittels Durchsuchen einer Liste von bekannten Lauten und Identifizierung mit einem Ähnlichkeitskriterium **(Dynamic Time Warping)** noch aus, Ein-Wort-Systeme mit einem

Wortschatz von ca. 100 Worten zu bauen, so stößt diese Ein-Schritt-Methode bei Sätzen schon an ihre Grenzen: In normalen Texten gibt es keine zwei gleichen Sätze, da die Kombinationsmöglichkeiten von Einzelworten zu groß sind. Gute mehrstufige Systeme versuchen, die regionalen und individuellen Sprachunterschiede auf Laut-, Wort- und Satzebene durch Systeme mit Wahrscheinlichkeitsaussagen zu modellieren **(Hidden Marcov Models)** und diejenigen Übergänge zwischen Spracheinheiten (Laute, Worte) zu finden, die am wahrscheinlichsten mit den unbekannten Lautsequenzen übereinstimmen [DE MORI].

Die Probleme, die diese Ansätze mit sich bringen, ähneln sehr denen der Bildverarbeitung:

- Es müssen Listen von gespeicherten, ähnlichen Worten durchsucht werden, um das Passende zu finden
- Variationen bzw. Fehler und fehlende Laute können zu Fehlern führen; beim Schichtenmodell der *hidden marcov models* müssen die Wahrscheinlichkeiten auf verschiedenen Ebenen gleichzeitig berücksichtigt werden, um das im Kontext Passende zu finden. Dabei sind aber auch die Übergangswahrscheinlichkeiten der Lautmuster individuell vorherzubestimmen (starker Rechenaufwand).

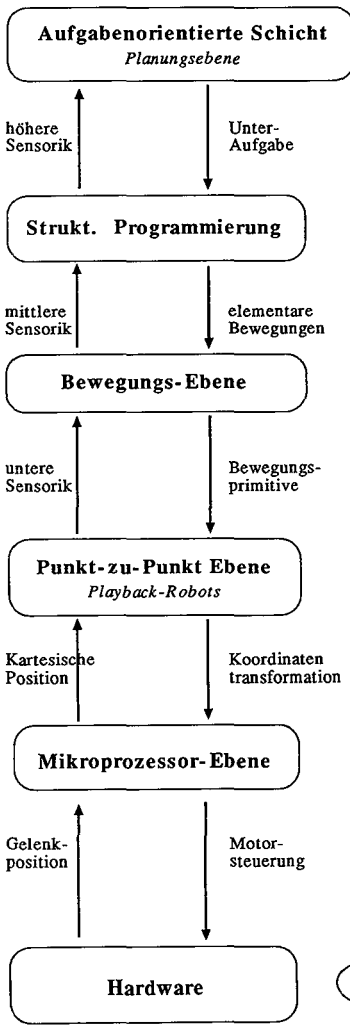


Bild 3. Roboter-Schichten.

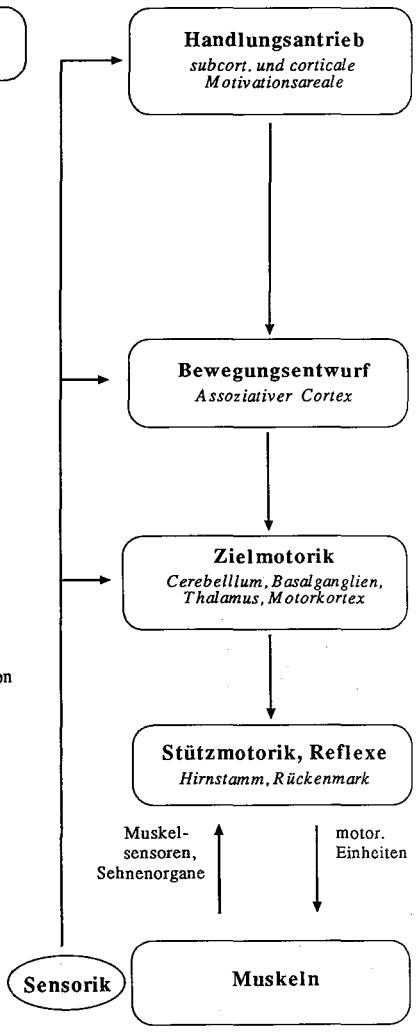


Bild 4. Schichtenmodell menschlicher Motorik.

Betrachtet man im Gegensatz dazu das menschliche Vermögen, Sprache zu erkennen, so zeigen sich die grundlegenden Unterschiede zwischen beiden Ansätzen. Schon auf Lautebene **(Phoneme)** registriert jeder Computer ein Kontinuum an physikalischen Lautformen; die menschliche Sprachverarbeitung aber läßt uns alle kleineren Lautvariationen als ein- und denselben Laut hören, sofern eine bestimmte Variationsgrenze nicht überschritten wird **(Kategoriale Sprachwahrnehmung).** Diese Kategorisierung ist eine Musterkennung und setzt sich auf höheren Stufen fort, so daß die Fehlertoleranz bezüglich der Eingabedaten als wichtiges Funktionsprinzip unserer Sprachwahrnehmung auf allen Ebenen realisiert sein muß.

Umgekehrt ergibt sich daraus aber auch der große Erfolg beim Computereinsatz bei der Sprecheridentifizierung (z. B. Erpresserstimmen am Telefon), wo Menschen ziemlich versagen, da sie nicht exakt die Lautform, sondern nur die Kategorien bewußt wahrnehmen.

Robotik

In der Robotik gelten die gleichen, vorher erläuterten Probleme bei der Sensorik (Bilderkennung, Spracherkennung, Taktile Sensoren). Zusätzlich sind hier Aktivierungssignale nötig, um die Gliedmaßen zu steuern. Diese Steuerung hat einige Probleme zu lösen. Um ein bestimmtes Ziel mit dem Greifer zu erreichen, müssen alle Segmente und Gelenke des Roboterarmes gleichzeitig mit der richtigen Beschleunigung und Geschwindigkeit um den richtigen Betrag bewegt werden.

Dies erfordert

- Umrechnung der Gesamtbewegung in Einzelbewegungen innerhalb der gelenkspezifischen Koordinatensysteme **(Inverse homogene Transformationen).** Dabei können Probleme auftreten (Singularitäten).
- Einbeziehung der Massen und Bewegungsenergien der Einzelsegmente und der aufgenommenen Last in die Bewegungsgleichungen der Gelenksteuerungen.

Dazu sind sowohl Lösungen von Matrizengleichungen als auch Differenzialgleichungen zweiter Ordnung nötig. Werden zusätzliche Forderungen an die Bewegung gestellt *(Konstante Kraft; gleichförmige, lineare Bewegung im Kartesischen Raum),* so müssen außerdem die Daten der Positions- und Kraftsensoren ausgewertet werden. Die bei der ungenauen Bewegung (Schlupf der Krafttransmission, Durchbiegen des Arms bei schweren Lasten, Bewegungshindernis) nötige Fehlertoleranz wird meist durch eine Kombination von Rückkopplung der Sensorsignale und einer Liste von Ausnahme-

situationen („Werkstück heruntergefallen") verwirklicht.

Probleme macht bei diesem Ansatz der Bewegungssteuerung neben der Berechnung diverser Gleichungen in Real-Time die Programmierung dieser Bewegungen. Die gewünschte Bewegungssequenz wird entweder in einer Programmiersprache eingegeben (z.B. moveTo(x,y,z)) oder, soweit möglich, direkt mittels der menschlichen Bewegung eines Testarms **(Playback robots)** [FU].

„Intelligente" Roboter, die selbst neue Bewegungen auf Grund äußerer Sensordaten vollführen können (z.B. Kollisionsvermeidung) gibt es praktisch ebensowenig wie Roboter, die nicht nur einen, sondern zwei oder mehr Arme oder Beine koordiniert bewegen können; vielleicht sogar noch durch Bildauswertung gesteuert. Bei all diesen Problemen trifft die von-Neumann Architektur auf ihre Grenzen.

Neuere Versuche, diese Grenzen zu überwinden, bauen auf Softwaresysteme aus hierarchischen Schichten auf. Jede Schicht ist ähnlich gebaut: Sie empfängt Befehle von oben und setzt sie in konkrete Handlungssequenzen für die untergeordneten Schichten um. Andererseits werden die Sensorsignale der tieferen Schichten bei der Erstellung der Handlungssequenzen verwendet *(feedback)* sowie in abstrakterer, codierter Form nach oben weitergereicht (Beispiel: digitale Positionsdaten werden zur Meldung „Greifer im kritischen Bereich"). In Bild 3 ist solch ein Schichtenmodell gezeigt; parallel dazu in Bild 4 der vermutliche Aufbau der menschlichen Motorik (nach [SCH]).

Allgemein läßt sich feststellen, daß für die Bewegungssteuerung der Roboter eine Hard- und Softwarearchitektur benötigt wird, die nicht explizit programmiert zu werden braucht, sondern selbst die Bewegungsprimitive lernen kann und daraus die benötigten Bewegungen synthetisiert.

3. Neuronale, fehlertolerante Computer

Im Abschnitt 1 wurden die allgemeinen und in Abschnitt 2 spezielle Probleme der fehlenden Fehlertoleranz bezüglich Eingabefehler (Intelligente, menschengerechte Benutzerschnittstelle) und der schwierigen oder fehlenden Programmierung (Lernende Schnittstellen und Roboter) vorgestellt.

Abstrahieren wir von den konkreten Anwendungen, so lassen sich im wesentlichen drei Problemkreise abgrenzen, deren Lösung zentrale Bedeutung für die Bewältigung obengenannter Probleme und damit für die Konstruktion der neuronalen Computer hat. Beschreiben wir die Bild-, Ton- und Aktivierungsereignisse durch eine Menge von Variablen, zusammengefaßt in einem Mustervektor, so werden folgende Operationen für diese Muster benötigt:

1) ***Assoziativer Speicher***

- Speichern der Referenzmuster
- Vergleich unbekannter Muster mit den gespeicherten Referenzmustern, Entscheidung auf das ähnlichste Referenzmuster.

Die Modelle für Assoziativspeicher haben eine lange Tradition und sind sehr zahlreich. Allgemein läßt sich feststellen, daß diese Assoziativspeicher für ihre Speicherfunktion weniger Muster speichern können (je nach Modell 15%–60%) als die konventionellen adress-orientierten Speicher und Assoziativspeicher mit flag-Architektur. Dafür aber sind die verteilten Assoziativspeicher schneller (die Speicher- und Ausleseoperation ist in einem Funktionszyklus beendet) und, bei nichtlinearer Ausgabefunktion, auch fehlertolerant.

2) ***Zeitsequenzen***

Im Unterschied zur statischen Bildverarbeitung, bei der alle Bildmerkmale gleichzeitig vorliegen, ist bei der Bildfolgeerkennung und bei den zeitlichen Sequenzen der Sprachlaute das zu erkennende Muster zeitlich auseinandergebrochen. Gesucht ist ein Mechanismus, der die Operationen des assoziativen Speichers auch bei zeitlich sequentiellen Mustern durchführt.

Auch die Aktivierungsmuster der Roboter sind zeitlich gegliedert. Beide Probleme hängen eng zusammen: Ist ein befriedigender Mechanismus gefunden, ein zeitlich sequentielles Muster fehlertolerant zu erkennen und den dazu assoziierten Code auszugeben, so wird auch bei autoassoziativem Gebrauch durch Eingabe des Codes mittels Ergänzung die entsprechende zeitliche Sequenz ausgelöst.

Leider gibt es nur wenige Modelle, die Zeitsequenzen darstellen können. Vielfach tritt auch das Problem auf, daß zwei Folgen, die identische Teilfolgen enthalten, bei der Generation bzw. Erkennung verwechselt werden können.

3) ***Feature extraction***

Aus unbekannten Sensordaten (Input-Muster) müssen die typischen, charakteristischen Merkmale als solche erkannt und extrahiert werden. Dabei muß das Wissen über bereits erkannte und gespeicherte Features herangezogen werden.

Leider ist der Begriff „typisches Merkmal" nicht sehr präzise definiert. Deshalb gibt es viele Modelle, die vorgeben, eine Merkmalsextraktion durchzuführen und in Wirklichkeit nur eine Form von Mustererkennung durchführen. Zu einer echten Merkmalsextraktion gehört dagegen das Erkennen und Auswählen von Merkmalen, die vorher unbekannt waren; also die Ermittlung von Zahl und Art von unbekannten Merkmalen und eine Rekodierung („Abstraktion") der Originalinformation, was bisher in den Systemen der symbolischen, künstlichen Intelligenz nicht möglich ist. Ein interessantes Modell dazu bildet beispiels-

weise die Gruppe der stochastisch selbstlernenden Konkurrenz-Modelle (**Competetive Learning,** z. B. [GROS]).

Mit diesen drei Grundoperationen lassen sich die untersten Schichten der vielschichtigen Bild- und Spracherkennung konstruieren: Bild- und taktile Erkennung mit Operation 1, Spracherkennung und Robotersteuerung mit Operation 2 und Selbstprogrammierung und Lernen sowie Codierung der Sensorereignisse mit Operation 3.

Dabei sind die drei Operationen sicher nicht unabhängig voneinander: Grundlegend ist das Speichern von Referenzmustern und der schnelle, fehlertolerante Vergleich mit unbekanntem Input durch assoziativen Abruf. Versehen wir den Assoziativspeicher mit einem Kurzzeitspeicher, so läßt sich Operation 2 konstruieren (vgl. [KOH2]). Auch bei der Merkmalsgewinnung der Operation 3 wird die Grundfunktion des Speicherns und Vergleichens benötigt. Es gibt sogar Arbeiten (Ritter und Schulten, TU München) die nur mit Hilfe eines fehlertoleranten, selbstorganisierenden Konkurrenzmodells (nachbarschaftserhaltende Abbildung, s. [KOH2]) eine Roboterbewegung programmiert.

Obwohl viele Modelle Berührungspunkte haben und ähnliche Phänomene erklären wollen, gibt es leider noch kein einheitliches Theoriegerüst für die Neuronalen Netze, ebensowenig wie umfassende Untersuchungen ihrer (zweifellos vorhandenen) Fehlertoleranzeigenschaften.

Fehlertoleranz in Assoziativspeichern

Da alle drei Operationen zur Zeit noch wenig analytisch erforscht sind, beschäftigten wir uns in Frankfurt zur Zeit noch ausschließlich mit der fehlertoleranten assoziativen Speicherung.

Es ist allgemein bekannt, daß Schwellwerte bei Signalpegeln zur Rausch- und Störunterdrückung und damit zur Fehlertoleranz bezüglich der Signalpegel beitragen.

In [BRA] wird genauer untersucht, was die Einführung von Schwellwerten (Nichtlinearitäten) bei der Ausgabefunktion von linear gekoppelten und zu einem Assoziativspeicher geschalteten Verarbeitungseinheiten (formale Neuronen) für die Tolerierung von defekten, fehlerhaften Eingabemustern bewirkt.

Es zeigt sich, daß die Menge aller möglichen Eingabemuster in Untermengen (Klassen) zerfällt, die durch die gespeicherten Muster als Klassenprototypen bestimmt werden. Die Ausleseoperation des Assoziativspeichers wird damit zu einer Mustererkennungsoperation, bei dem das Eingabemuster dasjenige Ausgabemuster erregt, das zu dem ihm am ähnlichsten Klassenprototypen assoziiert ist.

Die Vektorquantisierungseigenschaften dieses Modells bewirken somit nicht nur einen Mustererkennungs- oder Kategorisierungsprozeß, sondern sind auch mit einem inhärenten Fehlertoleranzmechanismus gegenüber gestörten Eingabedaten identisch.

Auch bei dem parallel dazu untersuchten Hardwaremodell zeigt sich, daß durch Einführung der Schwellwerte große Teile der Hardware ausfallen können, bevor die korrekte Funktion des Assoziativspeichers gestört wird.

Literatur

[BALL] D. H. Ballard, Ch. Brown: Computer Vision. Prentice Hall 1982.

[BRA] R. Brause: Fehlertoleranz in parallelen Systemen. Interner Bericht der Praktischen Informatik (VSFT), J.-W.-Goethe Universität Frankfurt, 1987.

[DE MORI] R. De Mori, C. Suen: New Systems and Architectures For Automatic Speech Recognition And Synthesis. Springer Verlag 1984.

[FELD] J. A. Feldman, D. H. Ballard: Computing with connections. University of Rochester, Computer Science Department, TR72, 1980.

[FU] Fu, Gonzales, Lee: Robotics: Control, Sensing, Vision and Intelligence. McGraw-Hill 1987.

[GOS] K. Goser, C. Foelster, U. Rueckert: Intelligent memories in VLSI. Information Sciences 34, p. 61–82, 1984.

[GROS] S. Grossberg: Adaptive pattern classification and universal recoding. Biological Cybernetics, 23.

[HILL] D. Hillis: The Connection Machine. MIT Press, Cambridge, Massachusetts, 1985.

[HIN1] Hinton, Anderson: Parallel Models of Associative Memory. Lawrence Erlbaum associates, Hillsdale 1981.

[KOH1] T. Kohonen: Correlation Matrix Memories. IEEE Transactions on Computers C21 1972.

[KOH2] T. Kohonen: Self-Organisation and Associative Memory. Springer Verlag Berlin, New York, Tokyo 1984.

[KUHL] J. Kuhl, S. Reddy: Distributed Fault – Tolerance for Large Multiprocessor Systems. Proc. 7th Symp. on Comp. Architect., La Baule, France 1981.

[LONG] H. C. Longuet-Higgins: Holographic model of temporal recall. Nature 217, 1968. p. 104.

[McCUL] W. S. McCulloch, W. H. Pitts: A Logical Calculus of the Ideas Imminent in Neural Nets. Bulletin of Mathematical Biophysics Vol. 5, 1943, pp. 115–133.

[PEA] M. Pearse, R. Shostak, L. Lamport: Reaching Agreement in the Presence of Faults. Comm. of the ACM, Vol. 27/2, April 1980.

[SCH] R. F. Schmidt, G. Thews: Einführung in die Physiologie des Menschen. Springer Verlag, Berlin 1976.

[SHA] E. Shapiro: A Subset of Concurrent Prolog and Its Interpreter. Technical Report TR003. Institute for New Generation Computer Technology (ICOT), Tokyo 108, Japan.

[TAM] N. Tamura, Y. Kaneda: Implementing Parallel Prolog on a Multi-Processor Machine. IEEE Int. Symp. on Logic Programming, Atlanta City 1984.

[WIL] D. Willshaw: Models of distributed associative memory. Unpublished doctoral dissertion, Edinburgh University 1971.