Published in Proc. SOAVE'97, VDI Fortschrittberichte Reihe 8, Nr. 663, 1997, pp. 203-211

Visuomotorische Koordination eines Roboterarmes mit Kohonen-Karten, Neuronalem Gas und Dynamischen Zellstrukturen - ein Vergleich

Jörg Bruske, Erika Ábrahám-Mumm, Gerald Sommer

Lehrstuhl für Kognitive Systeme
Christian-Albrechts-Universität zu Kiel
email: jbr@informatik.uni-kiel.de

Zusammenfassung

Gegenstand dieses Beitrages ist ein Vergleich dreier KNN Modelle am Beispiel der visuomotorischen Koordination eines Roboterarmes. Die KNN Modelle sind die Erweiterte Kohonen-Karte von Ritter et al., das Neuronale Gas von Martinetz und die Dynamischen Zellstrukturen von Bruske et al.. Dabei liegt der Schwerpunkt auf der Einordnung der Dynamischen Zellstrukturen. Es stellt sich heraus, daß alle drei Modelle die Lernaufgabe erfolgreich und mit ähnlichem Restpositionierungsfehler meistern. Hinsichtlich der Lerngeschwindigkeit erweist sich das Neuronale Gas den beiden anderen Modellen überlegen, wenn als Lernzeit die Anzahl der Trainingsschritte gemessen wird. Dieser Unterschied verschwindet, wenn die Lernzeit in CPU-Zeit (einer seriellen Maschine) gemessen wird. Die Dynamische Zellstruktur investiert während der Trainingsphase in den Aufbau ihrer topologieerhaltenden Verbindungsstruktur, kann diese dann aber zur beschleunigten Ausgabeberechnung einsetzen. Hinsichtlich des erreichbaren Restpositionierungsfehlers ergibt sich bei gleicher Neuronenanzahl ein leichter Vorteil der Dynamischen Zellstruktur. Es erweist sich auch die Verwendung einer Einer-Nachbarschafts-Lernregel als ausreichend zur Erzielung von Kooperationseffekten.

Unsere Untersuchungen umfassen sowohl Simulation als auch reales Experiment mit einem Stäubli RX90 Roboterarm.

1 Einführung

Diese Arbeit orientiert sich an den fast klassisch zu nennenden Versuchen von H. Ritter et al. [RMS91] zum Erlernen der Hand-Auge Koordination eines simulierten Roboterarms und eines den Arbeitsbereich beobachtenden simulierten Stereokamerasystems. Das System lernt allein aus der Beobachtung des Robotermanipulators durch das Kamerasystem, einen 4-dimensionalen Vektor, der die kartesische Position der Projektion eines Punktes im 3D Arbeitsbereich auf die Bildebenen der beiden Kameras repräsentiert, mit dem zum Anfahren dieses Punktes nötigen Gelenkstellwinkeln zu assoziieren. Trotz fehlenden Vorwissens in Form eines Roboterarmmodells oder der Abbildungsgleichungen der Kameras und ungünstigen Randbedingungen, wie zufälliger Initialisierung des Neuronalen Netzes und zufälliges Abtasten des Arbeitsraumes, lernte das System in deutlich weniger als 10000 Bewegungen mit hoher Genauigkeit Punkte im

Arbeitsraum anzufahren. Desweiteren wurde gezeigt, daß das System innerhalb seines Arbeitsbereiches in der Lage ist, sich schnell an Veränderungen der Kinematik des Roboterarms, z.B. eine Verkürzung eines Armsegments, anzupassen.

Der Versuchsaufbau wurde von zahlreichen Forschern auf dem Gebiet visuomotorischer Koordination mittels KNN aufgegriffen, so auch von Martinetz [Mar92] und Walter et al. [WS93]. Beide Autoren demonstrierten erfolgreich die Anwendbarkeit auf die Steuerung eines realen Industrieroboters. Diese Vorgeschichte war Motivation genug, auch unsere Dynamischen Zellstrukturen (DCS) [BS95] im Versuchsaufbau nach Ritter et al. zu testen, um so einen direkten Vergleich zwischen Erweiterter Kohonenkarten (EKK), Neuronalem Gas (NG) und DCS treffen zu können.

2 Der Positioniervorgang

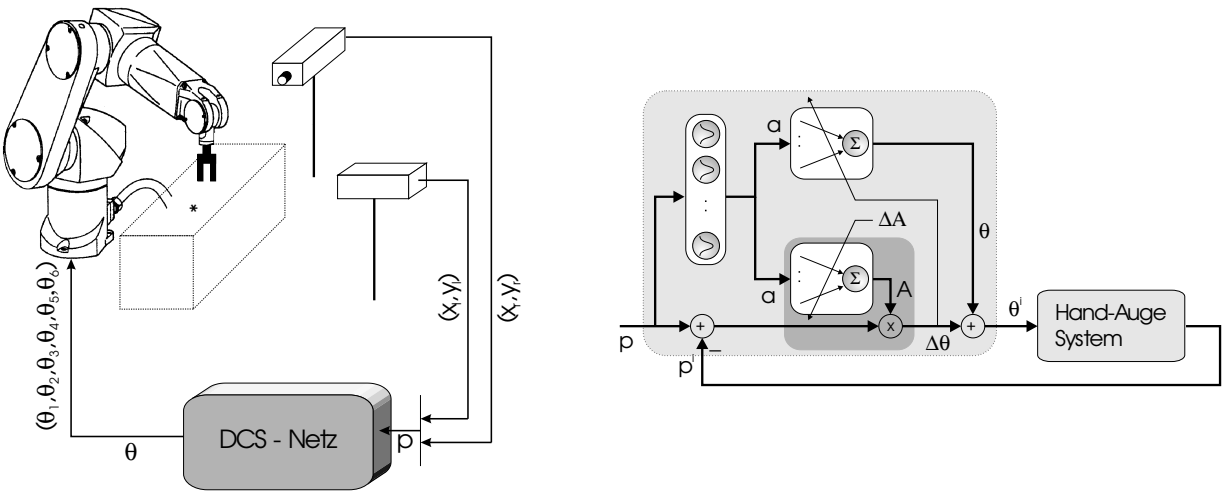

Abbildung 1: Versuchsaufbau zum Erlernen der Hand-Auge Koordination eines Roboterarmes und zweier Kameras (links) und Skizze des verwendeten DCS Netzwerkes (rechts). Beschreibung im Text.

Abbildung 1 zeigt den prinzipiellen Versuchsaufbau. Ein (starres) Stereokamerasystem erfaßt einen vorgegebenen Zielpunkt und liefert dessen retinale Koordinaten $p = (x_l, y_l, x_r, y_r)$. Der vierdimensionale Vektor p kodiert eindeutig die Position des Zielpunktes im dreidimensionalen Arbeitsbereich und wird genutzt, um aus einer Menge von Neuronen dasjenige zu ermitteln, dessen Zentrumsvektor c_i den geringsten Euklidischen Abstand zu p hat. Dieses Neuron sei im folgenden als *best matching unit*, bmu_p, bezeichnet. Jedem Neuron ist nun eine Jakobi Matrix A_i und ein Gelenkwinkelvektor θ_i zugeordnet. Der zum Anfahren des Zielpunktes mit dem Roboterarm nötige Gelenkwinkelvektor θ^0 wird damit zu

$$\theta^0 = \begin{cases} \theta_{bmu_p} + A_{bmu_p}(p - c_{bmu_p}) & : \text{ EKK und NG} \\ \sum_{i \in Nh^+(bmu_p)} [\theta_i + A_i(p - c_i)] a_i & : \text{ DCS} \end{cases} . \tag{1}$$

berechnet, wobei die Aktivierungen a_i für das normalisierte DCS Netz durch

$$a_i = \frac{h_i((p - c_i)^2)}{\sum_{j \in Nh^+(bmu_p)} h_j((p - c_j)^2)} . \tag{2}$$

gegeben sind. Die Funktionen $h_i : R^+ \rightarrow [0, 1]$ implementieren hierbei radiale Basisfunktionen

(RBF)[1]. $Nh^+(bmu_p)$ umfaßt die Menge der best matching unit und ihrer direkten Nachbarn bezüglich der lateralen Verbindungsstruktur des DCS Netzwerkes, siehe [BS95]. Diese laterale Verbindungsstruktur wird während des Trainings aufgebaut und durch Hebbsches Wettbewerbslernen in Richtung einer perfekt topologierhaltenden Karte adaptiert, siehe [MS94].

In allen drei Fällen findet eine lokal lineare Approximation statt. Der einzige Unterschied besteht darin, daß im ersten Fall der Term nullter Ordnung (die Gelenkwinkel θ_{bmu}) und der Term erster Ordnung (die Jakobi-Matrix A_{bmu}) in der gesamten Voronoizelle der bmu konstant angenommen werden, während dies im Falle des DCS Netzes durch RBF Interpolation zwischen der bmu und ihren Nachbarn nicht der Fall ist. Damit sollte - und dies wird sich im Experiment bestätigen - bei gleicher Neuronenzahl mit dem DCS Netz eine höhere Genauigkeit erzielbar sein.

3 Fehlerrückkopplung zur Adaption

Nach Anfahren der Gelenkwinkel θ^0 nimmt das Kamerasystem den Endeffektor an den retinalen Koordinaten p^0 wahr. Die Differenz zwischen Sollwert p und Istwert p^0 in Bildkoordinaten wird nach dem Fehlerrückkopplungsprinzip in eine Differenz $\Delta\theta$ für verbesserte Ausgabegelenkwinkel umgerechnet. Ein einfacher P-Regler reicht hier nicht aus. Ist aber die Jabcobimatrix A,

$$A = \begin{cases} A_{bmu} & : \text{ EKK und NG} \\ \sum_{i \in Nh^+(bmu)} A_i a_i & : \text{ DCS} \end{cases}, \tag{3}$$

lokal gültig, so kann diese benutzt werden, und man erhält als Fehlerrückkopplungssignal

$$\Delta\theta \sim A(p - p^0). \tag{4}$$

Die Jacobi-Matrix A nimmt jedoch zu Beginn entsprechend der Initialisierung der A_i einen eher zufälligen Wert an. Deshalb gilt es, zuerst die Ausgabematrizen A_i zu adaptieren, um dann das Fehlerrückkopplungssignal nach (4) zur Adaption der Ausgabegelenkwinkel θ_i nutzen zu können. Hierzu werden eine zweite Gelenkwinkelstellung θ^1 und ihre zugehörigen Bildkoordinaten p^1 benötigt. Wird diese unter Ausnutzung des Fehlerrückkopplungssignals in einem Korrekturschritt angefahren, i.e. $\theta^1 = \theta^0 + A(p - p^0)$, so konnten Ritter et al. [RMS91] zeigen, daß sich mit

$$\Delta A = [A(p - p^1)](p^1 - p^0)^T \tag{5}$$

eine Verbesserung von A durch stochastischen Gradientenabstieg auf einer quadratischen Fehlerfunktion ergibt.

4 Adaption der KNN Modelle

Wir wollen nun den Lernvorgang für die drei untersuchten Netzwerke erläutern. Dieser besteht im einzelnen aus: Initialisierung, Adaption der Zentrumsvektoren c_i, Adaption der Ausgabematrizen A_i und der Ausgabegelenkwinkel θ_i sowie Adaption der Topologie im Falle des DCS Netzwerkes.

[1] In den hier beschriebenen Versuchen wurden als Basisfunktionen rationale Funktion, $h_i(x) = 1/(\sigma x + 1)$, mit konstantem σ verwendet.

Initialisierung Um einen Vergleich der KNN Modelle zu ermöglichen, ist die Initialisierung für alle drei Netzwerke identisch: Es werden N Knoten vorgegeben, deren Zentren c_i zufällig im retinalen Eingaberaum initialisiert und deren Ausgabematrizen A_i mit kleinen zufälligen Anfangswerten belegt werden. Die Ausgabegelenkwinkel θ_i werden alle mit derselben Winkelstellungen initialisiert, die eine Manipulatorposition im Arbeitsbereich beschreibt. Im Falle des DCS Netzwerkes wird zudem beim Einfügen ein neuer Knoten mit seinem im retinalen Eingaberaum nächsten Nachbarn verbunden und so die laterale Verbindungsstruktur initialisiert. Alle N Knoten werden in der Initialisierungsphase eingefügt, und nicht wie in [BS95] im Rahmen eines fehlergesteuerten Wachstums der Zellstruktur während des Trainings.

Adaption der Zentren Die Adaption der Zentren erfolgt nach den selbstorganisierenden Lernregeln von Kohonen für die Kohonenkarte [Koh87] von Martinetz für das Neuronale Gas [Mar92] und der Lernregel nach Fritzke zur Adaption einer Einer-Nachbarschaft [Fri95] für das DCS Netzwerk:

$$\Delta c_i = \begin{cases} \epsilon(t)(p - c_i) g_{bmu,i}^{Kohonen}(t) & : \text{EKK} \\ \epsilon(t)(p - c_i) g_{ord(i)}^{NG}(t) & : \text{NG} \\ \epsilon_i(t)(p - c_i)\ ,\ i \in Nh^+(bmu) & : \text{DCS} \end{cases} . \tag{6}$$

Hierbei bezeichnen $g_{bmu,i}^{Kohonen}(t)$ die vom Abstand des i-ten Knotens von der *bmu* bezüglich der 3-dimensionalen Gitterstruktur abhängige Aktivierungsfunktion des Kohonennetzes und $g_{ord(i)}^{NG}(t)$ die von der Sortierreihenfolge $ord(i)$ bezüglich des Euklidischen Abstandes zur *bmu* abhängige Aktivierungsfunktion des Neuronalen Gases. Die Lernparameter ϵ und die Stärke der Nachbarschaftskooperation sowohl der Lernregel für die Kohonenkarte als auch der Lernregel für das Neuronale Gas sind zeitabhängig und nähern sich mit der Zeit dem Wert Null. Dies gilt in dieser Arbeit auch für die Lernparameter ϵ_i zur Adaption der Zentren der *bmu* und ihrer direkten topologischen Nachbarn im DCS Netz.

Adaption der Topologie Das Kohonennetzwerk hat feste dreidimensionale Topologie, angepaßt an den 3-dimensionalen Arbeitsbereich. Beim Neuronalen Gas erfolgt eine Neuberechnung der Ordnungsrelation in jedem Schritt, womit auch bei diesem Modell die Topologie nicht gelernt werden muß. Im Gegensatz hierzu muß das DCS Netzwerk erst lernen, daß die intrinsische Dimensionalität des Eingaberaumes dreidimensional ist. Hierzu wird die laterale Verbindungsstruktur des DCS Netzwerkes in Anlehnung an [MS94] durch Hebbsches Wettbewerbslernen in Richtung einer perfekt topologieerhaltenden Karte adaptiert, siehe [BS95]. Die Lernregel besagt im wesentlichen, daß bei Präsentation eines Eingabedatums jeweils zwischen *best* und *second best matching unit* eine laterale Verbindung aufzubauen ist.

Adaption der Ausgabematrizen und -gelenkwinkel Zur Adaption der Ausgabegelenkwinkel θ_i wird in allen drei Fällen das Fehlerrückkopplungssignal $\Delta\theta$ nach (4) verwendet. Die Adaption der Ausgabegematrizen efolgt nach (5). Mit diesen Größen lassen sich verbesserte Schätzer für θ und A angeben, vergl. [RMS91]:

$$\theta^* = \theta^0 + \Delta = \theta^0 + A(p - p^0) \quad \text{und} \tag{7}$$

$$A^* = A + \frac{\Delta A}{\| p^1 - p^0 \|^2} = A + \frac{[A(p - p^1)](p^1 - p^0)^T}{\| p^1 - p^0 \|^2} \tag{8}$$

Die Lernregeln lauten nun

$$\Delta A_i = \begin{cases} \alpha(A^* - A_i) g_{bmu,i}^{Kohonen}(t) & : \text{EKK} \\ \alpha(A^* - A_i) g_{ord(i)}^{NG}(t) & : \text{NG} \\ \alpha(A^* - A) a_i\ ,\ i \in Nh^+(bmu) & : \text{DCS} \end{cases} \tag{9}$$

$$\Delta\theta_i \;=\; \begin{cases} \alpha(\theta^* - \theta_i) g_{bmu,i}^{Kohonen}(t) & : \text{EKK} \\ \alpha(\theta^* - \theta_i) g_{ord(i)}^{NG}(t) & : \text{NG} \\ \alpha(\theta^* - \theta^0) a_i \;,\; i \in Nh^+(bmu) & : \text{DCS} \end{cases} . \qquad (10)$$

Ein in den obigen Lernregeln hervortretender Unterschied zwischen erweiterten Kohonenkarten, Neuronalem Gas und DCS ist, daß in den ersteren Modellen ein wirklicher Gradientenabstieg nur bezüglich der *bmu* stattfindet, während im DCS alle an der Ausgabeberechnung beteiligten Knoten nach den Gradienten adaptiert werden. Die Adaption der topologischen Nachbarn in der Kohonenkarte und im Neuronalen Gas basiert allein auf Nachbarschaftskooperation. Dies führt zwar in dieser Anwendung zu qualitativ ähnlichen Ergebnissen, es wird sich aber im allgemeinen keine zu minimierende Fehlerfunktion angeben und damit nur schwer eine Konvergenzaussage treffen lassen.

5 Experimentelle Resultate

Wir wollen uns nun den experimentellen Resultaten zuwenden und hier den Schwerpunkt auf die Erklärung der unterschiedlichen Lernverläufe legen. Die entsprechenden Simulationen und die Implementation auf einem realen Stäubli R90 Industrieroboter in Zusammenspiel mit einem Stereokamerasystem wurden in [Abr96] durchgeführt. Dort finden sich auch weitere experimentelle Details sowie die verwendeten Lernparameter, die hier aus Platzgründen nicht aufgeführt sind.

5.1 Simulation

Abbildung 2 zeigt die geglätteten Verläufe der Positionierungsfehler für die drei Netzwerkmodelle, wobei ein $7 \times 7 \times 7$ Kohonengitter sowie ein Neuronales Gas und ein DCS Netz mit jeweils 300 Neuronen verwendet wurden. Links ist der Positionierungsfehler über der Anzahl der Trainingsschritte aufgetragen, rechts über der verbrauchten CPU-Zeit zur Ausgabeberechnung und Adaption (wobei die Lernkurve des DCS-Netzwerkes 15000 Schritte umfaßt, die der Kohnenkarte und des Neuronalen Gases jeweils nur 4000!). Tabelle 1 zeigt den Positionierungsfehler nach 4000 und nach 15000 Schritten Training sowie die relative Zeit, die ein Trainingsschritt durchschnittlich benötigt. Der simulierte Arbeitsbereich für diese Experimente umfaßte $30 \times 30 \times 30\,cm^3$, der Fehler mißt die Abweichung zwischen Zielpunkt und Manipulatorkoordinaten im Arbeitsbereich in *cm*. Der (hier simulierte) Stäubli RX90 Roboterarm hat sechs Freiheitsgrade (6 DOF): Drehung um die vertikale Achse, Bewegung des unteren und oberen Armsegmentes in der vertikalen Ebene, Drehung des oberen Armsegmentes sowie Kippen und Drehen des Endeffektors. Zu sehen ist, daß das Neuronale Gas mit Abstand die wenigsten *Trai-*

Netztyp	Pos.fehler [*cm*] (4000 Schritte)	Pos.fehler [*cm*] (15000 Schritte)	rel. CPU-Zeit/ Trainingsschritt
Kohonen Karte	0.04	0.04	7
Neuronales Gas	0.03	0.03	80
DCS Netz	0.3	0.01	3

Tabelle 1: Präzision und relative CPU-Zeit pro Trainingsschritt für Kohonenkarte, Neuronales Gas und DCS Netz mit $7 \times 7 \times 7$ bzw. 300 Neuronen

ningsschritte, das DCS hingegen die meisten benötigt. Zudem fällt der Positionierungsfehler des

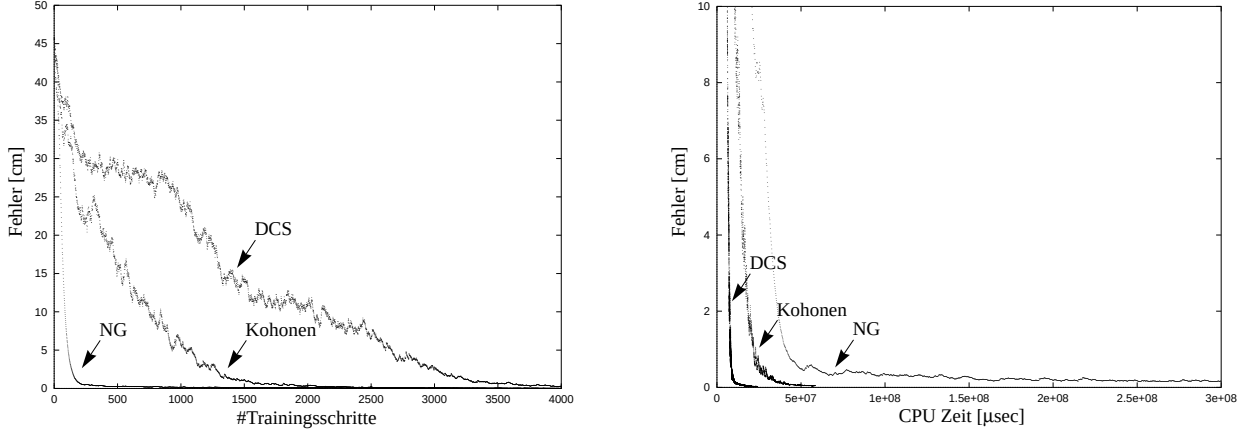

Abbildung 2: Positionierungsfehler im Arbeitsbereich für Kohonengitter, Neuronales Gas und DCS mit $7 \times 7 \times 7$ bzw. 300 Neuronen. Links: über der Anzahl der Trainingsschritte, rechts: über der zur Ausgabeberechnung und Adaption benötigten CPU-Zeit.

Neuronalen Gases und der Kohonenkarte nach 4000 Zeitschritten geringer aus als der des DCS Netzwerkes. Trainiert man jedoch weiter (über 10000 Schritte), so ist zu bemerken, daß das DCS Netz einen leicht geringeren Endpositionierungsfehler erreicht. Betrachtet man hingegen den Verlauf des Positionierungsfehlers über der *Trainingszeit*, wobei hier die CPU-Zeit gemessen wurde, die jedes Modell zur Berechnung der Anfahrpunkte und der Adaption benötigt, so schneidet das DCS-Netz am besten ab, das Neuronale Gas am schlechtesten. Die CPU-Zeit wurde dabei auf einer SPARC 10 Workstation gemessen. Für keines der Modelle erfolgte eine spezielle Optimierung. Zur Sortierung nach dem Euklidischen Abstand im Neuronalen Gas wurde eine Standardimplementierung des Quicksortverfahrens genutzt.

Das Experiment verdeutlicht die Unterschiede in der Lernphase zwischen den drei Netztypen: Im Neuronalen Gas sind die Neuronen nach der zufälligen Initialisierung im Arbeitsbereich schon nahezu optimal verteilt, und die topologische Beziehung zwischen den Neuronen muß nicht erst gelernt werden, sondern wird durch Sortierung in jedem Schritt neu berechnet. Damit ist die nötige Zentrenadaption minimal und es wird sofort mit dem Erlernen korrekter Ausgabematrizen und -gelenkwinkel begonnen. Sortierung aber benötigt auf einem seriellen Rechner $O(n \log n)$ Zeit, weshalb die Berechnungszeit pro Trainingsschritt ebenso steigt und bei 300 Neuronen deutlich höher ist als bei den beiden anderen Modellen. Die Kohonenkarte kommt ebenfalls ohne Topologielernen aus und ist in diesem Fall optimal an die dreidimensionale Geometrie des Arbeitsraumes angepaßt. Allerdings sind zu Beginn die Neuronen zufällig im Arbeitsraum verteilt, so daß sich das Netz erst entfalten muß.

Im DCS Netz hingegen muß erst die dreidimensionale Topologie des Arbeitsraumes erfaßt und seine laterale Verbindungsstruktur aufgebaut werden, bevor es zu einer korrekten Adaption der Ausgabematrizen und -gelenkwinkel kommen kann. In Experiment waren es am Ende des Trainings ca. 1800 laterale Verbindungen, die aufgrund des zufälligen Abtastens des Arbeitsraumes erlernt wurden. Nach dieser "Investitionsphase" hat das DCS Netz allerdings die topologische Information, die sich das Neuronale Gas in jedem Schritt neu errechnen muß, in seiner lateralen Verbindungsstruktur gespeichert. Die Ermittlung der *bmu* und ihrer direkten topologischen Nachbarn kann nun in der Zeit $O(n)$ erfolgen. Es ist auch nur diese Teilmenge von Neuronen, die in einem Trainingsschritt aktiviert und adaptiert werden muß. Zugleich nimmt das DCS Netz als einziges eine Interpolation zwischen den Ausgaben benachbarter Neuronen vor und ist dadurch in der Lage, bei gleicher Neuronenanzahl eine höhere Approximationsgüte als die Kohonenkarte oder das Neuronale Gas zu erzielen.

Einen ähnlichen Lernverlauf bezüglich der Anzahl der Trainingsschritte zeigt Abbildung 3,

in der auf dem selben Arbeitsbereich wesentlich kleinere Netze trainiert wurden (DCS und Neuronales Gas jeweils 30 Neuronen, $3 \times 3 \times 3$ Kohonen Karte). Zwar ist das Neuronale Gas immer noch das schnellste Netz, doch die Unterschiede in der Anzahl der benötigten Trainingsschritte sind weniger gravierend. Dies liegt daran, daß der Entfaltungsprozeß der Kohonenkarte kürzer ist und das DCS Netz weniger laterale Verbindungen aufbauen muß. Interessant ist der Peak in der DCS Lernkurve: Hier ist durch Ab- und Neuaufbau der lateralen Verbindungsstruktur die Zunahme des Positionierungsfehlers größer als seine durch fortdauernder Adaption der Ausgabeschicht bewirkte Abnahme.

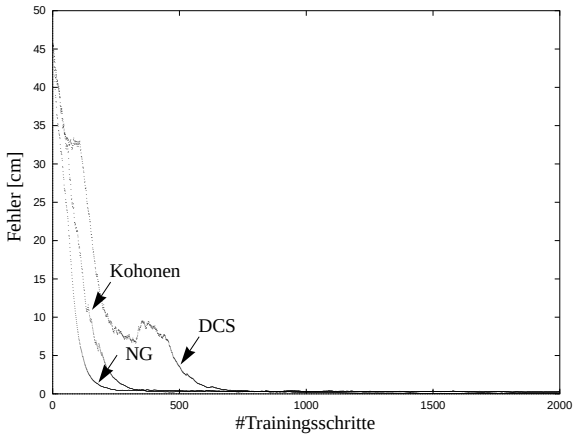

Abbildung 3: Positionierungsfehler im Arbeitsbereich für Kohonengitter, Neuronales Gas und DCS mit $3 \times 3 \times 3$ bzw. 30 Neuronen.

5.2 Reales Experiment

Zielstellung des realen Experiments war es, die Übertragbarkeit der Simulationsresultate insbesondere des DCS Netzes auf reale Einsatzbedingungen zu testen. Als Hardware dienten der Stäubli RX90 Roboterarm mit einem am Manipulator befestigten Lämpchen, zwei Sony CCD Kameras mit einer Auflösung von je 768×576 Pixeln (durch Unterabtastung auf 384×288 Pixel reduziert) und eine Sparc10 Workstation der Firma SUN inklusive zweier Framegrabber. Der Arbeitsbereich des Roboters umfaßte einen Bereich von $20 \times 40 \times 20\,cm^3$, die Kameras wurden mit einem Abstand von je $1.5\,m$ zum Arbeitbereich und einer Basislänge von $1\,m$ aufgestellt.

Die experimentellen Ergebnisse fanden sich qualitativ in Übereinstimmung mit unseren Simulationen und den in [WS93] publizierten Resultaten, wo bereits zuvor die Übertragbarkeit für das Neuronengas und die Kohonenkarte demonstriert worden war. Mit 30 respektive $3 \times 3 \times 3$ Neuronen lernten DCS, Kohonen-Karte und Neuronales Gas in ca. 500, 400 respektive 300 Schritten, den Manipulator mit Pixelgenauigkeit zu positionieren. Dieser durch die Auflösung der Kameras und durch die Detektionsgenauigkeit des Lämpchens bedingte Restfehler wurde auch von größeren Netzen nicht unterschritten. Die Lernkurven zeigten dabei einen ähnlichen Verlauf wie in Fig. 3.

Ein interessanter Aspekt des Experiments mit dem realen Roboterarm wie auch der Simulation war, daß sämtliche 6 Freiheitsgrade des Stäubli RX90 genutzt wurden. Die Jakobi-Matrizen A_i werden damit 6×4 dimensional. Durch die redundanten Freiheitsgrade des Roboterarms wird die Aufgabe, einen Punkt im Arbeitsbereich anzufahren, "schlecht gestellt", denn dies ist nun auf unendlich viele Weisen möglich. In ihren Simulationen zur visuomotorischen Koordination eines Roboterarms konnten Ritter et al. zeigen, daß aufgrund der Nachbarschaftskooperation im erweiterten Modell von Kohonen benachbarte Gitterknoten "gezwungen" werden, ihre Ausgabegrößen (Jakobi-Matrizen und Gelenkwinkel) hin zu ähnlichen Werten zu adaptieren,

und so die Ausgabegrößen nur wenig variieren. Das Resultat ist eine "glatte" Bewegung des Roboterarms, wenn dieser im Arbeitsbereich eine Reihe von benachbarten Punkten anfahren soll. Die Wirkung der Nachbarschaftskooperation kann damit als eine Regularisierung des Koordinationsproblems gedeutet werden. Im Experiment zeigte sich, daß derselbe Effekt auch für das trainierte DCS Netz zu beobachten ist: Beim Abfahren einer Linie im Arbeitsbereich (definiert durch eine Folge benachbarter Zielpunkte) ergibt sich ebenfalls eine glatte Armbewegung. Dies weist darauf hin, daß die Einer-Nachbarschaft der Lernregel nach Fritzke für den Nachbarschaftskooperationseffekt bereits ausreichend ist und auch die reine Gradientenregel zur Adaption der Ausgabegrößen (9) dem nicht entgegenwirkt.

6 Schlußbemerkung

Zum Abschluß der Betrachtungen und Experimente zu kalibrierungsfreien Hand-Auge Koordination mit den drei aufgeführten KNN Modellen sei noch einmal betont, daß unsere Versuche weniger die generelle Überlegenheit des einen über das andere KNN Modell als vielmehr ihre unterschiedliche Arbeitsweise demonstrieren sollen. Hierzu wurden die Experimente unter vereinheitlichten Rahmenbedingungen durchgeführt, die bewußt die individuellen Stärken der einzelnen Netztypen nicht nutzten. Auf fehlergesteuertes Wachstum der Dynamischen Zellstruktur oder des Neuronalen Gases wurde verzichtet, ebenso auf ein Vortraining zur Initialisierung der lateralen Verbindungsstruktur des DCS-Netzwerkes. Da für die Approximation in den ausgewählten Arbeitsbereichen eine gleichförmige Verteilung der Zentrumsvektoren die besten Ergebnisse zu liefern scheint, sind die selbstorganisierenden Lernregeln zur Generierung einer dichteabhängigen Zentrenverteilung auch die geeignetsten. Mit einer uniformen Eingabedichte erzeugen sie gerade eine gleichförmige Zentrenverteilung. Eine gleichförmige Zentrenverteilung ist aber im Allgemeinen nicht die beste, sie sollte am lokalen Approximationsfehler orientiert sein (Gleichverteilung bezüglich des lokalen Approximationsfehlers).

Literatur

[Abr96] E. Abraham. Neuronale Netze zur Visuomotorischen Regelung. Studienarbeit, Inst. f. Inf. u. Prakt. Math., CAU zu Kiel, 1996.

[BS95] J. Bruske and G. Sommer. Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, 7(4):845–865, 1995.

[Fri95] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1995.

[Koh87] T. Kohonen. Adaptive, associative, and self-organizing functions in neural computing. *Applied Optics*, 26:4910–4918, 1987.

[Mar92] T. Martinetz. *Selbstorganisierende Neuronale Netzwerkmodelle zur Bewegungssteuerung.* PhD thesis, Physik-Department TU Muenchen, 1992.

[MS94] T. Martinetz and K. Schulten. Topology representing networks. In *Neural Networks*, volume 7, pages 505–522, 1994.

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

[WS93] J. Walter and K. Schulten. Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Transactions on Neural Networks*, 4(1):86–95, January 1993.