Dynamic wave expansion neural network

Technische Fakultät – Universität Bielefeld
Seminar Humanoide Roboter
WS 04 / 05

Sven Kreft
svkreft@techfak.uni-bielefeld.de

Inhaltsverzeichnis

Einleitung

Diese Ausarbeitung soll einen Einblick in das Problem der Pfad- und Trajektorienplanung bei Robotern geben, sowie bisher entstandene Lösungsansätze vorstellen, um schließlich auf ein Modell näher einzugehen.
Die inhaltliche Struktur gliedert sich wie folgt: Beginnend mit einer generellen Beschreibung des Pfadplanungsproblems, wird anschließend eine Übersicht an bisher erarbeiteten Lösungsansätzen, welche die Aufgabe mit Hilfe neuronaler Netze behandeln, vorgestellt. Detailliert wird auf die Modellgrundlage und Funktionsweise des Dynamic wave expansion neural network (DWENN) eingegangen, welches im darauf folgenden Abschnitt hinsichtlich der Leistungsfähigkeit mit anderen Systemen, anhand von Simulationsergebnissen, bewertet wird. Abschließend wird ein Ausblick gegeben, der sowohl mögliche Einsatzgebiete, als auch bestehende Problemfelder in der Anwendung beleuchtet.

Problemstellung

Pfadplanung ist ein essentieller Bestandteil mobiler Roboter, insbesondere solcher, die humanoider Bauweise sind, da autonome Bewegungen, auch in einer nicht bekannten Umgebung, zu einer der natürlichen, aber auch herausfordernden Anforderung gelten. Prominente Beispiele sind Roboter, die Museumsführungen bewältigen, oder hinsichtlich eines kommerziellen Einsatzes, die Fähigkeit haben Aufgaben eigenständig im Haushalt zu bewältigen. Dieses weit in der Zukunft angesiedelte Beispiel verdeutlicht die Grundlagenproblematik mit der hier umgegangen wird.
Die Anforderungen, die an Systeme zur Berechnung von Pfaden gestellt werden sind intuitiv, Sicherheit, Optimalität sowie Natürlichkeit bzgl. der gewählten Strecke sind hier zu nennen. Während eine kollisionsfreie Bewegung eines Roboters als wichtigste Eigenschaft anzusiedeln ist um Schäden an sich und womöglich anderen vorzubeugen, und auch Optimalität hinsichtlich der Länge einer Strecke, als vorausgesetzt angesehen werden darf, ist die Natürlichkeit einer Bewegung ein Anliegen, welches eher zweitrangig ist, aber gerade in der humanoiden Robotik große Beachtung findet.

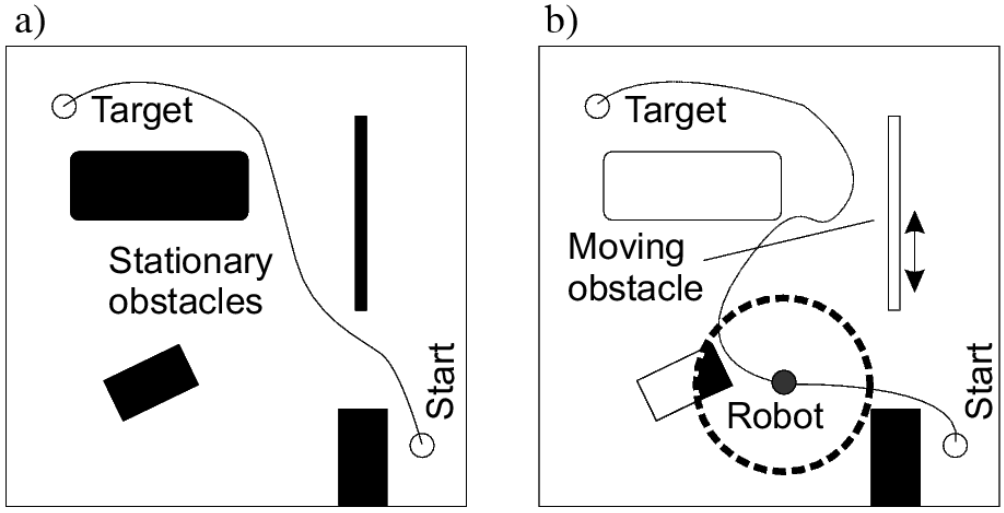

Abb. 1 a) statischer Fall b) dynamischer Fall

Man unterscheidet bei dem Problem der Pfadplanung mobiler Roboter den statischen von dem dynamischen Fall [Abb. 1].
Bei dem relativ trivialen statischen Fall ist die Umgebung, samt aller sich in ihr befindlichen (unbeweglichen) Hindernisse bekannt. Die Aufgabe besteht darin einen Weg vom Start- zum Zielpunkt zu finden, wobei der einfachste Ansatz darin besteht, eine gerade Linie zwischen den beiden Punkten zu benutzen und Hindernissen, die diesen Weg versperren, möglichst geschickt auszuweichen. Sind diese Hindernisse beweglich und sind ihre Bewegungstrajektorien bekannt, so beschränkt sich das Problem wieder auf den einfachen Fall, der um eine Zeitachse ergänzt wird.
Der deutlich kritischere Fall entsteht dann, wenn man initial weder die Umgebung, noch die sich in der Position veränderlichen Hindernisse kennt. Hier muss der Roboter zur Fahrzeit Informationen über seine Umgebung aufnehmen und diese in den Pfadplanungsprozess integrieren. Im folgenden wird nur der letztere Fall behandelt.

Existierende Lösungsansätze

Die folgende Tabelle gibt einen Überblick existierender Arbeiten. Sie ist in 2 Achsen geteilt. Zum einen wird die Art, wie dem Netz die Daten präsentiert werden aufgezeigt, zum anderen werden statischer und dynamischer Fall unterschieden. Alle Ansätze bedienen sich hierbei der Verwendung von neuronalen Netzen zur Berechnung des Pfades. Auffällig ist das Verhältnis in der Anzahl der Lösungsvorschläge, welches verdeutlicht, daß die hier im Mittelpunkt stehende dynamische Problemstellung sehr viel weniger behandelt wird, als der einfachere statische Fall.
Die Gemeinsamkeiten, der in der Tabelle dargestellten, zum DWENN konkurrierenden Projekte, sollen nun kurz erläutert werden. Die Pfadplanung wird bei allen Systemen im Roboter Arbeitsraum C durchgeführt. Es handelt sich hierbei um einen diskretisierten Hypercube im $\mathbb{R}^N$, wobei N die Anzahl der Freiheitsgrade des Roboters beschreibt.
Jede diskrete Position in C ist mit einem formalen Neuron assoziiert, welches mit allen Nachbarn verbunden ist.
Die jeweiligen neuronalen Netze berechnen Potentialfelder in dem Arbeitsraum, so daß sich der Roboter von der aktuellen Position in Richtung des Nachbarneurons bewegt, welches die größte Aktivität besitzt. Hindernisse haben entweder gar kein oder ein sehr geringes Potential.
Die Potentiale der einzelnen Neuronen werden unter Berücksichtigung aller Aktivitäten der Nachbarn berechnet, was zu einer hoch parallelen Architektur führt, die, wie man zeigen kann als parallele Prozesse implementierbar sind.

Authors	Network type	Dynamics	Repres.	Type
(Ageev & Istratov, 1998)	multi-layer, feed-forward	gradient minimiz. method	algebraic, explicit	stationary
(Lee & Kardaras, 1997)	multilayer network	simulated annealing	algebraic, explicit	stationary
(Meng & Picton, 1992)	single hidden layer back-propagation	learns collision penalties	algebraic, explicit	stationary
(Vleugels et al., 1997)	Kohonen-type, with 2 classes of nodes	road-map construction	algebraic, explicit	stationary
(Xia & Wang, 2000)	discrete-time, recurrent	linear optimization	algebraic, explicit	stationary
(Dracopoulos, 1998)	multilayer perceptron	learns to predict movement direction	grid-based	stationary
(Kassim & Vijaya Kumar, 1995)	two layers, locally connected	wave fronts propagation	grid-based	stationary
(Kindermann et al., 1996)	three layers, with local reccurent connections	diffusion-like activity propagation	grid-based	stationary
(Lemmon, 1991)	single layer, locally connected, oscillatory dynamics	produces oscillatory behavior	grid-based	stationary
(Siemiatkowska & Dubrawski, 1998)	cellular, two layers	diffusion-like activity propagation	grid-based	stationary
(Bugmann et al., 1995) (Tarassenko et al., 1991)	one or two layers, locally connected	diffusion-like activity propagation	grid-based	stationary, dynamic
(Chen et al., 2003)	Hopfield-type,	diffusion-like activity propagation	grid-based	stationary, dynamic
(Glasius et al., 1995)	topologically organised, locally connected	diffusion-like activity propagation	grid-based	stationary, dynamic
(Yang & Meng, 2000)	single layer, topologically organised, locally connected	diffusion-like activity propagation	grid-based	stationary, dynamic

Dynamic wave expansion neural network

- **Das Modell**

Die zu Grunde liegende Idee des Dynamic wave expansion neural network beruht auf der Rückverfolgung von Wellen, die von dem Ziel ausgehen.
Ein natürlich vorkommendes Phänomen verdeutlicht das Konzept. Ein in Wasser eintauchender Stein erzeugt Wellen, die sich konzentrisch von dem Punkt des Eintauchens ausbreiten [Abb. 2]. Verfolgt man nun eine beliebige Wellenfront zurück, so gelangt man zu der ursprünglichen Stelle, im Modell dem festgelegten Target [Abb. 3].

Abb. 2 natürliches Vorbild

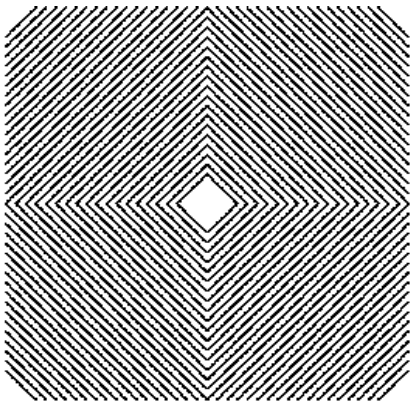

Abb. 3 Idee in der Modellierung

- **Funktionsweise und Architektur**

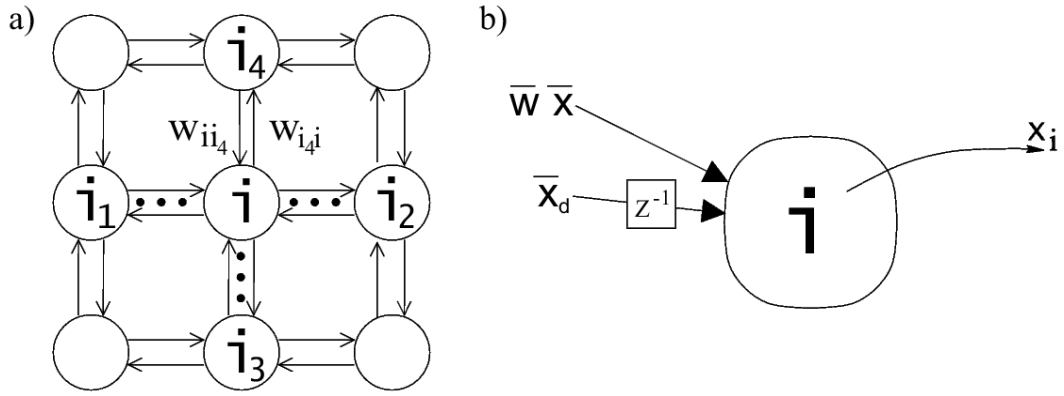

Abb. 4 a) Nachbarschaft b) formales Neuron

Das DWENN weist in seiner Funktion und Architektur einige Parallelen zu den anderen bereits erwähnten Arbeiten auf. Alle Aktivitäten der einzelnen Neuronen, die erneut für eine diskrete Position im Arbeitsraum des Roboters stehen, bilden ein skalares Potentialfeld. Auch hier ist jedes Neuron i mit der Menge $s_i = \{ i_1 \ldots i_n\}$ seiner Nachbarn verbunden [Abb. 4 a)].
Wellen neuronaler Aktivität entstehen ausgehend vom Target, die sich ähnlich dem natürlichen Vorbild ausbreiten, so daß sich eine höhere Aktivität eines Neurons in einem größerem Abstand zum Zielpunkt ausdrückt. Daraus ergibt sich ein Unterschied zu den konkurrierenden Arbeiten, da in der Modellierung des DWENN das Target die geringste Aktivität besitzt und sich der Roboter in Richtung desjenigen Nachbarneurons bewegt, welches das niedrigste skalare Potential aufweist. Damit ergibt sich eine Situation, die sich als 'Anziehen des Roboters durch das Target' ausdrücken lässt.
Die Aktivität x_i des einzelnen Neurons i zum Zeitpunkt t+1 hängt von der Aktivität der jeweiligen Nachbarn zum aktuellen Zeitpunkt ($\underline{\mathbf{x}} = (x_{i1}(t), \ldots, x_{in}(t))$, sowie der eigenen Aktivität und jener der Nachbarn zum vergangenen Zeitpunkt t−1 ab ($\underline{\mathbf{x}}_d = (x_i(t-1), x_{i1}(t-1), \ldots, x_{in}(t-1))$ [Abb. 4 b)]. Die Architektur des DWENN könnte als dynamisches System in diskreter Zeit charakterisiert werden, weil die Aktivitäten der einzelnen Neuronen und damit die Ausbreitung der Wellen in Zeitschritten berechnet werden.

- **Gewichte und Aktivität**

Da das DWENN die Gewichte ***vor*** der jeweiligen Aktivität der Neuronen bestimmt, soll an dieser Stelle auch zunächst die Berechnungsfunktion der Gewichte vorgestellt werden:

$$w_{ij}(t+1)=\begin{cases}\delta_{jk}, & \text{wenn } k\in s_i \text{ das } \textbf{\textit{erste}} \text{ neuron ist},\\ & \text{für welches (a)}-\text{(d) gelten},\\ 0, & \text{sonst}\end{cases}$$

(a) k ist kein Hindernis,
(b) $x_k(t)>0$, d.h. wenn es Teil der Aktivitätswellenfront ist und ***einige*** Informationen über die Äderungen im Arbeitsraum enthällt,
(c) $x_k(t)\neq x_k(t-1)$, d.h es enthällt ***neue*** Informationen,
(d) wenn $(x_i(t)+x_i(t-1))>0$, dann gilt $x_k(t)<x_i(t)$, d.h. seine Position ist näher an dem Target als die aktuelle Roboterposition

Bei δ_{jk} handelt es sich um das Kroneckersymbol, welches garantiert, dass es sich bei dem behandelten Gewicht um eines zu einem Nachbarn zeigendes handelt. Die Bedingungen (a) – (d) garantieren, daß jedes Neuron nur ein Gewicht besitzt, das nicht Null ist, welches auch noch in Richtung des negativen Gradienten, also zum Target zeigt.
Die Aktivierungsdynamik wird durch folgende Funktionen beschrieben:

(i) für das Target $(i=i_*(t))$:

$$x_i(t+1)=1,$$

(ii) für seine diekten Nachbarn $(i\in s_{i_*(t)})$:

$$x_i(t+1)=\begin{cases}x_i(t)+1, & \text{wenn } i_*(t+1)=i_*(t)\\ 2, & \text{sonst}\end{cases},$$

(iii) für alle anderen Neuronen:

$$x_i(t+1)=\sum_{j\in s_i} w_{ij}(t)\cdot(x_j(t)+2).$$

Hier wird in (i) festgelegt, dass das Target immer das geringste skalare Potential besitzt, während alle anderen Neuronen eine höhere Aktivität aufweisen, die mit der Entfernung zum Zielpunkt wächst (iii). (ii) behandelt die direkten Nachbarn des Targets und versorgt den möglichen Fall eines dynamischen Zielpunkts.

- **Eigenschaften**

Die vorgestellte Architektur des Netzes führt zu Eigenschaften die nun näher beleuchtet werden:

1. Jede Bewegung des Roboters geht entlang des Gradientenabstiegs in dem Potentialfeld, wobei dieser nicht explizit berechnet werden muss, da jedes Neuron nur ein Gewicht besitzt, welches in Richtung des negativen Gradienten und somit gen Target gerichtet ist.
2. In der vorgestellten Variante mit 4 Nachbarn [Fig XXX a)] besteht der resultierende Pfad nur aus geraden Linien, es entstehen unnatürliche Trajektorien. Wenn man die Anzahl der Nachbarn erhöht, um dieses Problem zu umgehen, entsteht ein erheblich höherer Rechenaufwand.
3. Im Gegensatz zu den anderen vorgestellten Arbeiten ist das DWENN Parameter frei, was ein erheblicher Vorteil ist, denn eine für alle Problemstellungen befriedigende Auswahl an Parametereinstellungen existiert für kein Modell.
4. Die hohe Performance des DWENN resultiert aus der Tatsache, daß nur eine Addition und einige bitweise Vergleiche (hängt von der Anzahl der Nachbarn ab) pro Neuron und Zeitpunkt, zur Berechnung der jeweiligen Aktivität benötigt werden.

Simulation

Um einen Eindruck von der Leistungsfähigkeit des DWENN im Vergleich mit konkurrierenden Systemen zu bekommen, werden ausgewählte Simulationsergebnisse grafisch dargestellt, wobei das DWENN zu den 3 betrachteten Problemstellungen jeweils mit einigen ausgewählten Systemen bzgl. des gefundenen Wegs und des berechneten Potentialfelds verglichen wird. Anschließende Tabellen geben einen Überblick mehrerer Systeme hinsichtlich der Anzahl der benötigten Schritte sowie der durchlaufenden Netziterationen für das aktuell behandelte Problem.

- **'Closing Gate':**

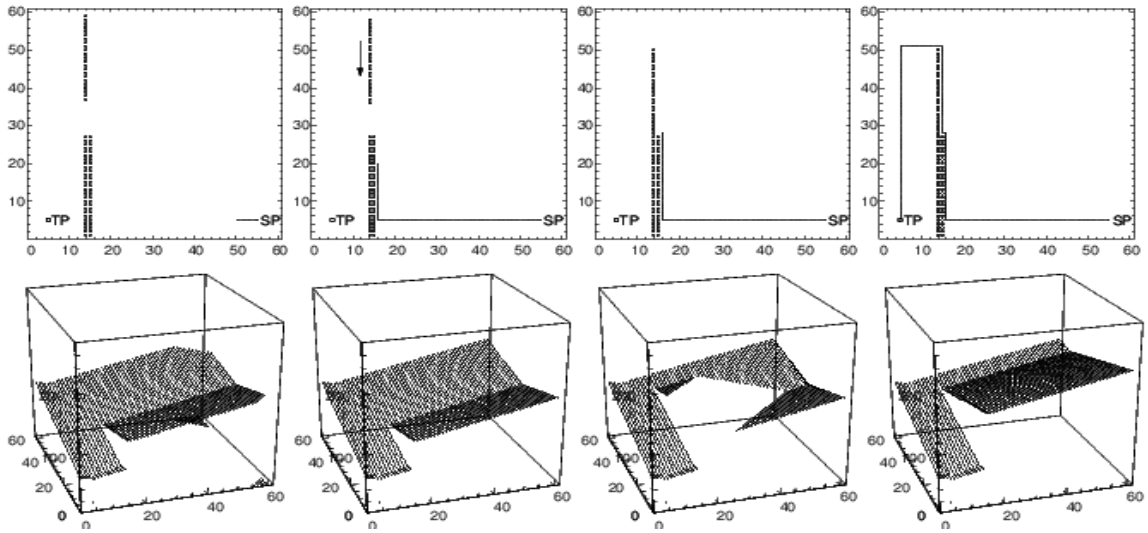

Abb. 5 DWENN: Gefundener Pfad und dazugehörige Potentialfelder

Hierbei handelt es sich um eine typische Aufgabe, an der Systeme gemessen werden. Der Roboter bewegt sich auf eine für ihn passierbare Stelle zu, kurz davor wird der Weg durch ein Hindernis blockiert.

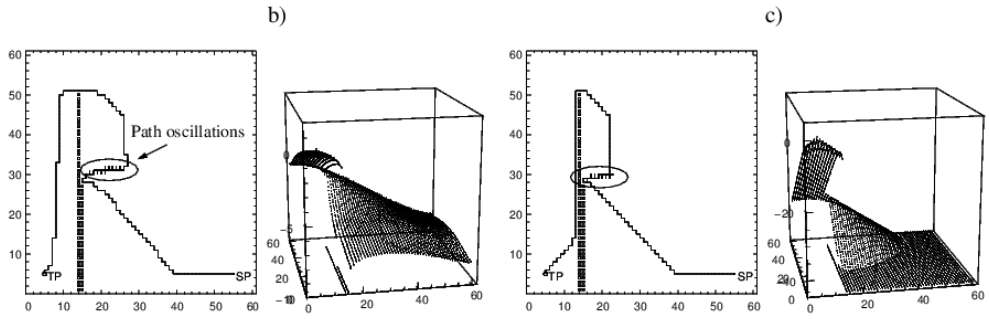

Abb. 6 resistive Grid & Glasius et al.

Network	path steps	network iterations
DWENN	142	283
resistive grid	464	560
Glasius et al.	274	380
Yang et al.	152	743
Chen et al.	270	412

Vergleicht man die Ergebnisse, so sticht das DWENN in dieser 'Disziplin' deutlich heraus. Es hat mit Abstand die wenigsten Netzwerkiterationen und auch der gewählte Pfad ist am kürzesten. Die Unterschiede des Pfades erkennt man durch Vergleich der Bilder [Abb. 5 & 6], die zeigen das konkurrierende Systeme nach dem Schließen des Gates oszillieren, bis sie das Problem erkannt haben und letzlich den Weg finden.

- **'Freezing Up Obstacles':**

Bei diesem Beispiel werden bewegliche Hindernisse auf und ab bewegt, ehe sie nach 2 Iterationen stehen bleiben. Das Netz muss sich auf die neue Situation einstellen und von dem ursprünglich geplanten Weg abweichen.

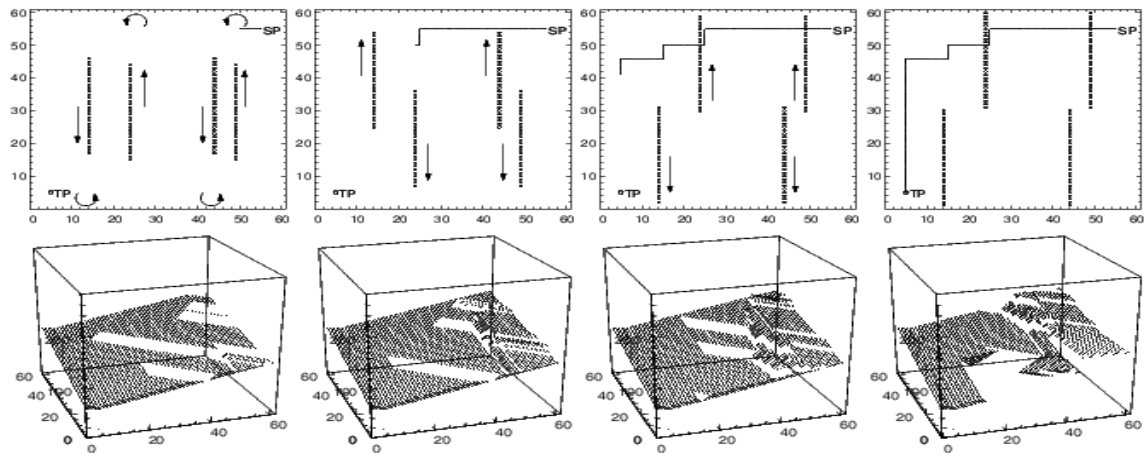

Abb. 7 DWENN: Gefundener Pfad und dazugehörige Potentialfelder

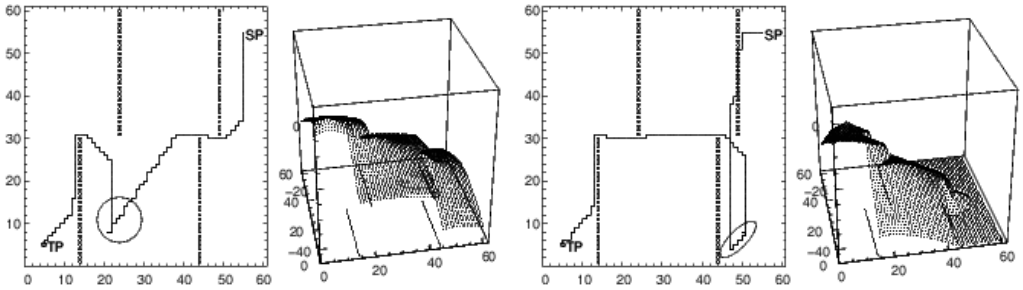

Abb. 8Yang et al. & Chen et al.

Network	path steps	network iterations
DWENN	100	216
resistive grid	250	1252
Glasius et al.	150	368
Yang et al.	150	652
Chen et al.	232	368

Auch bei dieser Aufgabe ist das DWENN beeindruckend stark im Vergleich mit den anderen Lösungsansätzen. Der gewählte Pfad ist der kürzeste, die Anzahl der Netzwerkdurchläufe zeigt jedoch erst deutlich die Dominanz des DWENN.

- **'Pursuit the moving Target':**

Auch das Finden eines beweglichen Ziels gehört zu dem Repertoire von Aufgaben, die einem Netz gestellt und zum Zwecke von Benchmarks verwendet werden.

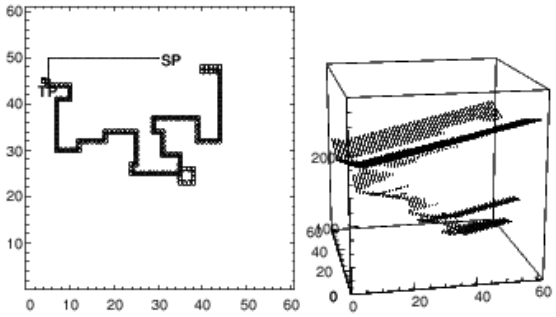

Abb. 9 DWENN: Gefundener Pfad und dazugehörige Potentialfelder

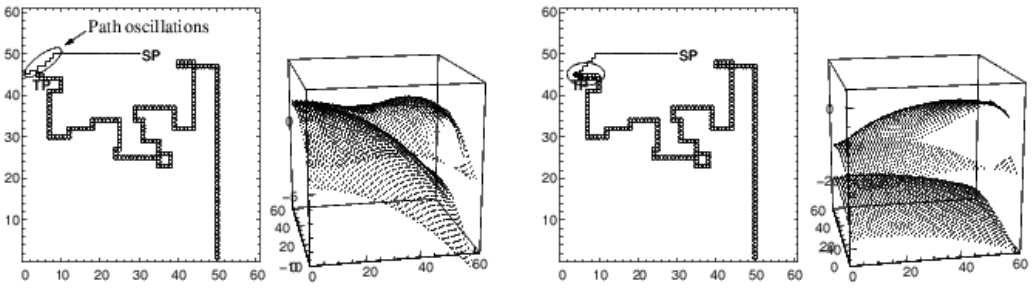

Abb. 10 resistive Grid & Glasius et al.

Da keine Daten über Netzwerkiterationen und die Länge des gefundenen Wegs vorliegen ist eine qualitative Aussage zu dieser Aufgabe nicht möglich. Es lässt sich lediglich feststellen, das die anderen Systeme um das Ziel oszillieren ehe sie es finden, es existiert ein System welches nicht in der Lage ist ein bewegtes Ziel zu finden.

Zusammenfassung und Ausblick

Zusammenfassend lässt sich über das DWENN sagen, dass es im Augenblick zu den leistungsfähigsten Systemen zur Berechnung zweidimensionaler Wege gehört. Es berechnet sehr effizient dynamische Abstandspotentiale, deren Gradient zum ausgewählten Ziel zeigt. Dabei ist keine a priori Kenntnis der Umgebung erforderlich, auch die Robustheit bei vielen Hindernissen, selbst wenn diese völlig neue Situationen schaffen ist an dieser Stelle herauszustellen.
Mit Blick auf mögliche Anwendungsgebiete sei hier auf multiagenten Umgebungen (z.B. Fussball) hingewiesen. Ein Einsatz in der Trajektorienplanung von Manipulatoren erfordert allerdings Eingriffe in den Rechenaufwand, weil diese Art von Robotern meist sehr hochdimensionale Arbeitsräume aufzuweisen hat.

Quellen

- Dmitry V. Lebedev, Jochen J. Steil, Helge J. Ritter (11.Oktober 2004), **The dynamic wave expansion neural network model for robot motion planning in time-varying environments**
- Dmitry V. Lebedev, Jochen J. Steil, Helge J. Ritter (2003), **Real-Time Path Planning in Dynamic Environments: a Comparison of Three Neural Network Models**
- http://www.techfak.uni-bielefeld.de/ags/ni/index_d.html