IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE PARA O ROBÔ PUMA 560 USANDO UMA REDE NEURAL AUTO-ORGANIZÁVEL

Guilherme de A. Barreto*
gbarreto@sel.eesc.sc.usp.br

Aluizio F. R. Araújo*
aluizioa@sel.eesc.sc.usp.br

Christof Dücker†
chrisd@techfak.uni-bielefeld.de

Helge Ritter†
helge@techfak.uni-bielefeld.de

*Departamento de Engenharia Elétrica, Universidade de São Paulo, C. P. 359, 13560-970, São Carlos, SP, Brasil

†Departamento de Ciência da Computação, Universidade de Bielefeld. P. O. 100131, 33501, Bielefeld, Alemanha

ABSTRACT

In this work it is proposed a self-organizing network, called Competitive and Temporal Hebbian (CTH) network, capable of learning and recalling complex temporal sequences. The CTH network handles sequences in which an item occurs many times or is shared with other sequences. In both cases, uncertainties occur during recall, but context information are used to resolve them. Competitive synaptic weights encode the static portion of a sequence, while the temporal order is encoded by lateral weights. The CTH network saves memory space since only a single copy of each repeated/shared sequence item is stored. Furthermore, a redundancy mechanism improves the robustness of the network against noise and faults. A distributed control platform was used to evaluate the CTH network in trajectory planning for real time, point-to-point control of trajectories of a PUMA 560 robot. The proposed system is compared with other neural network based approaches.

KEYWORDS: Unsupervised neural networks, temporal sequences, robotics, trajectory planning, distributed control.

RESUMO

Neste trabalho é proposta uma rede neural auto-organizável, chamada rede Competitiva e Hebbiana Temporal (CHT), capaz de aprender e reproduzir seqüências temporais complexas. Tais seqüências possuem elementos repetidos e/ou compartilham elementos com outras seqüências. Em ambos os casos ocorrem incertezas durante a fase de reprodução das seqüências armazenadas, sendo estas resolvidas por meio de informação de contexto. Pesos sinápticos competitivos codificam a parte estática das seqüências, enquanto a ordem temporal é codificada através de conexões laterais. A rede CHT faz uso eficiente de memória, pois armazena apenas uma única cópia de cada elemento repetido/compartilhado. Além disso, redundância no armazenamento dos elementos de uma seqüência a torna tolerante a ruídos e falhas. Foi utilizada uma plataforma de controle distribuído para avaliar a rede CHT no problema de planejamento de trajetórias para controle ponto-a-ponto, em tempo real, do robô PUMA 560. A performance do sistema de controle é comparada com a de outras redes neurais.

PALAVRAS-CHAVE: Redes não-supervisionadas, seqüências temporais, robótica, planejamento de trajetória, controle distribuído.

Artigo submetido em 20/12/00
1a. Revisão em 07/06/01; 2a. Revisão em 05/10/01
Aceito sob recomendação do Ed. Assoc. Prof. Paulo E. Miyagi

1 INTRODUÇÃO

A informação proveniente do ambiente externo e que se projeta sobre os órgãos sensoriais de um indivíduo possui duas dimensões: uma de natureza *estática* (ou *espacial*) que permite determinar a posição de um dado objeto naquele ambiente; outra de natureza *dinâmica* (ou *temporal*) que possibilita inferir se tal objeto está em movimento, bem como sua direção e sentido. Dificilmente é possível tratar tais dimensões em separado de modo que elas são comumente descritas como uma única entidade denominada *seqüência espaço-temporal* (SET). A capacidade de processar tais seqüências é de fundamental importância para a correta realização de tarefas consideradas rotineiras, como aquelas relacionadas à movimentação de braços e pernas.

Redes neurais artificiais (RNAs) vêm sendo utilizadas há algum tempo na modelagem matemática e computacional de diversos aspectos relacionados com o processamento de SETs (Amari, 1972). Contudo, a imensa maioria das RNAs de uso corrente não foi desenvolvida para lidar com padrões temporais. Os poucos exemplos que o fazem são baseados em Perceptrons multicamadas treinados com versões temporais do algoritmo *backpropagation*, na rede de Hopfield ou em redes neurais recorrentes (Wang, 1995). Mais recentemente, RNAs auto-organizáveis têm sido propostas com o intuito de processar padrões temporais e, em especial, a área de aprendizagem de robôs tem recebido contribuições relevantes (Heikkonen e Koikkalainen, 1997; Billard e Hayes, 1998; Barreto e Araújo, 1999a; Jockusch, 2000; Barreto e Araújo, 2001a).

A dinâmica de RNAs auto-organizáveis (ou não-supervisionadas) evolui dependendo unicamente dos estímulos oriundos do ambiente externo. À medida que estes estímulos, sejam eles seqüênciais ou não, vão sendo apresentados à rede, espera-se que ela extraia *por si só* algum tipo de regularidade a partir desses estímulos. Em geral, a informação aprendida é codificada na forma de estados de equilíbrio estável (padrões estáticos) ou seqüências de transições de estados (padrões dinâmicos). Uma vez que um dado estímulo é armazenado na rede, ele poderá ser evocado a qualquer instante, mesmo quando um estímulo ligeiramente distorcido é apresentado à rede. De forma semelhante, se uma seqüência de estímulos é adequadamente codificada pela rede, sua posterior reprodução pode ser iniciada fornecendo à rede um único estímulo que dispare a evocação dos estímulos subseqüentes que completam a seqüência codificada.

O princípio básico usado na modelagem de SETs é conhecido como *hipótese do encadeamento temporal* (temporal chaining hypothesis) e tem por objetivo obter uma representação da ordem temporal dos elementos de uma SET. Segundo esta hipótese, uma SET é vista uma cadeia causal de eventos conectados no tempo, na qual o conjunto de associações (transições) entre eventos consecutivos deve ser aprendida (codificada) para uma posterior reprodução total ou parcial da seqüência. A implementação desta idéia requer a definição de dois mecanismos de memória. O primeiro deles, chamado de *memória de curta duração* (MCD), permite extrair e armazenar as dependências temporais entre os elementos de uma seqüência (Barreto e Araújo, 2001b). Na MCD, os elementos da seqüência são representados de maneira que o padrão de ativação dos neurônios da rede codifique tanto os itens em si, quanto a ordem em que eles ocorreram. A informação armazenada na MCD é transitória e se perde rapidamente com o tempo. Daí, é necessário a definição de um segundo mecanismo, denominado *memória de longa duração* (MLD), que transfere a informação da MCD para os pesos sinápticos da rede. Uma correta interação entre a MCD e a MLD assegura que a dinâmica da rede reproduza as seqüências armazenadas na ordem temporal correta e na precisão desejada.

Neste artigo se está particularmente interessado na aprendizagem rápida e precisa de SETs complexas, representando trajetórias de um robô manipulador. Deseja-se também enfatizar a aplicação de redes auto-organizáveis no campo da robótica através: (*i*) do desenvolvimento de uma rede auto-organizável que utiliza informação de contexto temporal para processar SETs complexas, (*ii*) da proposição de uma regra de aprendizagem hebbiana que codifique a ordem temporal dos elementos de uma trajetória, e (*iii*) da implementação de um sistema de controle distribuído para um robô real baseado na rede proposta. Serão consideradas trajetórias com a forma de oito e também linhas retas. A performance da rede ao armazenar e reproduzir com precisão e sem ambigüidades as trajetórias será avaliada, bem como sua robustez à ruídos e falhas.

O restante do artigo está organizado como segue. Na Seção 2, o uso de redes auto-organizáveis convencionais (não-temporais) em robótica é brevemente discutido. Na Seção 3, um algoritmo de rede neural auto-organizável temporal é apresentado. Na Seção 4, uma plataforma de controle robótico distribuído e seus principais componentes são mostrados. Na Seção 5, a performance da rede é conferida via simulação e testes com o robô real. Os principais resultados são discutidos na Seção 6 e comparados com os de outras redes. O artigo é concluído na Seção 7.

2 REDES NEURAIS E ROBÓTICA

A pesquisa em RNAs tem se apresentado como uma alternativa viável para a solução de problemas complexos no campo da robótica (Prabhu e Garg, 1996). Em particular, redes neurais auto-organizáveis possuem características atraentes para uso em tal campo: (*i*) supervisão externa do aprendizado é mínima; (*ii*) em geral, seu treinamento é rápido, o que as habilita para aplicações em tempo real; e (*iii*) a informação é representada de maneira localizada, facilitando a interpretação dos resultados. É possível mostrar que tais características reduzem significativamente a carga envolvida na programação de robôs, item responsável por até um terço do custo total de implantação de um sistema robótico industrial (Heikkonen e Koikkalainen, 1997).

Tanto em sistemas biológicos quanto em sistemas robóticos, o controle de movimentos envolve diversas *transformações sensório-motoras* (sensorimotor transformations). Estas fazem a conversão de sinais provenientes de várias fontes sensoriais (visão, tato, etc.) em comandos motores que acionam um conjunto de músculos ou atuadores robóticos (Massone, 1995). Tais transformações são não-lineares e, em geral, de difícil representação matemática. RNAs podem ser utilizadas para aprender uma ou várias transformações sensório-motoras que sejam adequadas a uma determinada tarefa robótica, mesmo sem conhecimento preciso dos parâmetros do robô. Com este propósito, os padrões para treinamento da rede neural são comumente gerados da seguinte forma: um certo robô realiza movimentos aleatórios dentro de seu espaço de trabalho e, para cada movimento executado, os valores obtidos para as variáveis sensoriais de interesse são medidos. Este procedimento de ação-percepção, conhecido como *reação circular sensório-motora* (Gaudiano e Grossberg, 1991), permite que a rede aprenda com a experiência a realizar movimentos precisos. Em essência, o que se aprende é uma espécie de associação entre as ações motoras executadas aleatoriamente e suas respectivas conseqüências sensoriais. Após o treinamento da rede, a transformação pode ser utilizada para fins de controle, ou seja, toda vez que o robô experimentar um dado estímulo sensorial, a rede proverá o comando motor correspondente.

Em sistemas de controle, a técnica descrita no parágrafo anterior recebe o nome de *controle inverso direto* (direct inverse control) (Prabhu e Garg, 1996). *Aleatoriedade* é a principal característica deste método pois os movimentos sucessivos do robô são supostos independentes entre si. Ou seja, a ordem temporal em que movimentos sucessivos acontecem não é importante. Esta aleatoriedade permite que redes auto-organizáveis convencionais (ou seja, estáticas) sejam utilizadas para aprender transformações sensório-motoras, visto que o treinamento dessas redes é baseado na suposição de independência estatística entre os padrões de treinamento. Esta abordagem tem sido largamente empregada para controlar manipuladores robóticos e robôs móveis. Em manipuladores, ela pode ser usada para aprender cinemática inversa (Martinetz *et al.*, 1990; Gaudiano e Grossberg, 1991; Walter e Schulten, 1993).

Contudo, muitas tarefas robóticas possuem uma natureza seqüencial bem definida pela ordem das posições que um determinado robô assume, sucessivamente no tempo, ao longo de um caminho preestabelecido. Em geral, esta característica temporal não é incorporada no procedimento de aprendizagem de RNAs estáticas, de forma tal que apenas transformações sensório-motoras estáticas (cinemática inversa, por exemplo) podem ser aprendidas. Neste caso, a ordem temporal da tarefa robótica é estabelecida de antemão pelo projetista da RNA. Uma alternativa interessante consiste em usar *RNAs temporais*, visto que elas incorporam automaticamente os aspectos seqüências da tarefa robótica durante a fase de treinamento. Durante a fase de reprodução, um estado inicial qualquer do robô é fornecido como entrada para a RNA temporal que, por sua vez, fornece como saída o próximo estado a ser alcançado pelo robô e assim sucessivamente até que toda a trajetória memorizada pela rede seja reproduzida (Barreto e Araújo, 1999a).

Um problema que se enquadra na categoria descrita no parágrafo anterior e que tem recebido recentes contribuições da área de redes neurais é o de aprendizagem de seqüências para planejamento e controle ponto-a-ponto de trajetórias de robôs (Althöfer e Bugmann, 1995; Bugmann *et al.*, 1998; Barreto e Araújo, 1999a; Barreto e Araújo, 2001a). Esta tarefa converte uma descrição prévia do movimento desejado em uma trajetória definida como uma seqüência temporal de estados do robô entre um ponto de origem e um de destino. O robô deve seguir o caminho preestabelecido, tal que seus controladores coordenem o movimento de modo a realizá-lo com precisão, ponto-a-ponto, ao longo do caminho especificado. Nas redes propostas por Althöfer e Bugmann (1995) e Bugmann *et al.* (1998), a ordem temporal dos estados de uma trajetória não são aprendidas pela rede, mas sim estabelecidas previamente pelo usuário da rede. Para trajetórias com estados repetidos e/ou compartilhados (*trajetórias complexas*), a rede de Althöfer e Bugmann (1995) não consegue reproduzir corretamente as trajetórias armazenadas. Já a rede de Bugmann *et al.* (1998) reproduz trajetórias com estados repetidos, mas não é capaz de lidar com

várias trajetórias que compartilhem estados com outras.

Barreto e Araújo (1999a) propuseram uma rede auto-organizável na qual a ordem temporal dos itens da seqüência é aprendida por meio de uma regra de aprendizagem muito semelhante à descrita neste artigo (ver Eq. (9)). Contudo, esta rede aprende e reproduz apenas trajetórias que, quando tomadas isoladamente não possuem estados repetidos, mas quando tomadas em conjunto compartilham estados entre si. Além disso, uma cópia do estado compartilhado é criada e armazenada por um neurônio diferente toda vez que esse estado ocorrer. Uma extensão dessa rede, proposta por Barreto e Araújo (2001a), é capaz de aprender trajetórias contendo simultaneamente estados repetidos e compartilhados, mas continua mantendo várias cópias dos estados repetido/compartilhado na memória. A rede a ser proposta a seguir é uma extensão desses dois trabalhos, tornando possível processar trajetórias contendo tanto estados repetidos quanto estados compartilhados, ao mesmo tempo que mantém armazenada apenas uma única cópia de cada estado repetido/compartilhado. O objetivo desta rede é armazenar diversas trajetórias robóticas complexas que serão posteriormente usadas no controle ponto-a-ponto de um robô manipulador industrial.

3 ARQUITETURA DA REDE NEURAL

A arquitetura neural utilizada neste trabalho, chamada de rede *Competitiva e Hebbiana Temporal* (CHT), foi proposta por Barreto e Araújo (2000) e está mostrada na Fig. 1. Esta rede consiste em conexões de propagação para frente e conexões laterais que desempenham papéis distintos na sua dinâmica. Ela possui também dois tipos de unidades de contexto na entrada e atrasadores na saída. Contudo, os atrasadores são necessários apenas na fase de treinamento para a aprendizagem de transições de estado. As variáveis $\mathbf{a}(t)$ e $\mathbf{y}(t)$ simbolizam, respectivamente, os vetores de ativação e de saída dos neurônios da rede no instante de tempo t, enquanto $\mathbf{a}(t-1)$ e $\mathbf{y}(t-1)$ são os vetores de ativação e de saída no instante anterior. Mais adiante é mostrado como calcular $\mathbf{a}(t)$ e $\mathbf{y}(t)$ através das Eqs. (4) e (10).

A entrada da rede CHT consiste em unidades sensoriais $\mathbf{s}(t)$, unidades de contexto global $\mathbf{C}^g(t)$ e unidades de contexto local $\mathbf{C}^l(t)$. Unidades sensoriais recebem o estado da trajetória no instante t e o propaga em direção à saída. No problema de robótica estudado aqui, $\mathbf{s}(t) = \{\mathbf{z}(t), \boldsymbol{\theta}(t), \boldsymbol{\tau}(t)\}$, onde $\mathbf{z}(t) \in \Re^3$ é a coordenada cartesiana do efetuador do robô com relação a um sistema de coordenadas com origem na base, $\boldsymbol{\theta}(t) \in \Re^{dof}$ é

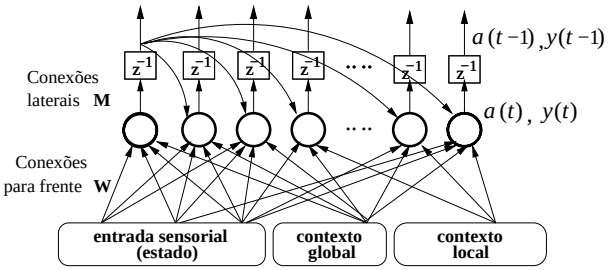

Figura 1: A arquitetura da rede CHT.

o vetor de ângulos das juntas que produz a posição $\mathbf{z}(t)$, e $\boldsymbol{\tau}(t) \in \Re^{dof}$ é o vetor de torques aplicados nas juntas para movê-las da sua posição inicial para a posição especificada por $\boldsymbol{\theta}(t)$. A dimensão de $\mathbf{s}(t)$ depende do número de graus de liberdade, dof, do robô.

O robô PUMA 560 adotado neste trabalho possui 6 graus de liberdade, sendo os três primeiros relativos à rotação da base, do ombro e do cotovelo, e os três últimos referentes à orientação (*roll*,*pitch*, *yaw*) do efetuador. Para evitar problemas de existência e unicidade de soluções da cinemática e dinâmica inversas, os valores que os ângulos das juntas assumem ao longo de uma trajetória são escolhidos (ou medidos) pelo operador do robô e os valores correspondentes da posição cartesiana do efetuador são calculados por meio das equações de cinemática direta, enquanto os torques a serem aplicados nas juntas são determinados por meios das equações de dinâmica direta. A cinemática e a dinâmica diretas garantem uma única solução para a posição cartesiana e para os torques das juntas, dados os ângulos das juntas. Estes cálculos são facilmente realizados por meio de várias ferramentas computacionais que auxiliam na modelagem e controle de robôs manipuladores, tais como o pacote RCCL/RCI (Robot Control C Library/Real-time Control Interface) (Lloyd *et al.*, 1988) e a toolbox *Robotics* para Matlab (Corke, 1996) (ambas contém especificações dos parâmetros cinemáticos e dinâmicos do robô PUMA 560).

Deve-se enfatizar que, apesar da escolha do robô PUMA para os testes a serem mostrados neste artigo, a aplicação da rede CHT na aprendizagem de trajetórias robóticas é independente do manipulador utilizado. Tão logo um certo número de trajetórias está disponível, o processo de aprendizagem pode ser iniciado, mesmo que os parâmetros do robô envolvido sejam desconhecidos. Esta última situação é possível porque as relações matemáticas entre as variáveis $(\mathbf{z}(t), \boldsymbol{\theta}(t), \boldsymbol{\tau}(t))$ são aprendidas *implicitamente* quando o estado $\mathbf{s}(t)$ é formado e armazenado na rede. Este é o mesmo mecanismo de

memória associativa da reação circular sensório-motora discutido na Seção 2, usado por muitas redes neurais não-supervisionadas quando aplicadas em robótica, e que vem sendo utilizado na aprendizagem de transformações cinemáticas e dinâmicas, tanto diretas quanto inversas, de um determinado manipulador (Martinetz *et al.*, 1990; Walter e Schulten, 1993).

Unidades de contexto são de dois tipos, *global* e *local*, e desempenham um papel essencial na resolução de ambigüidades durante a reprodução de trajetórias complexas. O contexto global é invariante no tempo e seu valor é feito igual ao de um estado qualquer da trajetória, sendo o estado final (posição-alvo) a opção usual. O contexto global atua como um tipo de identificador da trajetória a ser memorizada. Essa informação global é fundamental quando várias trajetórias que compartilham estados entre si têm que ser armazenadas para posterior reprodução. Se estas trajetórias, quando tomadas individualmente, não possuem elementos repetidos, então é possível mostrar que apenas o contexto global é suficiente para reproduzi-las na ordem temporal correta (Barreto e Araújo, 1999a; Barreto e Araújo, 1999b).

Quando uma trajetória contém estados repetidos, estes terão o mesmo contexto global já que pertencem à mesma trajetória. Assim, informação adicional é necessária para resolver ambigüidades durante a fase de reprodução da trajetória em questão. Estas ambigüidades resultam da incapacidade da rede em decidir, com base apenas na informação de contexto global, qual o próximo estado da trajetória a ser reproduzido quando a rede atinge a primeira ocorrência do estado repetido. Por exemplo, na seguinte seqüência hipotética, **a**→**b**→**c**→**d**→**e**→**c**→**f**, o elemento **c** ocorre 2 vezes. Ao atingir o primeiro **c** a rede não é capaz de decidir se o próximo elemento da seqüência é o elemento **d** ou o elemento **f**. Para resolver estes casos, informação de contexto local, variante no tempo, é adicionada à entrada da rede. Este tipo de contexto é formado por estados anteriores da trajetória que precedem o estado atual: $\mathbf{C}^l(t) = \{\mathbf{s}(t-1), \mathbf{s}(t-2), \ldots, \mathbf{s}(t-T)\}$, T é chamada de *profundidade da memória* (memory depth). O contexto local confere à rede a capacidade de decidir qual o próximo estado da trajetória a ser reproduzido com base no histórico dos elementos que precedem cada ocorrência do estado repetido.

A rede CHT possui dois conjuntos de conexões sinápticas: pesos de propagação para frente, $\mathbf{w}_j(t) = [\mathbf{w}_j^s(t), \mathbf{w}_j^g(t), \mathbf{w}_j^l(t)]$, onde $j = 1, \ldots, M$ e pesos laterais $\mathbf{m}_j(t)$. Para cada estado apresentado à rede, os pesos de propagação para frente são ajustados em primeiro lugar, para depois os pesos laterais serem modificados. Os pesos de propagação para frente armazenam os estados da trajetória, enquanto os pesos laterais codificam a ordem temporal em que os estados da trajetória se sucedem. Detalhes do algoritmo de aprendizagem são dadas a seguir.

3.1 Regras de Seleção e Ativação

Uma trajetória é apresentada uma única vez à rede CHT, estado por estado. Se esta trajetória contém N_c estados, serão necessários N_c passos de treinamento. Os mesmos N_c estados deverão ser reproduzidos posteriormente, toda vez que aquela trajetória for requisitada durante a execução de determinada tarefa robótica. Para cada estado apresentado à rede, a parte sensorial $\mathbf{s}(t)$ do vetor de entrada é comparada com a parte correspondente nos pesos de propagação para frente via distância euclideana:

$$D_j^s(t) = \|\mathbf{s}(t) - \mathbf{w}_j^s(t)\| \qquad (1)$$

onde $D_j^s(t)$, chamada de *distância sensorial*, é utilizada na determinação dos vencedores da competição atual. O processo de competição entre os neurônios envolve dois passos: (*i*) a comparação do estado da trajetória atualmente na entrada da rede com os vetores de pesos correspondentes (ver Eq. 1), e (*ii*) a determinação daqueles neurônios cujos vetores de pesos estão mais próximos de $\mathbf{s}(t)$ (ver Eqs. (2) e (3) a seguir). Define-se ainda uma *distância de contexto global*, $D_j^g(t)$, e uma *distância de contexto local*, $D_j^l(t)$:

$$D_j^g(t) = \|\mathbf{C}^g(t) - \mathbf{w}_j^g(t)\| \quad e \quad D_j^l(t) = \|\mathbf{C}^l(t) - \mathbf{w}_j^l(t)\|$$

As distâncias $D_j^g(t)$ e $D_j^l(t)$ são utilizadas para resolver ambigüidades durante a fase de reprodução.

É importante notar que redes neurais competitivas tendem a agrupar os padrões de entrada com base apenas em semelhanças espaciais (*clustering*) e, por isso, não costumam ser usadas na reprodução de trajetórias. Para evitar tal situação, a rede CHT "penaliza" todo neurônio vencedor, excluindo-o de competições subseqüentes, evitando que ele tente codificar mais de um estado da trajetória (Barreto e Araújo, 1999a). Este mecanismo de exclusão é implementado definindo-se uma função $R_j(t)$, chamada *função responsabilidade*, que indica se um dado neurônio já é responsável pelo armazenamento de um estado da trajetória. Se $R_j(t) > 0$, o neurônio j é excluído da competição atual e das subseqüentes. Se $R_j(t) = 0$, o neurônio j pode participar da competição atual.

De acordo com o mecanismo de exclusão, se um estado da trajetória ocorrer novamente, ele será codificado por outro neurônio e não mais por aquele que o armazenou antes, já que este foi excluído das competições. Portanto, várias cópias do *mesmo* estado existirão na rede.

Para aumentar a eficiência no uso de memória, toda vez que um estado ocorresse, ele deveria ser codificado pelo neurônio que o armazenou previamente mesmo que ele tenha sido excluído das competições. Esta heurística pode ser implementada definindo-se uma constante $0 < \varepsilon \ll 1$, chamada *raio de similaridade*. Esta constante define uma região em torno de $\mathbf{s}(t)$ dentro da qual se pode considerar o vetor $\mathbf{w}_j^s(t)$ suficientemente similar a $\mathbf{s}(t)$.

A ação conjunta do mecanismo de exclusão e do raio de similaridade resulta no seguinte fato: estados *diferentes* serão armazenados por neurônios *diferentes* (mecanismo de exclusão), enquanto ocorrências repetidas de um estado serão sempre armazenadas pelo neurônio que armazenou a primeira ocorrência deste (mecanismo de similaridade). Pode-se concluir, portanto, que cada neurônio armazenará um e somente um único estado da trajetória. Este comportamento pode ser formalizado em uma função $f_j(t)$, chamada *função escolha*:

$$f_j(t) = \begin{cases} D_j^s(t) & \text{Se } D_j^s(t) \leq \varepsilon \text{ ou } R_j(t) = 0 \\ R_j(t) \cdot D_j^s(t) & \text{Caso contrário} \end{cases} \tag{2}$$

Em seguida, os neurônios de saída são ordenados de acordo com o valor da função escolha:

$$f_{\mu_1}(t) < f_{\mu_2}(t) < \cdots < f_{\mu_N}(t) \tag{3}$$

onde $\mu_i(t)$, $i = 1, \ldots, N$, correspondem aos índices (posições) dos neurônios na rede. Destes, são escolhidos K neurônios, $\{\mu_1(t), \mu_2(t), \ldots, \mu_K(t)\}$, $K \ll N$, como os vencedores da presente competição. Assim, para cada trajetória apresentada, esta e um número adicional de $K-1$ cópias (trajetórias redundantes) são armazenadas. Por este motivo, a constante K é chamada de *grau de redundância*, conferindo à rede uma certa tolerância a falhas e ao ruído. Conforme será visto adiante, as trajetórias redundantes não são exatamente iguais à original, sendo levemente alteradas para que a rede possa lidar com padrões de entrada ligeiramente distorcidos por ruído.

Os valores de ativação correspondentes decaem linearmente de um valor máximo $a_{max} \in \Re$ atribuído a $\mu_1(t)$, para um mínimo $a_{min} \in \Re$ atribuído a μ_K, de acordo com a seguinte equação:

$$a_{\mu_i} = \begin{cases} a_{max} - \left(\frac{a_{max} - a_{min}}{\max(1, K-1)}\right)(i-1), & \text{for } i \leq K \\ 0, & \text{for } i > K \end{cases} \tag{4}$$

onde os valores de a_{max} e a_{min} são especificados pelo usuário da rede. Para $t = 0$, $a_j(0) = 0$, para todo j. A função responsabilidade $R_j(t)$ é atualizada toda vez que um novo padrão de ativação é calculado:

$$R_j(t+1) = R_j(t) + \beta a_j(t) \tag{5}$$

onde $\beta \gg 0$ é chamada *constante de exclusão*. Para $t = 0$, $R_j(0) = 0$ para todo j.

3.2 Regras de Aprendizagem

Após a seleção dos K neurônios vencedores da competição atual e a determinação de suas ativações, o vetor de pesos $\mathbf{w}_j(t)$ é ajustado de acordo com as seguintes regras de aprendizagem:

$$\mathbf{w}_j^s(t+1) = \mathbf{w}_j^s(t) + \eta a_j(t)[\mathbf{s}(t) - \mathbf{w}_j^s(t)] \tag{6}$$
$$\mathbf{w}_j^g(t+1) = \mathbf{w}_j^g(t) + \eta a_j(t)[\mathbf{C}^g(t) - \mathbf{w}_j^g(t)] \tag{7}$$
$$\mathbf{w}_j^l(t+1) = \mathbf{w}_j^l(t) + \eta a_j(t)[\mathbf{C}^l(t-1) - \mathbf{w}_j^l(t)] \tag{8}$$

onde $0 < \eta \leq 1$ é a taxa de aprendizagem. Para $t = 0$, $\mathbf{w}_j(0)$ recebe valores aleatórios entre 0 e 1. De acordo com a Eq. (10), o estado atual da trajetória, $\mathbf{s}(t)$, e os vetores de contexto $\mathbf{C}^g(t)$ e $\mathbf{C}^l(t-1)$ são armazenados respectivamente nos vetores de pesos $\mathbf{w}_{\mu_i}^s(t)$, $\mathbf{w}_{\mu_i}^g(t)$ e $\mathbf{w}_{\mu_i}^l(t)$, dos K vencedores, porque apenas eles têm ativações não-nulas. A intensidade do ajuste dos pesos é determinada pelo *ranking* mostrado na Expressão (3) e que está refletido no padrão de ativação na Eq. (4). Case se adote $\eta = 1$, cada trajetória é aprendida muito rapidamente, necessitando apenas de uma única apresentação de seus estados durante o treinamento para que ela seja memorizada.

É importante notar que a Eq. (8) utiliza informação de contexto local referente ao instante imediatamente anterior ao instante atual t. Isto é necessário para a rede aprender relações de dependência temporal entre os instante de tempo atual, t, e o instante anterior, $t-1$. Esta informação será utilizada durante a reprodução das trajetórias armazenadas para resolver qualquer ambigüidade decorrente da presença de itens repetidos/compartilhados nas trajetórias. Contudo, as Eqs. (6), (7) e (8) sozinhas não provêem à rede CHT informação suficiente sobre a ordem temporal dos estados dentro da trajetória. Para que a reprodução da trajetória seja possível, a rede CHT deve ser capaz de representar a ordem temporal da trajetória como uma cadeia de transições de estado que correspondem à mudança de um estado da seqüência para outro (hipótese do encadeamento temporal). Para este fim, os estados individuais da ttrajetória ou melhor dizendo, os neurônios que os armazenaram, são associados na ordem temporal correta ajustando os pesos sinápticos laterais através da

sequinte regra de aprendizagem:

$$m_{jr}(t{+}1) = \begin{cases} m_{jr}(t) & \text{Se } m_{jr}(t) \neq 0 \\ m_{jr}(t) + \lambda a_j(t) a_r(t-1) & \text{Caso contrário} \end{cases} \quad (9)$$

onde $0 < \lambda \leq 1$ é a taxa de aprendizagem dos pesos laterais. Segundo a Eq. (9), a rede CHT "olha" um passo de tempo para trás de modo a estabelecer um *link causal* referente à transição temporal $\mathbf{s}(t-1) \rightarrow \mathbf{s}(t)$, entre dois estados consecutivos da trajetória. Por este motivo, pode-se afirmar que a Eq. (9) é uma regra de aprendizagem do tipo hebbiana (Hebb, 1949), porém de natureza temporal. Cada transição fica representada pelo peso lateral que conecta os neurônios que geraram os pares de ativação no instante t e $t-1$, ou seja, $[a_{\mu_i}(t-1), a_{\mu_i}(t)]$, $i \leq K$ (Fig. 2). A aplicação sucessiva da Eq. (9) leva à codificação da ordem temporal da trajetória. Inicialmente, $m_{jr}(0) = 0$ para todo j, r, indicando que nenhuma ordem temporal preestabelecida existe no começo do treinamento. É importante notar que uma conexão lateral $m_{jr}(t)$ é ajustada uma única vez para fins de normalização dos pesos laterais.

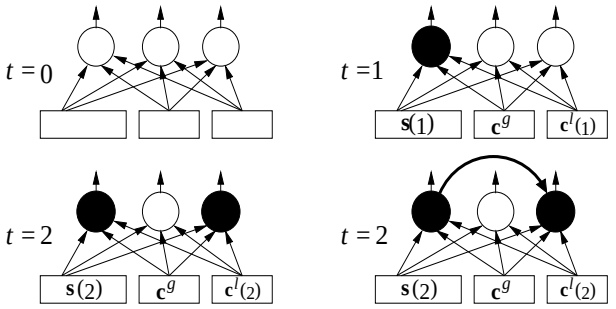

Figura 2: Neurônios vencedores em competições sucessivas sendo temporalmente ligados via conexões laterais. Para este exemplo, K=1.

3.3 Reprodução de Trajetórias

Uma vez que dada trajetória tenha sido memorizada, ela pode ser recuperada a partir de seu estado inicial ou de qualquer estado intermediário. O processo de reprodução é um esquema de controle em malha fechada (ver Fig. 3) consistindo de 5 passos: (1) início da reprodução, (2) cálculo das ativações neurais, (3) cálculo das saídas, (4) geração dos sinais de controle do robô, e (5) determinação das entradas sensoriais. Para fins de reprodução da trajetória, o parâmetro K é sempre igual a 1.

1. Início da Reprodução: Para iniciar a reprodução ($t = 0$ na Fig. 3), qualquer estado semelhante a um dos estados armazenados pode ser apresentado à rede CHT. Este estado inicial é chamado de *estado disparador* do processo de reprodução. O contexto global $\mathbf{C}_F$ assume o valor do último estado desta trajetória, enquanto que os valores iniciais do contexto local são feitos iguais ao estado disparador. A rede então se encarrega de reproduzir, de forma autônoma, o restante da trajetória ($t > 0$ na Fig. 3).

2. Cálculo das Ativações: Para cada estado $\mathbf{s}(t)$, a ativação do neurônio vencedor, $a_{\mu_1}(t)$, é calculada segundo a Eq. (4). Isto equivale a dizer que o vetor de pesos $\mathbf{w}^s_{\mu_1}(t)$ é o mais próximo de $\mathbf{s}(t)$.

3. Cálculo da Saída: O vencedor, o único neurônio de saída com ativação não-nula, ativará o neurônio cujo vetor de pesos armazenou o estado que vem em seguida ao estado da trajetória que está atualmente na entrada da rede. Isto é possível graças à transição de estado aprendida durante a fase de treinamento e codificada na conexão lateral que une estes dois neurônios. A equação de saída é definida como:

$$y_j(t) = y^c_j(t) \cdot g\left(\sum_{r=1}^{n} m_{jr}(t) a_r(t)\right) \quad (10)$$

onde $g(u) \geq 0$ e $dg(u)/du > 0$. Em geral, adota-se $g(u) = u$ para $u > 0$, e $g(u) = 0$, caso contrário. Para $t = 0$, $y_j(0) = 0$, para todo j. O fator $y^c_j(t)$ é definido em função das distâncias de contexto global e local:

$$y^c_j(t) = \left(1 - \frac{D^g_j(t)}{\sum_{r=1}^{n} D^g_r(t)}\right) \cdot \left(1 - \frac{D^l_j(t)}{\sum_{r=1}^{n} D^l_r(t)}\right) \quad (11)$$

Para trajetórias sem estados repetidos/compartilhados, o uso apenas do segundo fator no lado direito da Eq. (10) é suficiente para indicar o neurônio que armazenou o próximo estado da trajetória, ou seja, aquele com maior valor para $g(\sum_r m_{jr}(t) a_r(t))$ (Barreto e Araújo, 1999a). Para trajetórias com estados repetidos/compartilhados, informação adicional é necessária para resolver ambigüidades, pois o segundo fator no lado direito da Eq. (10) fornecerá o mesmo valor para todos os neurônios "candidatos" a conter o próximo estado. Possíveis ambigüidades são resolvidas pelo termo $y^c_j(t)$ mostrado na Eq. (11). O neurônio candidato com maior valor para $y^c_j(t)$, ou equivalentemente, com menores valores para $D^g_j(t)$ e $D^l_j(t)$, é considerado como aquele a ser escolhido, ou seja, aquele neurônio cujo contexto é o mais próximo do contexto do estado atualmente na entrada da rede.

4. Determinação dos Sinais de Controle: Os sinais de controle a serem enviados ao robô são calculados a partir do vetor de pesos do neurônio com maior valor de $y_j(t)$:

$$\mathbf{u}_{ctrl}(t) \equiv \mathbf{w}^s_{j^*}(t) \quad (12)$$

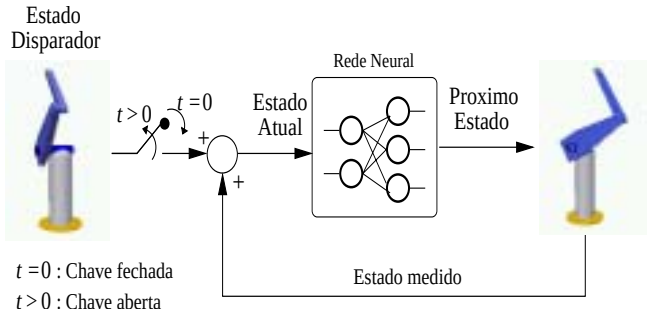

Figura 3: Esquema de controle neural autônomo em malha fechada.

onde $j^* = \arg\max_j[y_j(t)]$. Note que $\mathbf{u}_{ctrl}(t)$ fornece os valores de referência (*set-points*) para os ângulos das juntas, torques e posição cartesiana do efetuador.

5. Determinação de Novas Entradas Sensoriais: Um conjunto de medidas sensoriais fornece informação de realimentação sobre o estado atual do braço após a execução de um movimento. Quando o braço do robô atinge a posição especificada por $\mathbf{u}_{ctrl}(t)$, um novo vetor $\mathbf{s}$ é formado através das medidas sensoriais da posição atual do efetuador e dos ângulos das juntas correspondentes. Este novo vetor é então apresentado à rede CHT. Os passos 2-5 são executados várias vezes até que o final da trajetória tenha sido alcançado. Na próxima seção uma plataforma robótica é proposta com o intuito de utilizar a rede CHT no planejamento e controle ponto-a-ponto de trajetórias de um robô manipulador industrial.

4 PLATAFORMA ROBÓTICA

O sistema robótico usado para avaliar a rede CHT como planejador de trajetórias consiste essencialmente do *hardware*, que inclui o robô propriamente dito, o controlador do robô, uma estação de trabalho portadora do software de controle em baixo nível e um sistema de placas para interfaceamento do robô com a estação de trabalho; e o *software*, que abrange dois níveis de programação em alto nível: (1) a rede neural e (2) uma ferramenta de comunicação distribuída. Mais detalhes são dados a seguir.

4.1 Hardware e Software

O projeto de uma plataforma robótica para validação de algoritmos de redes neurais tem que superar algumas limitações dos controladores robóticos comercialmente disponíveis, tais como baixo poder computacional, limitada capacidade de expansão/atualização, pouca compatibilidade com outros sistemas e pouca transparência das suas linguagens de programação e sistemas operacionais. Assim, optou-se neste trabalho pela substituição de grande parte do hardware e, conseqüentemente, do software do controlador original do robô PUMA 560, empregando-se uma estação de trabalho Unix para controlar *diretamente* o robô em tempo real via um *link* de comunicação de alta velocidade. A estação de trabalho Unix, da marca SUN (modelo SPARC 4/370), está dotada de barramento VME e de placas para interfaceamento com o robô. O robô PUMA 560 é um manipulador de 6 graus de liberdade que está conectado ao controlador Unimation (Mark III) baseado na CPU LSI-11/73. Este, por sua vez, coordena vários controladores PID que acionam de maneira independente os servo-mecanismos associados a cada uma das 6 juntas rotacionais.

Em aplicações industriais o robô PUMA 560 é programado na linguagem interpretada VAL II. Tal linguagem não é capaz de processar dados de elevada largura de banda em tempo real, sendo por isso substituída pelo pacote RCCL/RCI (Robot Control C Library and Real-time Control Interface) (Lloyd *et al.*, 1988). Este consiste de um conjunto de bibliotecas que permite a um usuário experimentado requisitar a movimentação do robô a partir de um programa escrito em linguagem C. O controle do robô é realizado pelo RCCL/RCI por dois processos distintos que se comunicam entre si via memória compartilhada: *planejamento da trajetória* e *controle da trajetória*.

A tarefa de planejamento da trajetória consiste no programa do usuário. A tarefa de controle da trajetória consiste em um programa, executado periodicamente em prioridade alta, responsável pela leitura dos dados realimentados, pela geração de *setpoints* para as juntas, e pela execução de uma rotina de segurança (*watchdog function*) caso alguma das juntas atinja valores limites. Durante cada ciclo de controle (tipicamente 20 ms) um pacote de comandos é enviado ao controlador do robô pela porta paralela. O destino destes comandos é a CPU LSI-11, que é programada para despachar comandos para os atuadores das juntas, obter dados deles e checar valores limites. Quando da energização do sistema, o software de programação da CPU LSI-11 é carregado e iniciado pelo computador hospedeiro por meio de uma porta serial. Tal software permance respermanecea CPU e pode ser acessado pela porta paralela.

A plataforma descrita nos parágrafos anteriores foi originalmente proposta por Walter e Schulten (1993). Em relação ao software, o presente trabalho introduziu algumas novas idéias e melhorias. Mais especificamente, uma interface mais amigável foi desenvolvida para facilitar o envio de sinais de controle para o robô e a leitura de

sinais de realimentação a partir dele. Em vez de programar nos níveis de planejamento e controle da trajetória utilizando diretamente as funções do pacote RCCL/RCI (que exige um elevado grau de conhecimento), o usuário inclui no programa em que a rede CHT é desenvolvida chamadas a funções específicas que realizam, de modo transparente para ele, o controle do robô. Além disso, o controle pode ser feito remotamente de forma distribuída a partir de um terminal conectado à rede local de computadores.

4.1.1 Ferramenta para Processamento Distribuído

Comumente, nos campos da inteligência artificial, reconhecimento de padrões e robótica, diferentes módulos (aplicações), projetados para executar uma tarefa específica, devem ser integrados. Cada um desses módulos tem uma estrutura de dados própria e analisa determinado tipo de padrão. Uma comunicação eficiente entre diferentes módulos é de crucial importância para o correto funcionamento do sistema de controle distribuído como um todo. Para este fim, um novo *framework* de comunicação, denominado *Distributed Applications Communication System* (DACS) foi desenvolvido por Fink *et al.* (1995).

A ferramenta DACS foi projetada com o intuito de integrar padrões heterogêneos e manipular diferentes estruturas de dados. Seu funcionamento é baseado na arquitetura *cliente-servidor* em que uma determinada máquina (servidor) gerencia os recursos do sistema, atuando como elemento centralizador e direcionador das requisições feitas por outras máquinas (clientes). Tanto a aplicação cliente quanto o servidor rodam localmente um programa, chamado DACS-daemon, responsável pela codificação/decodificação e endereçamento correto das requisições. Cada aplicação existente tem que se registrar no sistema com um nome único, que é passado imediatamente a um servidor de nomes, possibilitando que outras aplicações também possam requisitar seus serviços. Note que um dado computador pode ser cliente e servidor ao mesmo tempo; ou seja, ele pode servir ou ser servido por outros.

Para ter acesso a uma determinada aplicação disponível em qualquer outra máquina da rede local de computadores, um cliente realiza um procedimento de chamada remota (*remote procedure call*), que consiste em uma mensagem requisitando a utilização daquela aplicação. Caso a requisição seja aceita, um fluxo de dados síncrono bidirecional é estabelecido. Uma chamada síncrona bloqueia o processo requisitante (cliente) até que um sinal de realimentação do servidor retorne. A ferramenta DACS também permite a realização de chamadas assíncronas que não bloqueiam o processo requisitante. Porém, para o sistema de controle utilizado neste trabalho, apenas chamadas síncronas serão utilizadas visto que o sistema deve funcionar em tempo real.

Neste trabalho, a aplicação cliente é a rede CHT que envia, por meio de uma chamada remota síncrona, sinais de controle (ângulos das juntas, por exemplo) requisitando o deslocamento do robô para uma determinada posição. O servidor processa tal chamada, codificando-a para a sintaxe do pacote RCCL/RCI, envia a requisição para o pacote RCCL/RCI e este envia o sinal de controle para o robô. Completado o movimento, a nova posição do robô é lida pelos sensores e o caminho de volta é realizado. Ou seja, do robô para o pacote RCCL/RCI, deste para o servidor que traduz o sinal de realimentação para a sintaxe da rede CHT e envia o sinal de realimentação para a rede. Todo este processo é transparente para o usuário da rede, restando a ele(a) apenas o conhecimento da sintaxe da ferramenta DACS para efetuar a chamada remota. Detalhes da interação via DACS entre as duas aplicações (rede CHT e RCCL/RCI) durante uma chamada síncrona é mostrada na Fig. 4.

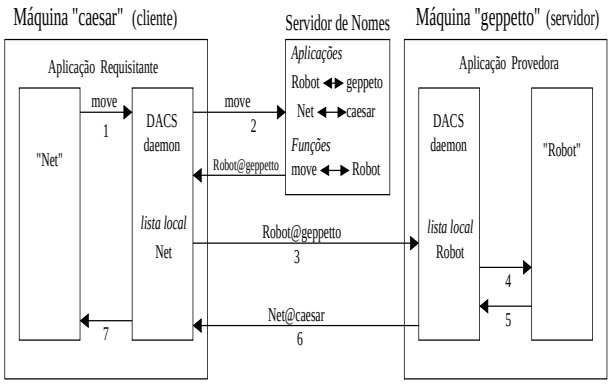

Figura 4: Interações durante uma chamada remota síncrona.

Nesta figura a aplicação denominada *robot* (RCCL/RCI) provê uma função *move*. A aplicação *net* (rede CHT) reside na máquina de nome *caesar*. Esta sabe apenas que existe um serviço proporcionado pela função *move* disponível em algum lugar no sistema. As aplicações e a função já estão registradas no servidor de nomes. Sete passos são realizados pela ferramenta DACS para executar uma chamada remota com origem em *net* requisitando a função *move*:

1. A aplicação *net* envia uma mensagem para o DACS-daemon local, endereçada a função *move*. A men-

sagem enviada é então rotulada como sendo do tipo "função" (necessário para encontrar a tabela de endereços de funções no servidor de nomes).

2. Usando o servidor de nomes, o DACS-daemon determina onde a função *move* está localizada. O resultado dessa busca é o endereço de rede da aplicação que registrou a função *move* no servidor de nomes: *robot@geppetto*, neste caso.

3. O DACS-daemon em *caesar* envia a mensagem para o DACS-daemon na máquina *geppetto*.

4. A aplicação *robot* é encontrada na tabela de aplicações locais, assim a mensagem é entregue para a aplicação que provê a função *move*. A ferramenta DACS decodifica o endereço e os argumentos da requisição e chama a função apropriada. Após o processamento da função requisitada, o resultado é recodificado e endereçado ao processo requisitante (cliente) – aqui *net@caesar*.

5. O resultado é entregue de volta ao DACS-daemon na máquina *geppetto*.

6. Este daemon entrega a mensagem diretamente ao daemon na máquina *caesar*. Note que neste ponto nenhuma chamada ao servidor de nomes é necessário pois o endereço do processo requisitante já é conhecido.

7. Finalmente, o daemon entrega a mensagem à aplicação *net* onde o resultado é decodificado e processado pela aplicação requisitante.

5 TESTES COM O SISTEMA PROPOSTO

Esta seção mostra diversos testes realizados com a rede CHT. Primeiro a rede é avaliada por simulação com o intuito de enfatizar algumas de suas principais propriedades. Em seguida, alguns testes usando a plataforma robótica proposta são apresentados.

5.1 Simulações com a Rede CHT

Nos testes a seguir, a rede proposta é avaliada por sua capacidade de aprendizagem e reprodução de diferentes trajetórias, tais como trajetórias abertas (linhas retas) e fechadas (na forma de oito), bem como na tolerância ao ruído e a falhas. As trajetórias de um robô PUMA 560 com $dof = 6$ graus de liberdade foram construídas usando-se a toolbox *Robotics* para Matlab (Corke, 1996). Cada estado $\mathbf{s}(t)$ de uma trajetória é formado pela posição espacial $\mathbf{z}(t)$ do efetuador do robô, pelos ângulos das juntas $\boldsymbol{\theta}(t) = \{\theta_1(t), \ldots, \theta_6(t)\}$ e pelos

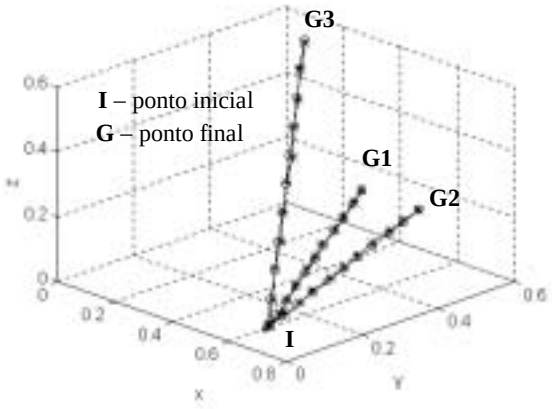

Figura 5: Reprodução das posições espaciais para três trajetórias abertas que tem o ponto inicial $(0,6\,m; 0,1\,m; 0,0\,m)$ em comum. Trajetórias reais são identificadas pelo símbolo 'o' e as reproduzidas por '*'.

torques aplicados nas juntas $\boldsymbol{\tau}(t) = \{\tau_1(t), \ldots, \tau_6(t)\}$, ou seja, $\mathbf{s}(t) = \{\mathbf{z}(t), \boldsymbol{\theta}(t), \boldsymbol{\tau}(t)\}$, $\mathbf{s}(t) \in \Re^{15}$. O efetuador do robô pode ser posicionado dentro dos limites de um cubo de dimensões máximas de $1\,m \times 1\,m \times 1\,m$.

O primeiro teste avalia a capacidade da rede em reproduzir três trajetórias em linha reta que têm o ponto inicial $(0,60\,m; 0,10\,m; 0,0\,m)$ em comum. O contexto global é fixado no estado final do efetuador do manipulador, i.e., $\mathbf{C}^g = \mathbf{s}^{final}$; logo, $\dim(\mathbf{C}^g) = 15$. O contexto local é escolhido como sendo de profundidade $T = 2$: $\mathbf{C}^l(t) = \{\mathbf{s}(t-1), \mathbf{s}(t-2)\}$. Portanto, $\dim(\mathbf{C}^l) = 2 \cdot 15 = 30$. Os parâmetros para todas as simulações que se seguem são $M = 250$, $\varepsilon = 10^{-5}$, $K = 2$, $a_{max} = 1$, $a_{min} = 0,98$, $\beta = 100$, $\eta = 1$, $\lambda = 0,8$. Após apresentar cada trajetória uma única vez para a rede CHT, a reprodução dessas trajetórias é realizada para verificar se elas foram aprendidas na ordem correta e com boa precisão. Os resultados na Fig. 5 ilustram um caso típico de reprodução correta. O simulador na Fig. 6 permite a visualização dos resultados para estas três trajetórias.

Para os próximos testes, o desempenho é medido pela raiz quadrada do erro quadrático médio de posicionamento do garra do robô:

$$RMSE = \sqrt{\frac{1}{N_c}\sum_{t=1}^{N_c} \|\mathbf{z}^{real}(t) - \mathbf{z}^{rep}(t)\|^2} \qquad [m] \quad (13)$$

onde N_c é o número de estados da trajetória sendo avaliada, $\mathbf{z}^{real}(t)$ é a posição cartesiana "real" do efetuador

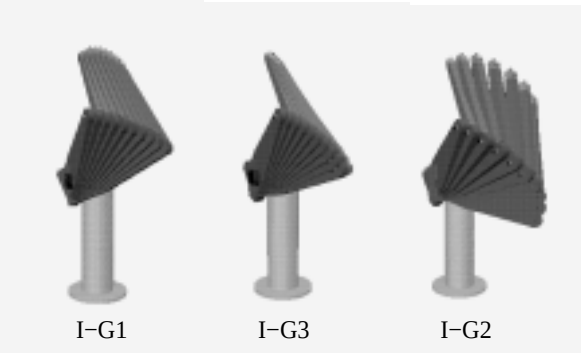

Figura 6: Visualização das trajetórias mostradas na Fig. 5.

e $\mathbf{z}^{rep}(t)$ é a posição do efetuador reproduzida pela rede CHT. Dados os estados inicial e final, avalia-se o efeito da variação do número de estados intermediários (taxa de amostragem) de uma dada trajetória no desempenho da rede CHT ante a presença de ruído Gaussiano de média gaussianona entrada. Assim, fixa-se K e treina-se a rede apenas com a trajetória I-G1 (ver Fig. 5) contendo 11, 21, 41 e 81 estados. Um resultado típico para $K = 3$ é mostrado na Fig. 7. Por simplicidade, apenas os resultados para a trajetória com 11 e 81 estados são apresentados. Analisando esta figura, pode-se concluir que um aumento gradual no número de estados intermediários resulta em valores menores de erro de reprodução. Os resultados para a trajetória I-G1 com 21 e 41 estados (não mostrados) também confirmam esta afirmação.

Figura 7: Efeito da variação da taxa de amostragem de uma dada trajetória, para um valor fixo de K, na performance da rede durante a reprodução da trajetória sob condições de ruído na entrada.

A próxima simulação estuda o efeito da variação de K (grau de redundância), para um número fixo de estados intermediários, na performance da rede para entradas adicionadas de ruído gaussiano. Utiliza-se a trajetória I-G1 com 81 pontos, e varia-se K de 1 a 5. Um resultado típico é mostrado na Fig. 8, onde estão apenas os resultados para $K = 3$ e 4. Analisando esta figura, pode-se inferir que existe um limite superior para valores de K, visto que na média o valor de $K = 3$ produz melhores resultados (menor RMSE) que $K = 4$. Os resultados para $K = 5$ (não mostrados) são piores que aqueles para $K \leq 4$. Isto ocorre porque, à medida que K aumenta, a região do espaço que os respectivos K neurônios vencedores para um dado estado da trajetória cobrem tende a se sobrepor à região coberta pelos K neurônios vencedores para o próximo estado da trajetória. Isto aumenta a chance de erro durante a reprodução da trajetória. Os menores erros foram encontrados para $K = 1$ e $K = 2$.

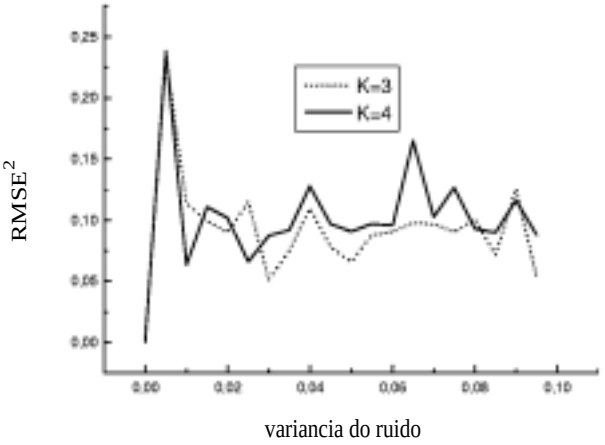

Figura 8: Efeito da variação do grau de redundância K, para um número fixo de 81 estados da trajetória I-G1, na performance da rede durante a reprodução para entradas ruidosas.

A última simulação usa uma trajetória em forma de oito. Esta contém um estado repetido na posição $(0,5\,m; 0,5\,m; 0,5\,m)$. Este tipo de trajetória ilustra a necessidade de contexto local (temporal), pois contém estados repetidos. Diferentemente das três trajetórias anteriores, o contexto global não tem influência nenhuma no processo de reprodução da trajetória em oito, tendo o mesmo valor para todos os estados da trajetória. Assim, a resolução das ambigüidades quando a rede chega ao ponto de cruzamento fica a cargo do contexto local.

Na Fig. 9 (gráfico superior), os resultados da reprodução da trajetória original são mostrados. No gráfico inferior

da Fig. 9, o teste de tolerância a falhas é realizado, de modo que os neurônios vencedores $\mu_1(t)$ são eliminados após o treinamento. Devido ao mecanismo de redundância, a reprodução da trajetória fica agora a cargo dos neurônios $\mu_2(t)$ às expensas de um erro de posicionamento ligeiramente maior. Note que a diferença é quase imperceptível. Os erros obtidos foram $0.0\,m$ (gráfico superior) e $0.008775\,m$ (gráfico inferior).

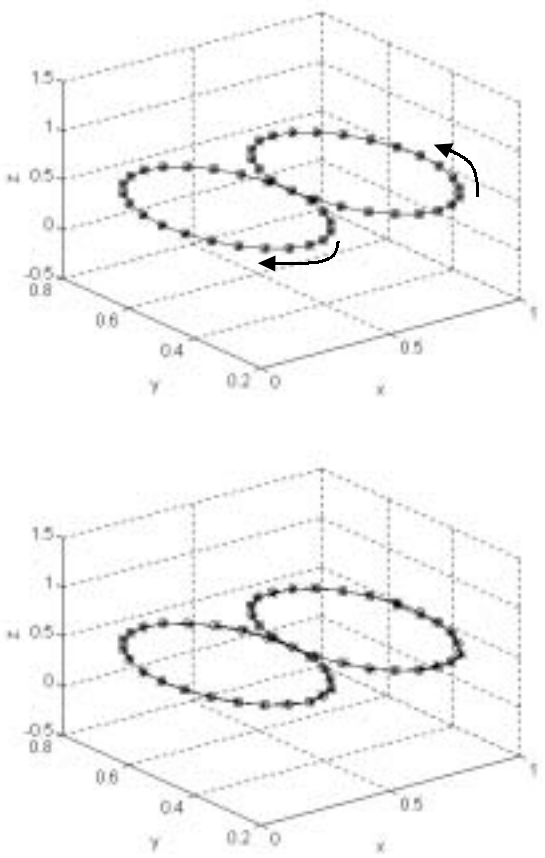

Figura 9: Reprodução de uma trajetória em forma de oito. Reprodução sem falhas (gráfico superior) e com falhas (gráfico inferior). A trajetória real é indicada por círculos e a reproduzida por asteriscos. As setas indicam o sentido do movimento.

5.2 Testes com o Robô Real

Ambientes de simulação são apenas uma aproximação da realidade. Assim, o modelo neural desenvolvido em um ambiente como esse tem que provar a sua robustez sob condições reais de trabalho. O robô PUMA 560 utilizado nos testes a seguir pertence ao Grupo de Neuroinformática da Universidade de Bielefeld (Alemanha), liderado pelo Prof. Helge Ritter. Sem perda de generalidade, nos testes com o robô real apenas os ângulos das juntas, $\boldsymbol{\theta}(t) = \{\theta_i\}$, $i = 1, \ldots, 6$ são utilizados como sinal de controle e realimentação. Logo, o vetor de estado passa a ser $\mathbf{s}(t) = \boldsymbol{\theta}(t)$. As faixas de valores para os ângulos das juntas (em graus) são as seguintes: $\theta_1 \in [-120, 45]$; $\theta_2 \in [-140, -90]$; $\theta_3 \in [-5, 90]$; $\theta_4 \in [-90, 90]$; $\theta_5 \in [-80, 0]$ e $\theta_6 \in [30, 150]$. Os valores dos parâmetros são os mesmos usados na simulação. o erro é calculado usando a Eq. (13) trocando as variáveis $\mathbf{z}^{real}(t)$ e $\mathbf{z}^{rep}(t)$ por $\boldsymbol{\theta}^{real}(t)$ e $\boldsymbol{\theta}^{rep}(t)$, respectivamente, e a unidade do erro passa ser $[graus]$.

O treinamento da rede CHT pode ser feito *on-line* ou *off-line*. Para o primeiro caso, o operador move o braço do robô pelo caminho a ser seguido e a própria rede se encarrega de amostrar este caminho a intervalos regulares de tempo determinados pelo número de estados da trajetória. As posições angulares correspondentes às amostras obtidas constituirão os itens da trajetória. Para o segundo caso, o próprio operador específica uma seqüência de posições angulares que obedeça a faixa de valores imposta a cada um dos ângulos das juntas do robô, sem a necessidade de movimentar o robô propriamente. Optou-se, neste trabalho, pelo treinamento *on-line* por ser mais comum no ambiente industrial. Após o treinamento, o(a) operador(a) do robô fornece à rede apenas o estado inicial da trajetória. O esquema de reprodução autônoma da trajetória funciona em malha fechada (ver Fig. 3), com a rede CHT se encarregando de reproduzir os estados subseqüentes, um a um.

É importante notar que as medidas da posição angular fornecida pelos *encoders* podem ser ligeiramente imprecisas. Esta imprecisão deve-se basicamente à velocidade do efetuador (acoplamento dinâmico entre os segmentos do braço), desgaste das peças do robô, incertezas na especificação dos parâmetros, mau funcionamento dos sensores e ruído de medida. Para minimizar possíveis erros de medição, a velocidade máxima do efetuador nos testes com o robô real foi fixada em 0.5 m/s – o suficiente para mover o braço de forma rápida e garantir o correto funcionamento do sistema de planejamento e controle da trajetória. A Tabela 1 mostra os valores dos ângulos das juntas armazenados pela rede para uma trajetória aberta, retilínea e de tamanho $N_c = 5$, e os valores medidos pelos *encoders*, usados como entrada após a realimentação.

A Tabela 2 traz os valores obtidos quando a rede é treinada com uma trajetória em forma de oito, tamanho $N_c = 7$ e com um estado intermediário ocorrendo duas vezes, em $t = 2$ e $t = 5$.

O papel do contexto local torna-se evidente quando a

Tabela 1: Valores armazenados e medidos (em itálico) dos ângulos das juntas para uma trajetória aberta. Unidades em graus.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
$t=1$	-89.978	-135.002	0.0201	0.0189	-45.0384	90.0216
	-90.007	*-134.492*	*0.3150*	*0.0379*	*-46.0798*	*90.0215*
$t=2$	-79.012	-127.515	0.3930	0.0208	-52.8943	100.989
	-79.369	*-127.773*	*0.3753*	*0.0379*	*-52.6418*	*100.595*
$t=3$	-68.046	-120.029	0.7660	0.0227	-60.7502	111.956
	-68.301	*-120.222*	*0.7507*	*0.0379*	*-60.5402*	*111.654*
$t=4$	-60.078	-120.050	10.9085	0.0229	-70.3688	119.927
	-60.269	*-120.043*	*10.6379*	*0.0379*	*-70.0904*	*119.688*
$t=5$	-52.110	-120.071	21.0511	0.0231	-79.9875	127.899
	-52.305	*-120.055*	*20.7864*	*0.0379*	*-79.7307*	*127.663*
RMSE	0.2315	0.2698	0.2149	0.0173	0.5169	0.2681

Tabela 2: Valores armazenados na rede CHT e os valores medidos (em itálico) dos ângulos das juntas para a trajetória em forma de oito. A unidade dos ângulos é graus.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
$t=1$	-90.000	-135.000	0.000	0.000	-45.000	90.000
	-89.478	*-134.822*	*0.2011*	*0.0189*	*-45.414*	*90.512*
$t=2$	-70.583	-122.664	-1.3379	0.0082	-55.997	109.368
	-70.958	*-122.543*	*-1.1261*	*0.0189*	*-56.335*	*108.980*
$t=3$	-53.634	-136.899	6.4990	0.0082	-49.597	126.317
	-53.990	*-136.600*	*6.3412*	*0.0189*	*-49.748*	*125.939*
$t=4$	-53.634	-121.623	13.588	0.0066	-71.962	126.320
	-53.617	*-122.159*	*13.359*	*0.0189*	*-70.971*	*126.260*
$t=5$	-70.583	-122.664	-1.3379	0.0082	-55.997	109.368
	-70.199	*-122.643*	*-0.9854*	*0.0189*	*-56.361*	*109.726*
$t=6$	-90.007	-116.281	8.4602	0.0070	-72.181	89.947
	-90.019	*-116.804*	*8.2314*	*0.0189*	*-71.221*	*89.8762*
$t=7$	-90.000	-135.000	0.000	0.000	-45.000	90.000
	-89.467	*-134.684*	*-0.0201*	*0.0095*	*-45.319*	*90.504*
RMSE	0.3726	0.3373	0.2202	0.0141	0.5912	0.3677

trajetória contém estados repetidos/compartilhados. A Tabela 3 mostra um caso em que o contexto local para processar a trajetória em forma de oito foi retirado, ou seja, $T = 0$. Nesta tabela, os neurônios que armazenaram os estados da trajetória durante o treinamento são mostrados juntamente com aqueles evocados durante a reprodução da trajetória. Percebe-se que ocorreu um erro, pois os neurônios ativados durante o treinamento são diferentes dos ativados durante a reprodução. Fisicamente, isto significa que o robô não consegue percorrer os dois trechos circulares que formam o oito, ficando "preso" em apenas um deles.

Tabela 3: Neurônios vencedores μ_1 ativados durante as fases de treinamento e reprodução com o contexto local desligado.

	t=1	t=2	t=3	t=4	t=5	t=6	t=7
Treinamento	19	27	13	4	27	3	19
Reprodução	19	27	3	19	27	3	19

6 DISCUSSÃO

A rede CHT foi aplicada em aprendizagem de trajetórias de robôs, problema este que tem larga ocorrência na indústria. Usualmente, a trajetória é "ensinada" ao robô pelo método conhecido como *walk-through*, no qual alguém guia o robô pela seqüência de posições desejadas

para o braço (Fu *et al.*, 1987). Estas posições são então armazenadas na memória do controlador para uma posterior reprodução. Este método consome bastante tempo e é, muitas vezes, inviável economicamente. Isto ocorre em parte porque o robô fica fora de produção durante o processo de armazenamento das trajetórias e, em parte porque, à medida que as trajetórias tornam-se mais complexas, o(a) operador(a) enfrenta dificuldades para resolver sozinho potenciais ambigüidades. Esta última causa motivou fortemente o desenvolvimento da rede proposta neste artigo, visto que é altamente desejável ter o processo de aprendizagem das trajetórias complexas rápido e com mínima intervenção humana.

É importante notar que a rede CHT atua basicamente como elemento planejador da trajetória, fornecendo os valores de referências para o controlador interno do robô a cada instante de tempo. O controle dos atuadores (baixo nível) é realizado, neste trabalho, usando controladores PID convencionais, que são baseados na redução do erro de rastreamento da trajetória. Outros filosofias de controle em baixo nível poderiam igualmente ser utilizadas (por exemplo, fuzzy ou neural). Assim, a rede CHT pode ser utilizada *independentemente* do método de controle em baixo nível e do robô utilizado.

O sinal de realimentação mostrado na Fig. 3 permite que a rede CHT funcione remota e autonomamente, a fim de monitorar, passo-a-passo, o processo de reprodução da trajetória desejada. A trajetória só continua a ser reproduzida se o sinal de realimentação existir. Assim, caso algum problema (por exemplo, colisão com obstáculos) ocorra durante a execução do movimento real por parte do robô, este caminho de realimentação pode ser interrompido e a reprodução terminada. O método convencional *walk-through* de aprendizagem e reprodução de trajetória não possui o caminho de realimentação. Neste caso, todos os estados da trajetória são enviados para o *buffer* do controlador do robô e executados em lote (*batch*), ou seja, de uma única vez. Caso algum problema ocorrer, é necessário esperar pelo término da execução de toda a trajetória para que alguma ação fosse tomada ou então desligar e ligar novamente o robô. Pode-se afirmar portanto que o esquema ilustrado nas Figs. 3 e 4 é um tipo de *reprodução assistida de trajetória*. Outra propriedade da rede CHT que não está presente no método convencional de aprendizagem de trajetórias está na sua tolerância ao ruído e a falhas.

Por fim, compara-se a rede CHT com outras redes não-supervisionadas temporais e discute-se a seleção dos parâmetros da rede CHT. Quando comparado à rede auto-organizável proposta por Wang e Arbib (1990), que também é capaz de lidar com seqüências com estados repetidos, a rede CHT tem o mesmo desempenho usando menos neurônios, visto que aquela mantém várias cópias do estado repetido. Uma abordagem muito similar à desenvolvida neste artigo usando o conceito de padrões espaciais conectados seqüencialmente no tempo via regra hebbiana temporal foi proposto por Kopecz (1995). Entretanto, seu modelo somente é capaz de reproduzir seqüências que não possuam itens repetidos.

Embora a rede CHT possua 8 parâmetros, eles são de fácil seleção e podem ser reduzidos a 4 parâmetros, caso sempre se adote $a_{max} = \eta = \lambda = 1$ e $a_{min} = 0.98a_{max}$. Para os quatro parâmetros restantes têm-se as seguintes orientações: (1) Um valor de $\beta = 100$ ou 1000 mostra-se sempre adequado. (2) As simulações sugeriram $K = 1$ ou 2, valores estes que se confirmaram eficientes na prática. (3) Se o número N de trajetórias a serem aprendidas é conhecido de antemão, o número de neurônios da rede é $M = K \cdot \sum_{l=1}^{M} N_c^l$, onde N_c^l é o número de itens da l-ésima trajetória. Este valor garante que as M trajetórias originais e as $K - 1$ trajetórias redundantes sejam armazenadas. Se N é desconhecido, deve-se deve assumir um valor elevado para M. Após o treinamento, os neurônios que não foram utilizados (i.e, $R_j = 0$) podem ser descartados. (4) O tamanho, T, do contexto local é escolhido por "tentativa e erro", mas pode-se torná-lo adaptativo (Wang, 1995).

7 CONCLUSÃO

Uma rede neural auto-organizável, chamada rede Competitiva e Hebbiana Temporal (CHT), para aprendizagem e reprodução de seqüências temporais foi proposta neste artigo. As seqüências escolhidas para avaliar o algoritmo consistiam de trajetórias de um robô manipulador PUMA 560. O algoritmo é bastante simples, porém poderoso, visto que ele armazena e reproduz com precisão trajetórias complexas, ou seja, contendo estados repetidos e/ou compartilhados. A rede possui certo grau de robustez, sendo capaz de reproduzir as trajetórias armazenadas mesmo na presença de ruído e falhas. Além disso, ela faz uso eficiente de memória (economia de neurônios) se comparada com outras redes equivalentes. Uma plataforma para controle distribuído do robô PUMA 560 baseada na rede CHT mostrou a plausibilidade de se utilizar redes auto-organizáveis no campo da robótica. Testes adicionais podem ser desenvolvidos para avaliar detalhadamente a robustez da rede ao ruído e falhas, bem como a relação entre o número de estados da trajetória (taxa de amostragem) e o raio de similaridade e sua influência na performance da rede.

AGRADECIMENTOS

Os autores agradecem à FAPESP pelo suporte financeiro e aos revisores pelas sugestões dadas para a melhoria do artigo.

REFERÊNCIAS

Althöfer, K. e Bugmann, G. (1995). Planning and learning goal-directed sequences of robot arm movements. *Proc. of the Int. Conf. on Artificial Neural Networks* (ICANN'95), Paris, Vol. 1, pp. 449–454.

Amari, S. (1972). Learning patterns and pattern sequences by self-organizing nets. *IEEE Trans. on Computers*, **21**:1197–1206.

Barreto, G. de A. e Araújo, A. F. R. (1999a). Unsupervised learning and recall of temporal sequences: An application to robotics. *Int. J. Neural Systems*, **9**(3):235–242.

Barreto, G. A. e Araújo, A. F. R. (1999b). Unsupervised context based learning of multiple temporal sequences. *Proc. IEEE Int. Joint Conf. Neural Networks* (IJCNN'99), Washington D.C., pp. 1102–1106.

Barreto, G. A. e Araújo, A. F. R. (2000). Storage and recall of complex temporal sequences through a contextually guided self-organizing neural network. *Proc. IEEE Int. Joint Conf. Neural Networks* (IJCNN'00), Lago di Como, Italy, vol. 3, pp. 207–212.

Barreto, G. A. e Araújo, A. F. R. (2001a). Unsupervised learning and temporal context to recall complex robot trajectories. *Int. J. Neural Systems*, **11**(1):11–22.

Barreto, G. de A. e Araújo, A. F. R. (2001b). Time in self-organizing maps. *Int. J. Computer Research*, **10**(2):139–180.

Billard A. and Hayes G. (1998). DRAMA: A connectionist architecture for control and learning in autonomous robots. *Adaptive Behaviour*, **7**(1):35–64.

Bugmann, G., Koay, K. L., Barlow, N., Phillips, M. e Rodney, D. (1998). Stable encoding of robot trajectories using normalized radial basis functions: Application to an autonomous wheelchair. *Proc. of the 29th Int. Symp. on Robotics* (ISR'98), Birmingham, UK, pp. 232–235.

Corke, I. (1996). A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, **3**(1):24–32.

Fink, G. A., Jungclaus, N., Ritter, H. e Sagerer, G. (1995). A communication framework for heterogeneous distributed pattern analysis. *Proc. IEEE Int. Conf. on Algorithms and Applications for Parallel Processing*, pp. 881–890.

Fu, K. S., Gonzalez, R. C. e Lee, C. S. G. (1987). *Robotics: Control, Sensing, Vision, and Intelligence*. New York:McGraw-Hill.

Gaudiano, P. e Grossberg, S. (1991). Vector associative maps: Unsupervised real-time error-based learning and control of movement trajectories. *Neural Networks*, **4**:147–183.

Hebb, D. O. (1949). The Organization of Behavior. New York, NY:Wiley.

Heikkonen, J. e Koikkalainen, P. (1997). Self-organization and autonomous robots. In: *Neural Systems for Robotics*, O. Omidvar evan der Smagt (eds.), Academic Press, pp. 297–337.

Jockusch, J. (2000). Exploration based on neural networks with applications in manipulator control. *Tese de Doutorado*, Faculdade de Tecnologia, Grupo de Neuroinformática, Universidade de Bielefeld, Alemanha.

Kopecz, K. (1995). Unsupervised learning of sequences on maps with lateral connectivity. *Proc. of the Int. Conf. on Artificial Neural Networks* (ICANN'95), Vol. 2, pp. 431–436.

Lloyd, J., Parker, M. e McClain, R. (1988). Extending the RCCL programming environment to multiple robots and processors. *Proc. IEEE Int. Conf. Robotics and Automation*, Philadelphia, PA, pp. 465–469.

Martinetz, T. M., Ritter, H. J. e Schulten, K. J. (1990). Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Trans. on Neural Networks*, **1**(1):131–136.

Massone, L. (1995). Sensorimotor learning. In: *The Handbook of Brain Theory and Neural Networks*, M. Arbib (ed.), MIT Press.

Prabhu, S. M. e Garg, D. P. (1996). Artificial neural network based robot control: An overview. *J. Intell. Robot. Syst.*, **15**:333–365.

Walter, J. A. e Schulten, K. J. (1993). Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Trans. on Neural Networks*, **4**(1):86–95.

Wang, D. L. e Arbib, M. A. (1990). Complex temporal sequence learning based on short-term memory. *Proc. IEEE*, **78**(9):1536–1543.

Wang, D. L. (1995). Temporal pattern processing. In: *The Handbook of Brain Theory and Neural Networks*, M. Arbib (ed.), pp. 967–971, MIT Press.