CONTROLE DISTRIBUÍDO DO ROBÔ PUMA 560 USANDO UMA REDE NEURAL AUTO-ORGANIZÁVEL TEMPORAL

Guilherme de A. Barreto, Aluizio F. R. Araújo

Departamento de Engenharia Elétrica, Universidade de São Paulo (USP)
Av. Trabalhador Sancarlense, 400, São Carlos, SP, 13560-950
{gbarreto, aluizioa}@sel.eesc.sc.usp.br

Christof Dücker, Helge Ritter

Departamento de Ciência da Computação, Universidade de Bielefeld
P. O.-Box 100131, Bielefeld, Alemanha, 33501
{chrisd, helge}@techfak.uni-bielefeld.de

Resumo— Neste artigo é proposto um sistema de controle distribuído para robôs manipuladores baseado em uma rede neural auto-organizável, chamada rede competitiva e hebbiana temporal (CHT). Esta aprende e reproduz trajetórias complexas por meio de dois conjuntos de conexões sinápticas: conexões competitivas de propagação para frente armazenam os estados individuais de uma trajetória, enquanto conexões hebbianas laterais codificam a ordem temporal dos estados dessa trajetória. Uma ferramenta de comunicação distribuída é utilizada junto com a rede CHT no planejamento em tempo real de trajetórias para um robô PUMA 560. A performance do sistema proposto é discutida e comparada com outras abordagens por redes neurais.

Abstract— A distributed robot control system is proposed based on a temporal self-organizing neural network, called competitive and temporal hebbian (CTH) network. The CTH network can learn and recall complex trajectories using two sets of synaptic connections: competitive feedforward weights that encode the individual states of the trajectory, and hebbian lateral weights that encode the temporal order of trajectory states. A distributed processing scheme is proposed to evaluate the CTH network in point-to-point, real-time trajectory planning of a PUMA 560 robot. The performance of the proposed system is discussed and compared with other neural network approaches.

Key Words— self-organization, neural networks, robotics, trajectory planning, distributed control.

1 Introdução

Com freqüência, tarefas envolvendo robôs manipuladores possuem uma natureza seqüencial bem definida pela ordem temporal das posições que o braço de robô deve assumir ao longo de um caminho preestabelecido. Em geral, esta informação temporal não é incorporada ao processo de aprendizagem de uma rede neural artificial (RNA), de forma tal que apenas transformações sensório-motoras estáticas (cinemática inversa, por exemplo) podem ser aprendidas. Nestes casos, a ordem temporal da tarefa robótica é estabelecida pelo projetista da rede. Uma alternativa interessante consiste em usar *RNAs temporais*, visto que elas incorporam automaticamente os aspectos seqüenciais da tarefa robótica de interesse durante a fase de treinamento. Durante uma fase posterior de reprodução da trajetória armazenada, o estado atual 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. Este procedimento é repetido até que toda a trajetória seja reproduzida (Barreto e Araújo, 2001).

Este artigo demonstra a viabilidade da aplicação de redes neurais auto-organizáveis temporais no controle distribuído em tempo real de robôs manipuladores. O algoritmo de aprendizagem é avaliado por sua habilidade de aprender e reproduzir trajetórias complexas de maneira precisa e sem ambigüidades. O restante do artigo está organizada da seguinte forma. Na Seção 2, a rede neural é apresentada. Na Seção 3, a plataforma de controle robótico distribuído e seus principais componentes são introduzidos. Na Seção 4, testes com o sistema proposto no controle ponto-a-ponto do robô PUMA 560 são descritos. O artigo é concluído na Seção 5.

2 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. A rede CHT consiste basicamente de conexões de propagação para frente (*feedforward*) e conexões laterais que desempenham papéis distintos na sua dinâmica. Ela possui também unidades de contexto na entrada e atrasadores na saída. Contudo, os atrasadores são necessários apenas na fase de treinamento durante a aprendizagem de transições de estado. A entrada da rede CHT é formada por um vetor de estado $\mathbf{s}(t) \in \Re^m$, um vetor de contexto fixo $\mathbf{C}_F \in \Re^m$, e um vetor de contexto variante no tempo $\mathbf{C}_T(L,t) \in \Re^{L \cdot m}$. O

vetor de estado, $\mathbf{s}(t)$, define a configuração do braço do robô no instante t, sendo definido como $\mathbf{s}(t) = [\mathbf{r}(t) \;\; \boldsymbol{\theta}(t)]^T$, onde $\mathbf{r}(t) \in \Re^3$ é a coordenada cartesiana do efetuador do robô, e $\boldsymbol{\theta}(t) \in \Re^{gdl}$ é o vetor de ângulos das juntas que posicionam o efetuador em $\mathbf{r}(t)$. O expoente *gdl* especifica o número de graus de liberdade do robô. O contexto fixo $\mathbf{C}_F$ é invariante no tempo e, em geral, recebe o valor do estado final da trajetória. $\mathbf{C}_F$ é mantido invariável até que o estado final da trajetória tenha sido alcançado e sua função é atuar com um *identificador global da trajetória.* O contexto temporal consiste no histórico de vetores de estados, ou seja, $\mathbf{C}_T(L,t) = \{\mathbf{s}(t-1), \ldots, \mathbf{s}(t-L)\}$, $\mathbf{C}_T(L,t) \in \Re^{L \cdot m}$ onde $L > 0$ é um número inteiro determinado por tentativa-e-erro.

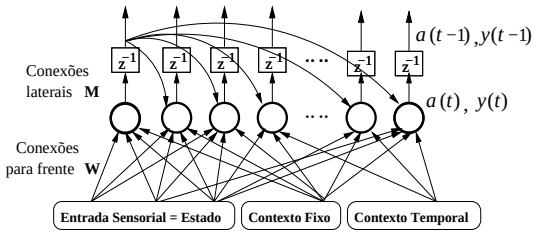

Figura 1. Arquitetura da rede CHT. Por simplicidade, apenas algumas conexões laterais são mostradas.

2.1 Ajuste dos Pesos Sinápticos da Rede

Um vetor de conexões de propagação para frente, $\mathbf{w}_j$, j=1,...,M, conecta as unidades de entrada ao neurônio de saída j, sendo definido como $\mathbf{w}_j(t) = [\mathbf{w}_j^s(t) \;\; \mathbf{w}_j^F(t) \;\; \mathbf{w}_j^T(t)]^T$, onde $\mathbf{w}_j(t) \in \Re^{(L+2)\cdot m}$. Em cada instante t, um certo $\mathbf{w}_j$ é escolhido para armazenar o estado atual do braço via aprendizagem competitiva, i.e., um único neurônio (ou um pequeno grupo de neurônios) de saída fica responsável pelo armazenamento de tal estado. Para isso, a distância entre o estado atual da trajetória $\mathbf{s}(t)$ e cada $\mathbf{w}_j$ é calculada como $D_j^s(t) = (\mathbf{s}(t) - \mathbf{w}_j^s(t))^T \mathbf{P} (\mathbf{s}(t) - \mathbf{w}_j^s(t))$, onde $\mathbf{P}$ é uma matriz diagonal usada para encontrar o neurônio vencedor baseando-se apenas em $\mathbf{r}(t)$. Esta matriz é definida como:

$$\mathbf{P} = \mathbf{diag}\,(\mathbf{1_r}, 0, \ldots, 0)$$

onde $\mathbf{1_r}$ é um vetor de 1's, tal que $\dim(\mathbf{1_r}) = \dim(\mathbf{r}(t))$. São também definidas uma distância de contexto fixa, $D_j^F(t) = \|\mathbf{C}_F(t) - \mathbf{w}_j^F(t)\|$ e uma distância de contexto temporal, $D_j^T(t) = \|\mathbf{C}_T(t,L) - \mathbf{w}_j^T(t)\|$. Enquanto $D_j^s(t)$ é usada para encontrar o(s) neurônio(s) vencedor(es) da competição atual, $D_j^F(t)$ e $D_j^T(t)$ são usadas para resolver ambigüidades durante a reprodução das trajetórias memorizadas pela rede CHT. A escolha do neurônio vencedor é feita com base na função $f_j(t)$, chamada *função escolha*, dada por:

$$f_j(t) = \begin{cases} D_j^s(t) & \text{Se } D_j^s(t) \leq \varepsilon \text{ ou } R_j(t) = 0 \\ R_j(t) \cdot D_j^s(t) & \text{Caso contrário.} \end{cases} \quad (1)$$

onde o parâmetro $0 < \varepsilon \leq 1$, chamado *raio de similaridade*, define uma vizinhança em torno de cada $\mathbf{s}(t)$ dentro da qual o vetor de pesos $\mathbf{w}_j^s(t)$ pode ser considerado *suficientemente próximo* a $\mathbf{s}(t)$. A função $R_j(t)$ é chamada *função responsabilidade*, pois indica se o neurônio j já é responsável pelo armazenamento de um estado trajetória, ou seja, se ele já venceu alguma competição. Se $R_j(t) > 0$, o neurônio j é excluído da competição e das subseqüentes. Se $R_j(t) = 0$, o neurônio j pode participar da competição atual. Durante o treinamento, faz-se $\varepsilon \ll 1$ de modo que cada neurônio da rede armazene um estado *distinto* da trajetória. Durante a reprodução da trajetória armazenada, o raio e similaridade deve assumir valores maiores (i.e. $\varepsilon \approx 1$) para evitar avaliações incorretas da Eq. (1) resultantes da presença de ruído na entrada. Os neurônios de saída são então ordenados da seguinte forma:

$$f_{\mu_1}(t) < f_{\mu_2}(t) < \cdots < f_{\mu_M}(t) \quad (2)$$

onde M é o número de neurônios de saída e $\mu_i(t)$, $i = 1, \ldots, M$, é o índice do i-ésimo neurônio de saída mais próximo de $\mathbf{s}(t)$. Escolhe-se K neurônios vencedores, $\{\mu_1, \ldots, \mu_K\}$ para representar o estado atual $\mathbf{s}(t)$ e seu contexto $\{\mathbf{C}_F, \mathbf{C}_T(t,L)\}$. Em seguida, a ativação de cada um dos neurônios de saída é computada:

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

Portanto, os valores de ativação decaem linearmente de um valor máximo $a_{max} \in \Re$ para $\mu_1(t)$, até um valor mínimo $a_{min} \in \Re$ para $\mu_K(t)$. As constantes a_{max} e a_{min} são definidas pelo projetista da rede. O próximo passo consiste em atualizar a função responsabilidade $R_j(t)$ da seguinte forma $R_j(t+1) = R_j(t) + \beta a_j(t)$, onde $\beta \gg 1$ é chamada *constante de exclusão.* Finalmente, os vetores de pesos da rede CHT são ajustados:

$$\begin{aligned} \mathbf{w}_j^s(t+1) &= \mathbf{w}_j^s(t) + \eta a_j(t)[\mathbf{s}(t) - \mathbf{w}_j^s(t)] \\ \mathbf{w}_j^F(t+1) &= \mathbf{w}_j^F(t) + \eta a_j(t)[\mathbf{C}_F(t) - \mathbf{w}_j^F(t)] \\ \mathbf{w}_j^T(t+1) &= \mathbf{w}_j^T(t) + \eta a_j(t)[\mathbf{C}_T(t-1) - \mathbf{w}_j^T(t)] \end{aligned} \quad (3)$$

onde $0 < \eta \leq 1$ é a taxa de aprendizagem. Para $t = 0$, $\mathbf{w}_j(0)$ recebe valores aleatórios entre 0 e 1.

Um conjunto de pesos laterais, $\mathbf{m}_j(t) = [m_{j1} \;\; m_{j2} \;\; \cdots \;\; m_{jM}]^T$, codifica a ordem temporal dos estados da trajetória usando uma regra de aprendizagem hebbiana que conecta o vencedor da competição anterior com o vencedor da com-

petição atual:

$$\Delta m_{jr}(t+1) = \begin{cases} 0 & \text{Se } m_{jr}(t) \neq 0 \\ \lambda a_j(t) a_r(t-1) & \text{Caso contrário.} \end{cases} \quad (4)$$

onde $0 < \lambda < 1$ é uma constante. Segundo a Eq. (4) a rede CHT "olha" um passo 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. Esta transição é codificada pelo peso lateral que conecta os neurônios que geraram os pares de ativação $[a_{\mu_i}(t-1), a_{\mu_i}(t)]$, $i \leq K$. A aplicação sucessiva da Eq. (4) leva à codificação da ordem temporal da trajetória. Inicialmente, $m_{jr}(0) = 0$ para todo j, r, indicando que nenhuma associação temporal existe no começo do treinamento.

2.2 Reprodução de Trajetórias

O processo de reprodução de uma trajetória memorizada é um esquema de controle em malha fechada (ver Fig. 2) consistindo de 4 passos: (1) Início da reprodução, (2) cálculo das ativações e das saídas, (3) geração dos sinais de controle do robô, e (4) determinação das entradas sensoriais. Para fins de reprodução, faz-se sempre $K = 1$.

1. Início da Reprodução: Para iniciar a reprodução ($t = 0$ na Fig. 2), qualquer estado semelhante a um dos estados armazenados pode ser apresentado à rede CHT. Este estado inicial é chamado de *estado disparador* da reprodução. O contexto fixo $\mathbf{C}_F$ assume o valor do estado final desta trajetória, enquanto que os valores iniciais do contexto temporal são feitos iguais ao estado disparador. A rede então se encarrega de reproduzir, de forma autônoma, o restante da trajetória ($t > 0$ na Fig. 2).

2. Cálculo das Ativações e Saídas: Para cada estado $\mathbf{s}(t)$, a ativação do neurônio vencedor, a_{μ_1}, é calculada, indicando qual neurônio armazenou o estado mais próximo do estado atualmente na entrada da rede CHT. Em seguida, o neurônio vencedor (único neurônio de saída com ativação não-nula) ativará o neurônio cujo vetor de pesos armazenou o estado que vem em seguida ao atual estado da trajetória. 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 dada por:

$$y_j(t) = y^F(t) \cdot y^T(t) \cdot G\left(\sum_{r=1}^{m} m_{jr}(t) a_r(t)\right) \quad (5)$$

onde $y^F(t) = 1 - D_j^F(t) / \sum_{r=1}^{m} D_r^F(t)$ e $y^T(t) = 1 - D_j^T(t) / \sum_{r=1}^{m} D_r^T(t)$. A função G é escolhida tal que $G(u) \geq 0$ e $dG(u)/du > 0$. Para trajetórias sem estados repetidos, o terceiro fator no lado direito da Eq. (5) indicará corretamente o neurônio que armazenou o próximo estado da trajetória, $\mathbf{s}^{next}=[\mathbf{r}^{next}\ \boldsymbol{\theta}^{next}]^T$, e que fornecerá o próximo sinal de controle ao robô (Passo 3). Para trajetórias com estados repetidos, informação adicional é necessária, pois o terceiro termo sozinho produzirá o mesmo valor de $y_j(t)$ para todos os neurônios "candidatos" a conter o próximo estado. Estas "ambigüidades" são resolvidas pelos primeiro e segundo fatores no lado direito da Eq. (5): o neurônio cujo contexto armazenado mais se aproxima daquele atualmente na entrada da rede é aquele a ser escolhido.

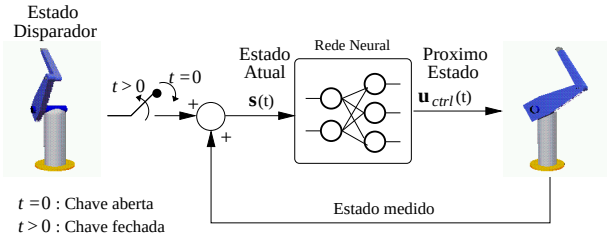

Figura 2. Reprodução autônoma de trajetórias.

3. Determinação dos Sinais de Controle: O sinal de controle (ângulos das juntas) a ser enviado ao robô é calculado a partir do vetor de pesos do neurônio com maior valor de $y_j(t)$ e da matriz $\bar{\mathbf{P}} = (\mathbf{I} - \mathbf{P})$:

$$\mathbf{u}_{ctrl}(t) = \bar{\mathbf{P}}\mathbf{w}_{j^*}^s(t) = \bar{\mathbf{P}}\begin{pmatrix} \mathbf{r}^{next} \\ \boldsymbol{\theta}^{next} \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{\theta}^{next} \end{pmatrix} \quad (6)$$

onde $j^* = \arg\max_j[y_j(t)]$. Note que $\mathbf{u}_{ctrl}(t)$ fornece os ângulos associados com $\mathbf{r}_{next}$.

4. Determinação das Entradas Sensoriais: Um conjunto de medidas sensoriais fornecem 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 2-4 são executados várias vezes até que o final da trajetória tenha sido alcançado.

3 Plataforma Robótica

Este trabalho utilizou o robô PUMA 560, um manipulador de 6 graus de liberdade conectado ao controlador Unimation Mark III. Este, por sua vez, coordena vários controladores PID que acionam de maneira independente os servomecanismos de cada uma das juntas. Estes controladores são acionados por um controlador principal, a CPU LSI-11/73. Grande parte do software do controlador original do robô PUMA foi substituído e uma estação de trabalho SUN Sparcsystem 4/370 com barramento VME foi empregada para controlar *diretamente* o robô em tempo real via um *link* de comunicação de alta velocidade

(Walter e Schulten, 1993). Para controlar o robô a partir da estação SUN foi instalado o pacote RCCL/RCI (*R*obot *C*ontrol *C* *L*ibrary e *R*eal-time *C*ontrol *I*nterface) (Lloyd *et al.*, 1988). Este consiste num conjunto de bibliotecas que permite requisitar a movimentação do robô a partir de um programa escrito em linguagem C.

Neste trabalho foi desenvolvida uma interface amigável para facilitar o envio de sinais de controle para o robô e a leitura de sinais de realimentação a partir dele. Em vez de utilizar diretamente as funções do pacote RCCL/RCI (exige elevado grau de conhecimento), o usuário inclui no programa em C que a rede neural é desenvolvida chamadas a funções (bibliotecas) específicas que realizam, de modo transparente para ele, o controle do robô. Além disso, o controle pode ser feito remotamente de qualquer computador conectado à rede local.

3.1 Processamento Distribuído

A rede CHT e o robô PUMA 560 se comunicam de forma síncrona por meio de uma ferramenta de comunicação denominada *Distributed Applications Communication System* (DACS), desenvolvida por Fink *et al.* (1995). A ferramenta DACS é baseada na arquitetura *cliente-servidor* e, quando aplicada ao presente trabalho, a rede CHT exerce o papel da aplicação-cliente e o pacote RCCL/RCI funciona como servidor. 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. Uma comunicação eficiente entre as aplicações e seus respectivos DACS-daemon, bem como a comunicação entre aplicações, é possível graças a um conjunto de bibliotecas DACS e DACS-daemons que são implementados como tarefas paralelas via *threads*[a].

Detalhes da interação entre duas aplicações durante uma chamada remota síncrona é mostrada na Figura 3. Nesta figura a aplicação denominada *robot* provê uma função *move*. A aplicação *net* 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]Em programação, *thread* é uma parte de um programa que pode ser executada independente de outras partes.

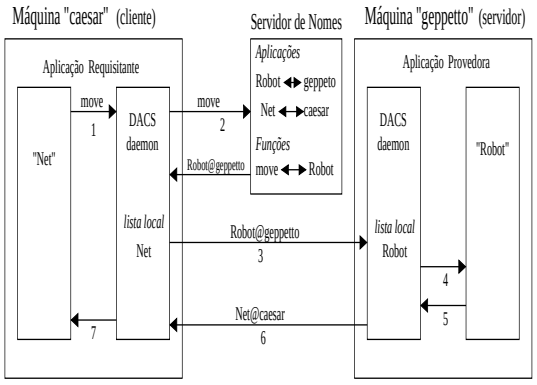

Figura 3. Interações durante uma chamada síncrona.

Esta mensagem é 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 cliente, ou seja, *net@caesar*.

5. O resultado é enviado 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 necessária 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. Os passos 1-7 são executados enquanto a rede CHT estiver reproduzindo uma trajetória.

4 Testes com o Sistema Proposto

A rede CHT foi previamente avaliada em tarefas robóticas apenas por meio de simulações (Barreto e Araújo, 2000). O robô PUMA 560 usado nos testes a seguir pertence ao Laboratório de Robótica do Grupo de Neuroinformática da Universidade de Bielefeld, Alemanha. Para tanto, uma trajetória com a forma aproximada de um oito (ou gravata borboleta!) foi gerada movendo-se o braço do robô e gravando o ângulos das juntas e a posição cartesiana resultante do efetu-

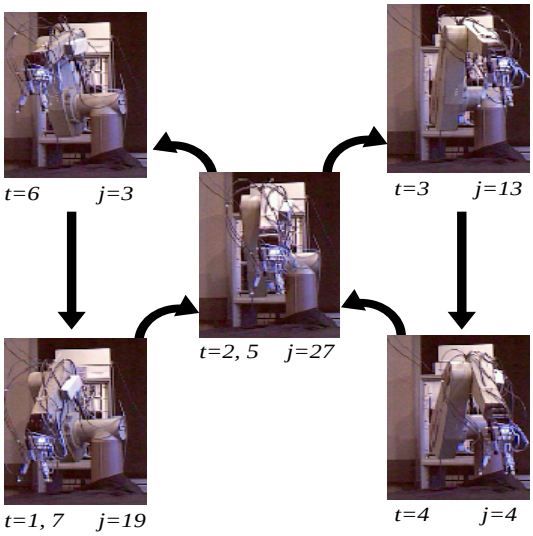

Figura 4. Trajetória em forma de oito executada pelo robô.

ador do robô em 7 posições (estados) ao longo da trajetória. Este tipo de trajetória tem sido largamente usada em tarefas de reprodução de seqüências porque possui um estado que ocorre duas vezes ($t = 2$ e $t = 5$), porém em diferentes contextos temporais. A rede CHT foi então treinada com esta trajetória e usada para controlar o robô PUMA conforme mostrado na Fig. 2. As faixas de valores (em graus) para os ângulos das juntas são os 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 $\theta_6 \in [30, 150]$. O contexto fixo, $\mathbf{C}_F$, é sempre feito igual ao estado final da trajetória que, neste caso, é igual ao estado inicial. Para o contexto temporal faz-se $L = 2$, daí $\mathbf{C}_T(L,t) = \{\mathbf{s}(t-1), \mathbf{s}(t-2)\}$. Os parâmetros para todos os testes são os seguintes: $M = 30$, $\varepsilon = 10^{-6}$ (treinamento), $\varepsilon = 1$ (reprodução), $K = 2$, $a_{max} = 1$, $a_{min} = 0.98$, $\beta = 100$, $\eta = 1$, e $\lambda = 0.8$. Nota-se que, como $\eta = 1$, basta uma única apresentação da trajetória para a rede CHT memorizar todos os seus estados. Para os testes que se seguem a velocidade do efetuador foi fixada em 0.5 m/s, valor suficiente para mover o robô rapidamente e ainda obter medidas precisas que garantam o correto funcionamento do sistema de controle como um todo.

A Fig. 4 mostra uma seqüência de posições do braço do robô PUMA 560 descrevendo a trajetória em oito, juntamente com o índice do neurônio que armazenou determinado estado da trajetória e o instante de tempo (posição na seqüência) correspondente. As setas indicam as transições de estado que a rede CHT codificou. Toda posição alcançada pelo robô é medida por *encoders* óticos montados em cada junta e então enviada para a entrada da rede para dar continuidade ao processo de reprodução autônoma da trajetória. Este processo é repetido até que o fim da trajetória tenha sido alcançada. Um polinômio é ajustado à seqüência de posições angulares evocadas para suavizar os movimentos do robô.

A Fig. 5 ilustra como a Eq. (5) resolve ambigüidades. Não importa que trecho da trajetória o robô esteja atualmente executando, quando ele alcança o ponto de cruzamento (estado repetido em $t = 2$ e $t = 5$), ele tem que decidir entre um dos dois sentidos possíveis a seguir. Esta indecisão é resolvida através do uso do contexto temporal, $\mathbf{C}_T(L,t)$, já que o contexto fixo é o mesmo para as duas ocorrências desse estado.

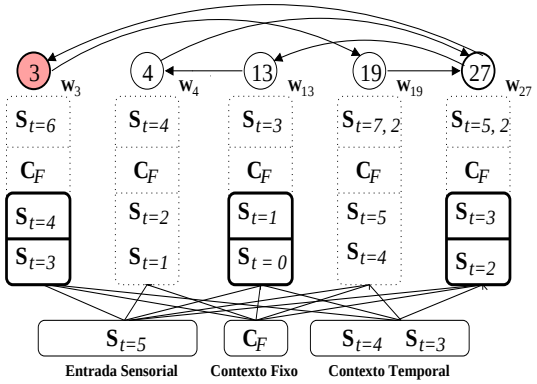

Figura 5. Resolvendo ambigüidades durante a reprodução.

Conforme já mencionado, os estados $\mathbf{s}_{t=2}$ e $\mathbf{s}_{t=5}$ são iguais, mas ocorrem em contextos diferentes. Durante o treinamento estes estados foram armazenados pelo neurônio $j = 27$ (Fig. 5). Durante a reprodução, por exemplo, quando o robô alcança o estado $\mathbf{s}_{t=5}$, a rede CHT tem que decidir qual o próximo estado (sinal de controle) a ser enviado para o robô. Acompanhando as conexões laterais na Fig. 5, os neurônios $j = 3$ e $j = 13$ são os candidatos a conter o próximo estado em seus vetores de pesos. Como se trata de um estado repetido dentro de uma mesma trajetória e não um compartilhado com outra trajetória, o contexto fixo armazenado é o mesmo para os dois neurônios. Apenas o contexto temporal, $\mathbf{C}_T$, contém informação que pode resolver a ambigüidade, ou seja, os estados anteriores $\mathbf{C}_T(5,L) = \{\mathbf{s}_{t=4}, \ \mathbf{s}_{t=3}\}$. Esta informação "casa" perfeitamente com aquela armazenada na porção inferior do vetor de pesos $\mathbf{w}_3$, i.e., $\mathbf{C}_T(5,L) \equiv \mathbf{w}_3^T$. Assim, $D_3^T(t) < D_{13}^T(t)$, implicando que $y_3 > y_{13}$. logo, o neurônio $j = 3$ é escolhido e o próximo sinal e controle é extraído de $\mathbf{w}_3^s$ de acordo com a Eq. (6).

A Tabela 1 ilustra o que acontece durante a reprodução da trajetória armazenada quando o contexto temporal é desconsiderado. Nesta tabela, os neurônios que armazenaram os 7 estados da trajetória durante o treinamento são mostrados juntamente com aqueles ativados durante a fase de reprodução. Nota-se que um erro ocorreu visto que os neurônios vencedores durante o treinamento são diferentes daqueles evocados durante a reprodução. O significado físico deste erro é que o

robô não consegue passar pelos dois lados da trajetória em oito, ficando preso em um deles.

	t=1	t=2	t=3	t=4	t=5	t=6	t=7
T	19	27	13	4	27	3	19
R	19	27	3	19	27	3	19

Tabela 1. Neurônios vencedores μ_1 ativados durante treinamento (T) e reprodução (R) sem contexto temporal.

5 DISCUSSÃO E CONCLUSÃO

Usualmente, trajetórias são "ensinada" ao robô pelo método conhecido como *walk-through*, no qual alguém guia o robô pela seqüência de posições desejadas para o braço. Estas posições são então armazenadas na memória do controlador (*look-up table*) para uma posterior reprodução (Fu *et al.*, 1987). 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 também que a rede CHT atua basicamente como elemento planejador da trajetória, fornecendo os valores de referências (*set-points*) 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. 2 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, 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 de uma única vez. Se algum problema ocorrer, tem-se que esperar pelo término da execução de toda a trajetória para que alguma ação fosse tomada ou então desligar e ligar novamente o robô. Assim, o esquema adotado neste trabalho é um tipo de *reprodução assistida de trajetória*. Outra propriedade da rede CHT que não está presente no método convencional está na sua tolerância ao ruído e a falhas.

Além disso, o esquema proposto implementa um esquema de *modelagem e controle inverso especializado* (Prabhu e Garg, 1996) pois a rede CHT é treinada para operar em regiões *específicas* do espaço de trabalho do robô. Outros sistemas de controle em tempo real que usam redes auto-organizáveis (Walter e Schulten, 1993) implementam um esquema de *modelagem e controle inverso generalizado* visto que estes tentam aprender transformações sensório-motoras globalmente. Segundo Prabhu e Garg (1996), o esquema especializado tem treinamento mais rápido e é mais preciso. Até onde se tem conhecimento, o sistema de controle neural não-supervisionado e distribuído proposto neste artigo é o primeiro a implementar em um robô real um método inverso especializado que leva automaticamente em consideração aspectos seqüenciais (temporais) da tarefa robótica. Abordagens supervisionadas têm sido propostas e apenas simuladas (Massone e Bizzi, 1989), mas elas não são adequadas para treinamento $on-line$ pois exigem um longo processo de treinamento.

AGRADECIMENTOS

Os autores agradecem à FAPESP pelo suporte financeiro (projeto 98/12699-7).

Referências

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-INNS IJCNN*, Como, Italy, vol. 3, pp. 207–212.

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

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.

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

Massone, L. e Bizzi, E. (1989). A neural network model for limb trajectory formation. *Biol. Cybern.* **61**:417–425.

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.