Proceedings of the V Brazilian Conference on Neural Networks - V Congresso Brasileiro de Redes Neurais
pp. 571–576, April 2–5, 2001 - Rio de Janeiro - RJ - Brazil

Geração de Trajetórias para Robôs Móveis Autônomos Usando Redes Neurais Artificiais

José F. A. de Andrade , André Mendeleck e Douglas E. Zampieri
Departamento de Mecânica Computacional - DMC
Faculdade de Engenharia Mecânica - FEM
Universidade Estadual de Campinas - UNICAMP
E-mail: jfabreu@fem.unicamp.br , douglas@fem.unicamp.br

Abstract

The autonomous navigation of vehicles has been an area of great interest for researchers for a long time, mainly the artificial intelligence subject. One of the problems of greatest interest is that related to the trajectory generated of a vehicle, which shall be guided in an environment, avoiding colliding with obstacles in order to reach pre defined positions. This paper presents a methodology to generate trajectories using neural networks applied to mobile robot navigation. The numerical simulation results presented graphically shown that the proposed methodology were efficient to guide the robot in dynamic unknown environment

1. Introdução

A navegação autônoma de veículos há muito desperta o interesse de pesquisadores, principalmente na área de inteligência artificial. Um dos problemas que desperta maior interesse, concentra-se na determinação de uma trajetória, de modo que o veículo possa navegar pelo ambiente evitando colidir com obstáculos até alcançar uma ou mais posições pré-estabelecidas no ambiente[1].

O planejamento de trajetória envolve um grande número de etapas. Planejar significa organizar as informações disponíveis e previstas, de tal forma a conduzir um objeto de um estado a outro, de preferência com segurança e estabilidade. Como o ambiente de trabalho geralmente não é estático, o planejamento das ações torna-se uma tarefa dinâmica e o sistema deve ter a capacidade de realizar as adaptações e a tomada de decisões diferentes (ajustes) da planejada, ou seja, um replanejamento. E mais, no planejamento deve-se prever o aprendizado de fatos e ações não previstas, para futura utilização [2].

Neste trabalho, a simulação da interação do robô móvel com o meio é feita através de mecanismos sensoriais, permitindo que o mesmo seja capaz de operar autonomamente em um local desconhecido e não necessariamente estruturado.

As abordagens baseadas em sensores utilizam medições realizadas diretamente no ambiente, para determinar o próximo movimento do veículo, com o objetivo de evitar obstáculos, ao mesmo tempo que dirigem o veículo em direção ao alvo. A principal vantagem da navegação baseada em sensores é que o veículo pode navegar em ambientes dinâmicos e desconhecidos, pois as únicas informações necessárias para a movimentação são obtidas dos próprios sensores [3].

Enquanto se move, o robô deve possuir a capacidade de colecionar informações do ambiente (mapeamento e modelagem do ambiente) e também deve ter alguma noção da sua localização [4].

Atualmente algumas técnicas são empregadas para solucionar o problema da navegação na robótica móvel destacando-se na área de inteligência artificial o uso de redes neurais artificiais [5],[6],[7], lógica *fuzzy* [8], aprendizado por reforço [9], computação evolutiva [10], além do uso de sistemas híbridos como *neuro-fuzzy* [11] e *genetic-fuzzy* [12]. Entre outras técnicas tradicionais utilizadas com o mesmo propósito, destaca-se o uso dos filtros de Kalman [13], [14].

Na área da inteligência artificial foram escolhidas as redes neurais artificiais, pois as mesmas se mostraram eficientes para solucionar o problema básico da navegação, sendo utilizada uma rede do tipo MLP (*Multilayer Perceptron*) com uma camada intermediária e recorrência externa. No seu treinamento foram usados padrões de comportamentos esperados durante a navegação, padrões estes obtidos a partir de uma série de regras pré-definidas por um instrutor.

A seguir será feita uma breve revisão das redes neurais artificias.

2. Redes Neurais Artificiais

O estímulo inicial que conduziu ao desenvolvimento de modelos matemáticos de redes neurais, denominadas redes neurais artificiais, foi um esforço para entender mais detalhadamente o funcionamento do cérebro humano. O objetivo era construir mecanismos que operassem de modo similar, ou seja, que tomassem decisões, processassem informação, aprendessem, lembrassem e otimizassem da mesma forma e, se possível, até de forma mais eficiente que o cérebro humano [15].

Um neurônio é a unidade fundamental de processamento de informação de uma rede neural [16]. Basicamente no modelo do neurônio artificial são identificados três elementos: um conjunto de conexões

de entrada, cada qual ponderada por um peso sináptico; uma junção de soma e uma função de ativação.

Redes neurais artificiais têm sido aplicadas com sucesso nos mais diversos problemas. Dentre as principais áreas de aplicação de redes neurais artificiais destacam-se: sistemas de controle, reconhecimento de padrões e aproximação de funções. Embora existam inúmeras arquiteturas de redes neurais, a arquitetura multicamadas é a mais freqüentemente encontrada na literatura [17]. Entre as razões para a sua popularidade podemos citar sua capacidade de aproximação universal e sua flexibilidade para formar soluções de qualidade para uma ampla classe de problemas, a partir de um mesmo algoritmo de aprendizado.

As redes neurais do tipo MLP ou perceptron multicamadas são arquiteturas de múltiplas camadas com pelo menos uma camada intermediária (camada que não é nem de entrada, nem de saída).

Elas apresentam aprendizado do tipo supervisionado, caracterizado pela presença de um "professor" ou "instrutor" externo. A função do instrutor durante o processo de aprendizado é suprir a rede neural com uma resposta desejada a um determinado estímulo apresentado pelo ambiente. É definido então um sinal de erro como a diferença entre a resposta desejada e a resposta observada na saída da rede neural. A partir desse sinal de erro os parâmetros da rede são então ajustados.

No treinamento das redes neurais do tipo MLP é utilizado o algoritmo de retropropagação (*backpropagation*) que consiste basicamente de dois passos de computação: o processamento direto (*forward*) e o processamento reverso (*backward*). No processamento direto, uma entrada é aplicada à rede neural e seu efeito é propagado pela rede camada a camada. Durante o processamento direto, os pesos da rede permanecem fixos. No processamento reverso, um sinal de erro calculado na saída da rede é propagado no sentido reverso, camada a camada, e ao final deste processo os pesos são ajustados de acordo com uma regra de correção de erro [17].

Como o treinamento de redes neurais multicamadas é um problema de otimização não-linear de uma função de custo, que mede o erro quadrático médio produzido pela saída da rede neural frente a uma saída desejada, pode-se classificá-lo de acordo com o método de otimização não-linear aplicado. Assim sendo, tem-se os chamados métodos de primeira ordem, como por exemplo o método do gradiente, onde é utilizado apenas a informação do gradiente da função de erro para ajustar os pesos da rede, e os métodos de segunda ordem, destacando-se o método do gradiente escalonado proposto por Moller em 1993, onde além do vetor gradiente da função objetivo, é feito uso também da matriz hessiana da função erro [18].

3. Rede Neural MLP Utilizada

A rede neural artificial MLP, utilizada no gerador de trajetórias, é responsável pela seleção de uma nova direção de movimento durante a geração dos pontos da trajetória, que vai fazer com que um cursor (simulação de um robô móvel) possa atingir uma posição alvo pré-definida, partindo de uma posição inicial também previamente estabelecida.

Na figura 1 tem-se a representação da rede, com suas respectivas camadas de entrada, intermediária e de saída. Sua arquitetura, por simplicidade, é referida como uma rede 28-20-2, isto é, 28 neurônios de entrada, 20 neurônios escondidos e 2 neurônios de saída.

Na figura também estão representadas suas conexões de realimentação e suas respectivas unidades de atraso.

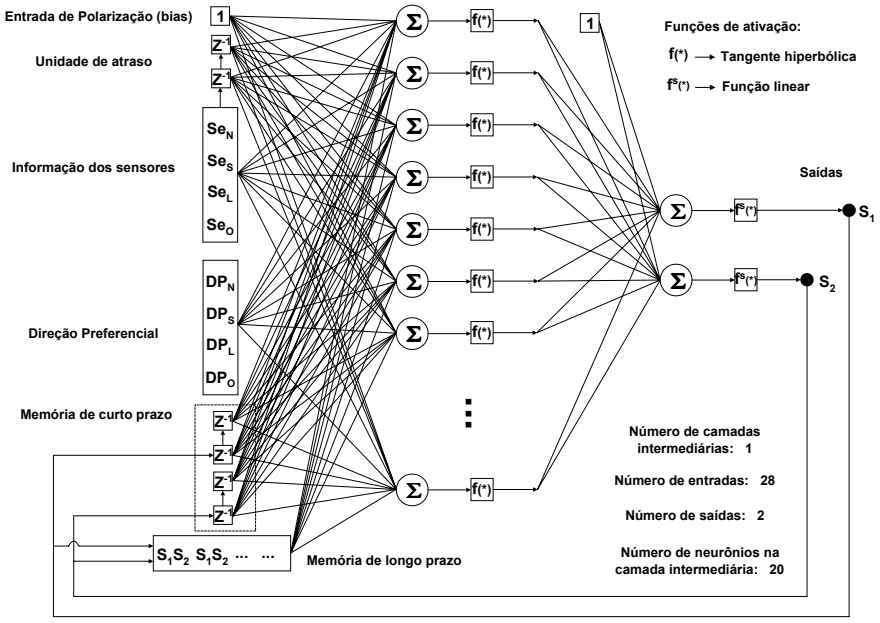

Figura 1: Arquitetura da rede neural MLP utilizada

3.1. Entradas da rede neural

Informação dos sensores. Foi simulado o uso de sensores de posição, cuja função é fornecer a distância do cursor até a próxima colisão (obstáculo). **Se $_{N,S,L,O}$** corresponde a informação dos sensores nas direções possíveis de movimento (Norte, Sul, Leste e Oeste), informando a rede quais as direções estão livres para a movimentação do cursor. Por exemplo, se **Se $_{N}$** for igual a zero significa que a direção norte está livre. Quando ocorre a detecção de um obstáculo numa dada direção o valor informado a rede é 1. Devido a importância deste tipo de informação para a rede, foi inserido também como entrada, um atraso do vetor (com dimensão 4) de informação dos sensores, ou seja, foi considerada também a informação dos sensores na iteração "k-1". O mesmo ocorre com o segundo atraso também incluído como entrada da rede. Neste caso a informação é da iteração "k-2".

Direção preferencial. Outra informação importante para a rede é a direção preferencial que o cursor deve adotar para ir de um determinado ponto no ambiente para a posição alvo. O programa identifica quais as direções que o cursor pode seguir para atingir o seu alvo. Neste caso, será usado o valor zero para identificar uma dada direção preferencial de movimento e o valor 1 para as outras direções.

Memória de curto prazo. No caso desta entrada da rede, são considerados os atrasos da saída da rede, informando para a mesma as duas últimas direções adotadas pelo cursor. Cada saída da rede vai realimentar, servindo como uma nova entrada, a rede na próxima iteração. Da mesma forma que na informação dos sensores, foi considerado também um segundo atraso das saídas, tem-se na chamada memória de curto prazo as informações de saída da rede das iterações "k-1" e "k-2", respectivamente.

Memória de longo prazo. A chamada memória de longo prazo guarda e informa para a rede quais foram as quatro últimas direções adotadas pelo cursor. Neste caso não ocorre repetição de direção, e pode-se afirmar que a memória de longo prazo funciona como um *buffer* das direções, ativado no momento que ocorre a mudança para uma nova direção. A importância desta entrada da rede foi verificada principalmente para os casos de ambientes com obstáculos formando um zigue-zague.

Em síntese, a rede neural decide qual a direção a ser seguida considerando a disposição dos obstáculos a sua volta (*Informações dos sensores*), o objetivo a ser alcançado (*Direção preferencial*) e qual o caminho seguido até o momento (*Memórias de longo e curto prazo*).

3.2. As saídas da rede neural

As duas saídas da rede neural **S1** e **S2** determinam qual a direção que o cursor deve adotar de acordo com os padrões de entrada recebidos pela mesma. Essas informações de saídas são binárias e estão representadas na tabela 1.

Tabela 1: Direção versus saídas da rede

Direção:	Saída:	
	S1	**S2**
Norte	1	1
Sul	0	0
Leste	0	1
Oeste	1	0

3.3. Treinamento da rede neural

Para o treinamento da rede neural foi aplicado o método do gradiente (método de primeira ordem) para um conjunto de treinamento com 258 padrões. A linguagem usada para a implementação do algoritmo de treinamento [15] foi a do software MATLAB® versão 5.3. No treinamento foram usados padrões de comportamentos esperados durante a navegação em alguns ambientes, cuja configuração era considerada básica, como por exemplo os representados pelas figuras 4, 5 e 6. Estes padrões foram obtidos a partir de uma série de regras pré-definidas pelo instrutor. Na tabela 2 tem-se alguns exemplos destes padrões.

Tabela 2: Exemplos de padrões usados no treinamento

Entrada:	Saída:
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1	0 0
0 1 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 1 1 0 1 0 0	1 1
0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 1 1 0 1 0 0	0 1
0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 1	1 0

O algoritmo de retropropagação foi executado de acordo com o modo padrão-a-padrão ou instantâneo (*pattern mode*), onde os pesos são atualizados após a apresentação de cada padrão à rede neural [17]. Foi utilizada também uma taxa de aprendizado fixa com valor igual a 0.1, sendo adotada a taxa de variação absoluta no erro quadrático médio por época para o critério de convergência com valor igual a 0.05.

A inicialização dos pesos da rede foi feita de maneira aleatória, com média zero e distribuição uniforme no intervalo [-0.1, +0.1].

Como o número de padrões adotados no treinamento não era muito grande, o método do gradiente com taxa de aprendizado fixa foi suficiente para que ocorresse a convergência, pois se trata de um algoritmo de custo computacional relativamente baixo. Em contra partida, o tempo de treinamento foi elevado, cerca de 5 horas para 1505 épocas (Pentium III, 550 MHz).

4. Gerador de Trajetórias

O programa chamado de Gerador de Trajetórias encontra um caminho que realiza a conexão entre uma configuração (posição e orientação) inicial e uma final, ou seja, gera os pontos intermediários entre as posições inicial e final (alvo) previamente definidas, de acordo com o ambiente onde estão inseridos estes pontos. Desta forma, a presença ou não de obstáculos, a localização e as dimensões dos mesmos são informações iniciais importantes para o programa.

Um elemento fictício, que foi denominado de "cursor", dotado de sensores e com a capacidade de deslocar-se pela área de trabalho, realizará movimentos reconhecendo regiões "livres de obstáculos" e caminhos que poderão ser utilizados por um mecanismo físico (robô móvel) [2]. O cursor ao qual foi dado um formato quadrangular, com dimensões reduzidas (10x10 *pixels*), parte de um ponto inicial e desloca-se (com um avanço fixo), adotando uma direção de movimentação determinada pela rede neural, para um novo ponto. Em seguida, esse novo ponto passa a ser considerado como ponto inicial e assim sucessivamente, até que um desses pontos da trajetória faça parte do conjunto de pontos que formam uma região de tolerância (circular) e que tenha como centro o ponto final definido anteriormente. A existência dessa região circular deve-se ao fato de que o objetivo do gerador de trajetórias é determinar uma série de pontos intermediários entre os pontos inicial e final, e não achar um ponto que tenha as mesmas coordenadas do ponto alvo. Essa região pode ser considerada como uma região de tolerância, pois o cursor se movimenta com um avanço fixo e diferente de 1. Assim, é muito improvável que um ponto da trajetória coincida com o ponto alvo.

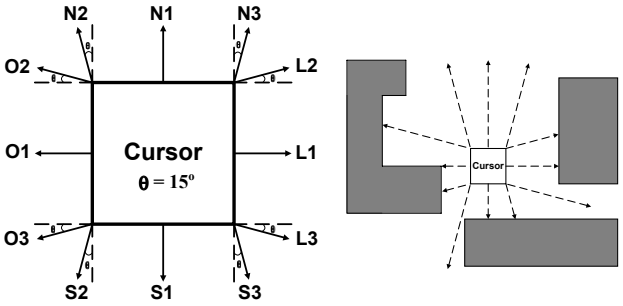

Figura 2: Cursor e a leitura dos sensores

Para a localização do cursor foi adotado um sistema de referências cartesiano ortogonal no plano, onde a origem desse sistema não necessariamente deve coincidir com a posição de início da trajetória. Também para a sua localização e mapeamento do ambiente foi utilizado um conjunto formado por 12 sensores de posição, agrupados três a três nas direções de movimento. A combinação do valor dos sensores fornece subsídios para a rede neural que seleciona a direção do movimento [2]. A figura 2 mostra o cursor, a posição dos sensores e as suas direções de medidas.

Por fim o programa salva os pontos da trajetória juntamente com os pontos inicial e final num vetor para que os mesmos possam ser utilizados posteriormente, como por exemplo na primeira etapa de um outro programa denominado *Kernel*, desenvolvido utilizando-se uma estrutura de programação orientada à objetos através da linguagem C++. Este programa é utilizado entre outras aplicações para simular o movimento de robôs móveis em ambientes desestruturados, ou seja, área de trabalho cuja estrutura física e seus elementos agregados são desconhecidos, e que numa etapa posterior vai contribuir para a implementação da "estrutura neural para o aprendizado" proposta por Mendeleck em 1995 [2] e cujo procedimento está dividido em 4 fases: identificação do meio, gerar pequenas rotas, sequenciar essas rotas formando macro rotas e verificar "vias expressas".

5. Simulações

A simulação da movimentação do robô móvel por um ambiente desestruturado, sendo conhecidas as suas posições inicial e alvo, usando o Gerador de Trajetórias, foi feita no próprio programa. Um gráfico, representando o ambiente juntamente com o cursor, era gerado para cada posição do mesmo, dando uma idéia de movimento do cursor. O ambiente com um formato retangular possui dimensões fixas iguais a 650x340 *pixels*.

A seguir serão mostrados os resultados finais da simulação, destacando a representação dos ambientes utilizados para o treinamento da rede neural na figuras 4, 5 e 6, e outros ambientes que serviram para testar a generalização dos resultados da rede representados pelas figuras 7, 8, 9 e 10.

Na figura 3 tem-se o início da simulação para um ambiente sem obstáculos. Estão representados, também, os pontos inicial e final da trajetória, bem como a região de tolerância do ponto final.

Os ambientes representados nas figuras 4, 5 e 6 foram utilizados para o treinamento da rede neural, devido ao fato de suas configurações terem sido consideradas as mais básicas e representativas dentre as configurações geralmente encontradas [2].

As figuras 7, 8 e 9 representam outros ambientes derivados dos que foram usados para treinar a rede neural. Estes ambientes serviram para testar os resultados do aprendizado e comprovar a eficiência do programa.

No caso da figura 10 está representada a trajetória do cursor na presença de um obstáculo móvel, cujo movimento apresentava, num determinado momento, a mesma direção do cursor porém em sentido contrário e com uma velocidade 4 vezes menor que a do cursor. A resposta do programa foi adequada, pois o cursor contornou o obstáculo móvel sem que ocorresse colisão, como está mostrado na posição 1 na figura.

5.1. Resultados das Simulações

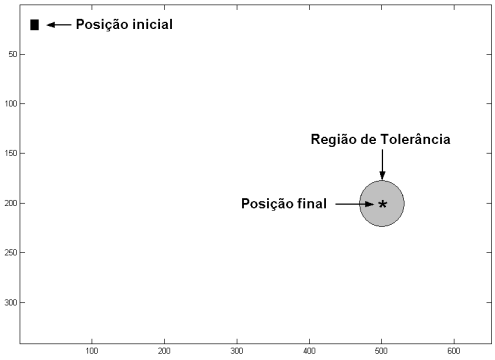

Figura 3: Região sem obstáculos

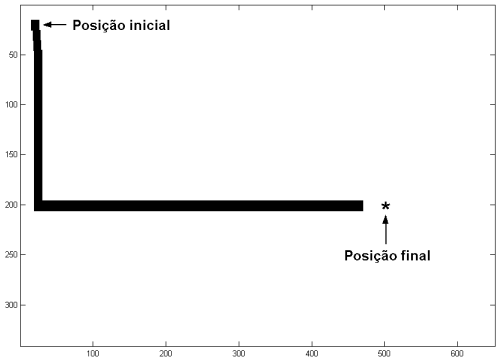

Figura 4: Região sem obstáculos

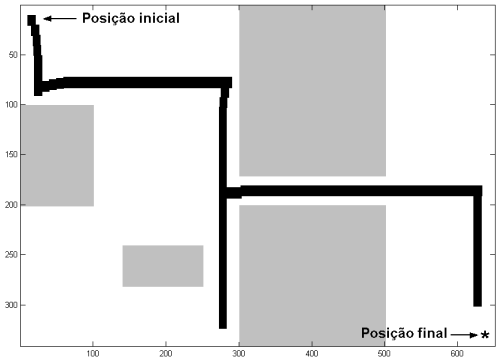

Figura 5: Região com obstáculos

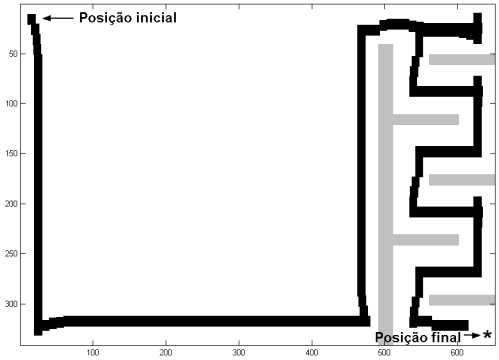

Figura 6: Região com zigue-zague

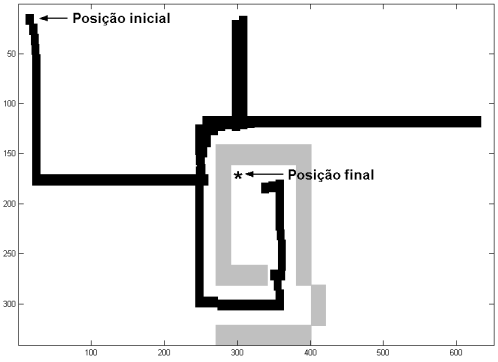

Figura 7: Região côncava

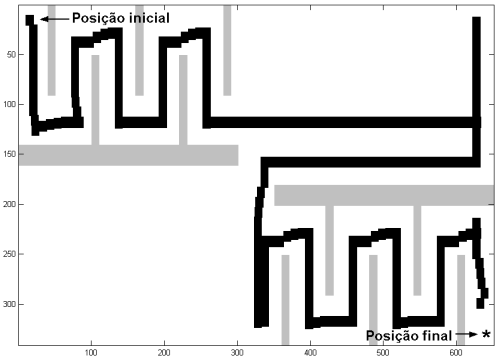

Figura 8: Região com zigue-zague

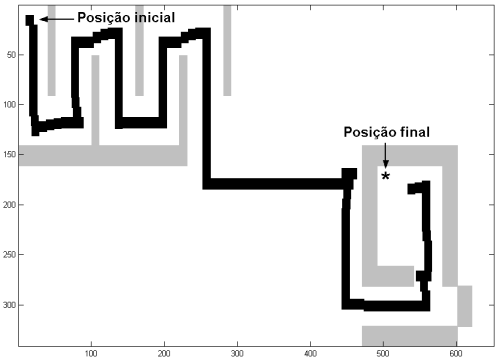

Figura 9: Região côncava + zigue-zague

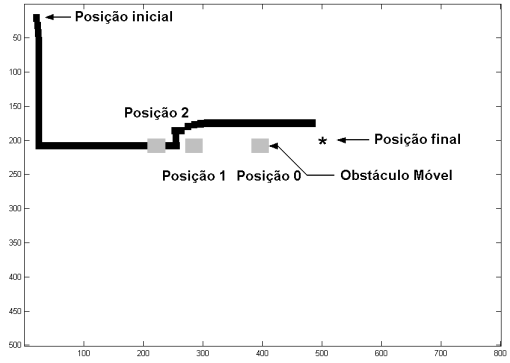

Figura 10: Região com obstáculo móvel

6. Conclusão

Neste trabalho foi proposto um sistema para a geração de trajetórias, utilizando redes neurais artificiais, aplicado na navegação de robôs móveis, permitindo que os mesmos possam se deslocar de uma maneira mais autônoma entre dois determinados pontos no seu ambiente de trabalho considerado desestruturado.

O sistema apresentou bons resultados, comprovados pelas simulações, permitindo que, numa etapa posterior, o mesmo possa ser testado em robôs móveis reais.

Por desvantagens da metodologia empregada têm-se o fato da locomoção ser restrita apenas a quatro direções (Norte, Sul, Leste e Oeste) e o custo computacional do treinamento da rede neural. Dependendo do número de neurônios da camada intermediária e da quantidade de padrões usados, o tempo de processamento cresce expressivamente.

A principal vantagem da metodologia apresentada reside no fato de que o robô pode ter um nível de aprendizado proporcional a quantidade de padrões utilizados no treinamento da rede neural, isto é, a depender da complexidade da sua aplicação, o sistema necessitará de um número menor ou maior de padrões de treinamento.

Destaca-se também, a simplicidade do sistema e o seu baixo custo, tanto de implementação quanto computacional, visto que, após o treinamento os pesos são transferidos para uma rede neural inserida no Gerador de Trajetórias e esta rede, através do processamento direto das informações de entrada, decide qual a nova direção deverá ser adotada. Seu baixo esforço computacional decorre da utilização de um único passo computacional (*forward*) e também pelo fato das informações serem fornecidas à rede na forma de binários.

Sua aplicação futura pode estar no campo dos robôs móveis usados para serviços em escritórios, como o transporte de correspondência; em bibliotecas, arquivando livros nas suas respectivas estantes; em feiras de eventos, servindo como guia; etc.

Referências Bibliográficas

[1] J. M. Vaz e João A. Fabro. SNNAP – Sistema Neural de Navegação em ambientes Pré-Mapeados. *Proceedings of the IV Brazilian Conference on Neural Networks*, pp. 118-123, 1999.

[2] André Mendeleck. Um Modelo Conexionista para a Geração de Movimentos Voluntários em Ambiente Desestruturado. *FEM / UNICAMP,* Campinas-SP, 1995, Tese de Doutorado.

[3] João A. Fabro. Grupos Neurais e Sistemas Nebulosos: Aplicação à Navegação Autônoma. *FEE / UNICAMP,* Campinas-SP, 1996, Tese de Mestrado.

[4] M. A. Salichs and L. Moreno. Navigation of Mobile Robots: Open Questions. *Robotica,* Vol. 18, Nº 3, pp. 227-234, 2000.

[5] Charles C. Chang and Kai-Tai Song. Environment Prediction for a Mobile Robot in a Dynamic Environment. *IEEE Transactions on Robotics and Automation,* Vol. 13, Nº 6, pp. 862-872, December 1997.

[6] Jason A. Janét, et al. Autonomous Mobile Robot Global Self-Localization Using Kohonen and Region-Feature Neural Networks. *Journal of Robotic Systems,* Vol. 14, Nº 4, pp. 263-282, December 1997.

[7] Malur K. Sundareshan and Thomas A. Condarcure. Recurrent Neural-Network Training by a Learning Automaton Approach for Trajectory Learning and Control System Design. *IEEE Transactions on Neural Networks,* Vol. 9, Nº 3, pp. 354-368, May 1998.

[8] H. Maaref and C. Barret. Sensor-based Fuzzy Navigation of an Autonomous Mobile Robot in an Indoor Environment. *Control Engineering Practice,* Vol. 8, pp. 757-768, 2000.

[9] A. Stafylopatis and K. Blekas. Autonomous Vehicle navigation Using Evolutionary Reinforcement Learning. *European Journal of Operational Research,* Vol. 108, pp. 306-318, 1998.

[10] M. -R. Akbarzadeh-T, et al. Soft Comuting. for Autonomous Robotic Systems. *Computers and Electrical Engineering,* Vol. 26, pp. 5-32, 2000.

[11] Wei Li, Chenyu Ma and F. M. Wahl. A Neuro-Fuzzy System Architecture for Behavior-Based Control of a Mobile Robot in Unknown Environments. *Fuzzy Sets and Systems,* Vol. 87, pp. 133-140, 1997.

[12] Dilip K. Pratihar, et al. A Genetic-Fuzzy approach for Mobile Robot Navigation Among Moving Obstacles. *International Journal of Approximate Reasoning,* Vol. 20, pp. 145-172, 1999.

[13] Hans J. S. Feder, et al. Adaptive Mobile Robot Navigation and Mapping. *The International Journal of Robotics Research,* Vol. 18, Nº 7, pp. 650-668, July 1999.

[14] Leopoldo Jetto, Sauro Longhi and Giuseppe Venturini. Development and Experimental Validation of na Adaptive Extended Kalman Filter for the Localization of Mobile Robot. *IEEE Transactions on Robotics and Automation,* Vol. 15, Nº 2, pp. 219-228, April 1999.

[15] Fernando J. Von Zuben. Redes Neurais. *FEEC / UNICAMP,* Campinas-SP, Segundo semestre 1999, Notas de aulas.

[16] S. Haykin. Neural Networks – A Comprehensive Foundation. Second Edition. Prentice Hall, 1999.

[17] Eduardo M. Iyoda. Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas. *FEEC / UNICAMP,* Campinas-SP, 2000, Dissertação de Mestrado.

[18] Leandro N. de C. Silva. Análise e Síntese de Estratégias de Aprendizado para Redes Neurais Artificiais. *FEEC / UNICAMP,* Campinas-SP, 1998, Dissertação de Mestrado.