Controle de um Robô do tipo *differential drive* através de Lógica Fuzzy e Redes Neurais

FELIPE A. W. BELO, MARCO A. MEGGIOLARO, CARLOS HALL, RICARDO TANCHEIDT

ICA : Laboratório de Inteligência Computacional Aplicada

Departamento de Engenharia Elétrica

Pontifícia Universidade Católica do Rio de Janeiro, PUC-Rio

Rua Marquês de S. Vicente 225, Gávea, Rio de Janeiro, CEP 22453-900, RJ, Brasil

Palavras Chave: Inteligência Artificial, Lógica Fuzzy, Redes Neurais, Robótica, Controle, Differential Drive .

1 – Resumo

O problema aqui apresentado é o de se determinar a ação de um robô do tipo *differential drive* para se chegar a uma posição desejada. Possíveis soluções podem ser determinadas através da cinemática indireta do robô, porém, as soluções aqui empregadas utilizam Lógica *Fuzzy* e Redes Neurais. Para aplicação e teste das técnicas propostas foi desenvolvido um simulador para o robô.

O Controlador Fuzzy desenvolvido trabalha com regras arbitradas. Já a Rede Neural, é treinada através de um histórico gerado pelo controle humano do robô.

São propostos e testados diferentes Controladores Fuzzy. O mesmo acontece com Redes Neurais. Diferentes Redes são treinadas e aqui apresentadas.

Foram obtidos bons resultados com as duas técnicas e são feitas diversas propostas visando resultados ainda melhores.

2 – Introdução

O uso de robôs móveis tem se popularizado nos últimos anos com o aparecimento de robôs pessoais e privados. Do uso doméstico, que inclui aspiradores robóticos autônomos, a robôs para entretenimento, que incluí de brinquedos (como o AIBO da Sony) a kits de robótica (como o *Mindstorms* da LEGO), é visível que a robótica não está mais restrita ao cenário industrial. Se a tecnologia para robôs de uso pessoal crescer como promete, com preços competitivos, e boa aceitação comercial, este pode vir a ser um grande mercado em um futuro próximo.

Grandes avanços têm sido feitos na área de robôs móveis pessoais como pode ser visto em (Vivek, A.S. e Meggiolaro, M.A. [3]). Apesar dos custos envolvendo *hardware* terem decrescido, fatores econômicos vão requerer a substituição de arquiteturas complexas por sistemas inteligentes e de baixo custo.

Robôs do tipo *differential drive* são possivelmente os de mecânica mais simples dentre os robôs móveis, tornando-os bastante atrativos dentro do panorama atual. Dentre outras vantagens, estes robôs possuem características vantajosas na locomoção em ambientes estreitos devido a sua cinemática. O design *differential drive* é muito comum em robôs pequenos, de baixo custo e para uso *indoor*.

Além do design, outras características são de grande importância quando se trata de robôs móveis. A criação de robôs autônomos é de maior relevância em robótica e requer o planejamento e execução do movimento de maneira automática, como pode ser verificado em (Latombe, J.C. [11]). Porém, a inteligência operacional da qual o ser humano é dotado, e com a qual lida de modo inconsciente para interagir com o ambiente, perceber o mesmo e planejar o movimentos, é de difícil reprodução por um programa de computador.

Do desafio de se criar uma inteligência artificial inspirada na inteligência humana, a Inteligência Computacional é uma área da ciência que busca reproduzir aspectos humanos e da natureza, tais como: aprendizado, reprodução, raciocínio, evolução e adaptação.

Dentre suas técnicas, as Redes Neurais compreendem modelos computacionais não lineares, inspirados no funcionamento do cérebro humano, procurando reproduzir, dentre outras características, o aprendizado. Das várias áreas onde Redes Neurais Artificiais são empregadas, uma miríade de aplicações estão relacionadas a veículos inteligentes e robótica. Algumas destas aplicações podem ser encontradas em (Pomerlau, D.A [13]). Provavelmente, a aplicação mais famosa é o sistema ALVINN [13], usado no controle de movimento do véiculo ALV(*Autonomous Land Vehicle*) da *Carnegie-Mellon University*. ALVINN utiliza uma Rede Neural para controlar a direção do veículo a partir de imagens obtidas por câmeras e sensores de distância. O aprendizado desta rede é feito através do histórico da direção realizada por uma pessoa. Este tipo de treinamento é parecido com o que será apresentado aqui.

Outra técnica da Inteligência Computacional que encontra um vasto campo de aplicações em robótica é a Lógica *Fuzzy*. A Lógica *Fuzzy* trabalha com a modelagem do raciocínio aproximado do ser humano, sendo capaz de inferir soluções em ambientes não lineares e imprecisos. Um grande número de aplicações relacionadas à robótica trabalha com o uso de Sistemas *Fuzzy* como controladores (Mendel, J.M. [7]), uso que aqui também será explorado.

O problema aqui apresentado é o de se determinar a ação de um robô do tipo *differential drive* para se chegar a uma posição desejada. Este é um problema de controle usual onde se deseja determinar a atuação dos motores do robô sabendo-se de seu estado e sua posição relativa à posição de destino. Soluções possíveis podem ser determinadas através da cinemática indireta do robô (Dudek, G. e Jenkin, M. [2]), porém, as soluções aqui empregadas utilizam Lógica *Fuzzy* e Redes Neurais. A aplicação destas técnicas permite a extensão deste projeto para a solução de problemas mais complexos sem a necessidade de ampla modelagem matemática dos mesmos. Há inclusive a pretensão de futura implementação destas técnicas em conjunto com o algoritmo de modelagem e navegação proposto em (Vivek, A.S. e Meggiolaro, M.A. [3]).

Tanto o sistema *Fuzzy* quanto a Rede Neural aqui desenvolvidos são utilizados como controladores do robô em um ambiente simulado. A maior diferença entre as duas técnicas está em a Rede Neural ser treinada através de um histórico gerado pelo controle humano do robô (em simulação), e a lógica *Fuzzy* ter suas regras arbitradas através do conhecimento e familiaridade com o problema.

Para aplicação e teste das técnicas propostas foi desenvolvido um simulador para o robô, já visando seu uso futuro junto ao algoritmo citado anteriormente e proposto em (Vivek, A.S. e

Meggiolaro, M.A. [3]). Um grande número de simuladores de robôs móveis tem sido desenvolvido nos últimos anos, alguns deles simulando robôs do tipo *differential drive*. A opção de desenvolver um novo simulador ao invés de se usar algum existente vem da possibilidade de se trabalhar os problemas aqui abordados, e os problemas a serem desenvolvidos em trabalhos futuros, levando em consideração suas particularidades.

3 – Controle de um robô do tipo *differential drive*

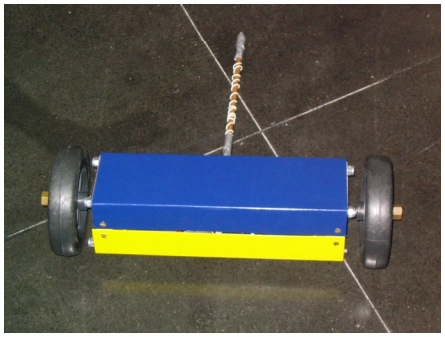

Figura 1. Robô Sentinela. Robô do tipo differential drive desenvolvido na PUC-Rio.

Descrição e cinemática de um robô do tipo *differential drive* podem ser encontrados em (Bräun, T. [1]) ou (Dudek, G. e Jenkin, M. [2]).

O design *differential drive* é definido por uma arquitetura com dois motores, de eixo em comum, montados em posições fixas nas laterais de um robô, e controle independente para cada roda. Devido à necessidade de três pontos de contato com o solo, utiliza-se uma ou mais rodas ou pontos de apoio passivos.

Este design é possivelmente o de mecânica mais simples dentre os robôs móveis. Porém, o seu controle e sua física possuem certa complexidade devido à coordenação das duas rodas ser independente. O design *differential drive* costuma ser utilizado em robôs pequenos, de baixo custo e ou para uso *indoor*, tal como o robô *Khepera* [6].

Sua cinemática lida com a relação entre as velocidades das duas rodas. Com as duas rodas em mesma velocidade e direção, o robô se move para frente ou para trás. Para uma roda com velocidade superior a outra, o robô anda em curva ao longo do um arco de um círculo. Para os motores com mesma velocidade e direções opostas, o robô roda em torno do centro do eixo comum as duas rodas.

Ao serem aplicadas velocidades aos motores, o robô roda em torno de um ponto comum ao eixo de suas rodas. Variando-se a velocidade das rodas, este ponto pode ser mudado alterando as trajetórias escolhidas. Este ponto de rotação será definido como *ICC*. O problema está ilustrado na figura 2.

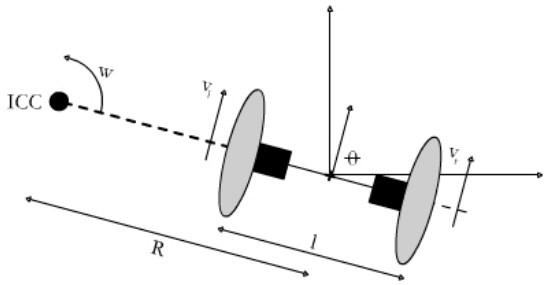

Figura 2.

Em determinado momento do tempo, o robô segue um caminho em torno de ICC com uma velocidade angular comum w :

$$w\left(R + l/2\right) = v_r,$$
$$w\left(R + l/2\right) = v_l$$

Onde:
l – Distância entre as duas rodas;
v_l – Velocidade da roda esquerda;
v_r – Velocidade da roda direita;
R – Distância do ponto ICC para o ponto central entre as duas rodas;

Em determinado instante no tempo, resolvendo-se para R e para w:

$$R = \frac{l}{2}\frac{(v_l + v_r)}{(v_r - v_l)},$$
$$w = \frac{v_r - v_l}{l}$$

As equações acima levam a algumas situações interessantes. Para $v_r = v_l$, tem-se R infinito,e portanto, o robô move-se em linha reta. Para $v_r = - v_l$, R é igual a zero e portanto o robô roda em torno do ponto central entre as duas rodas. Esta característica torna este tipo de robô atraente para navegação em ambientes estreitos. Porém, sua cinemática impede determinados movimentos, tal como sua navegação sobre o eixo comum à suas rodas.

Um problema deste design é a sensibilidade relacionada às velocidades das duas rodas. Pequenos erros podem levar a diferentes trajetórias, limitando sua aplicabilidade.

3-1 – Cinemática direta

Considerando a posição do robô em (*x, y*) direção θ (em relação ao eixo *x*) e velocidades v_r e v_l, durante um período $t \rightarrow \partial t$, tem-se:

$$ICC = [x - R\sin(\theta), y + R\cos(\theta)],$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(w\partial t) & -\sin(w\partial t) & 0 \\ \sin(w\partial t) & \cos(w\partial t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ w\partial t \end{bmatrix}$$

Para o caso particular em que $v_r = v_l = v$, a equação é simplificada para:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + v\cos(\theta)\partial t \\ y + v\sin(\theta)\partial t \\ \theta \end{bmatrix}$$

E para o caso em que $v_r = -v_l = v$, obtém-se:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + 2v\partial t / l \end{bmatrix}$$

As equações apresentadas acima foram às utilizadas na construção do simulador desenvolvido.

Integrando-se tais equações a partir de uma condição inicial (x_0 ,y_0 , θ_0), é possível computar a posição do robô em qualquer instante *t*. Em contrapartida a cinemática direta, pode-se também calcular a cinemática inversa, para a qual, dado um instante *t* e posição desejada *x* e *y* (sem controle independente de θ), encontra-se v_r e v_l. Há porém, infinitas soluções para v_r e v_l. Neste projeto, não se procura traçar uma trajetória calculada para se chegar a uma posição desejada, mas sim, se utilizar Lógica *Fuzzy* e Redes Neurais para tal objetivo. O uso destas técnicas não implica em um melhor resultado, mas possibilita através de variações na implementação, do trato de problemas semelhantes, envolvendo outras complicações, e de modelagem matemática difícil.

3-2 – Lógica Fuzzy

Em (Mendel, J.M. [7]) é apresentado uma proposta para controladores *fuzzy* de caráter geral onde as entradas do sistema *fuzzy* são usualmente o erro entre o sinal de referência e a saída da planta, e a variação deste erro. A saída do controlador é geralmente uma variação no controle.

Outras propostas, como em (Tanscheit, R. [8]) sugerem como entradas parâmetros específicos do problema.

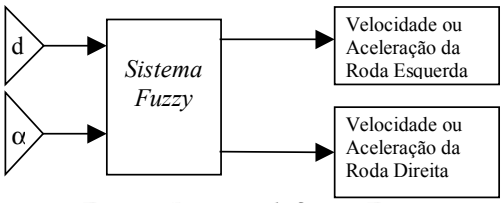

Figura 3. Diagrama do Sistema Fuzzy

No problema aqui apresentado, foram escolhidas como entradas do Sistema Fuzzy a distância cartesiana *d* entre o robô e a posição destino, e o ângulo α entre o robô e a posição destino. O sistema está exemplificado na figura 3.

Para as saídas do sistema, foram exploradas duas possibilidades. Na primeira, as acelerações relativas aos motores são as saídas do Sistema *Fuzzy*. Esta é a escolha mais coerente com uma implementação real. Porém, não é a mais coerente com as entradas escolhidas. Como segunda possibilidade, as velocidades das rodas do robô são as saídas do sistema.

Nos dois casos, têm-se duas variáveis de saída, uma associada ao motor esquerdo e outra ao motor direito. A segunda possibilidade não trata de um cenário real, mas sim ideal, pois não se impõe a velocidade de um motor diretamente. Esta possibilidade pode ser coerente se as velocidades encontradas na saída do sistema *fuzzy* forem utilizadas como entradas de novos controladores, que poderiam ser *fuzzy* ou não. O uso destes novos controladores atrelados ao sistema *fuzzy* não foi simulado, ficando portanto proposto para futuros trabalhos. Nas simulações onde as velocidades foram utilizadas como saídas do sistema, definiu-se o cenário como ideal, sendo as velocidades diretamente aplicadas as rodas.

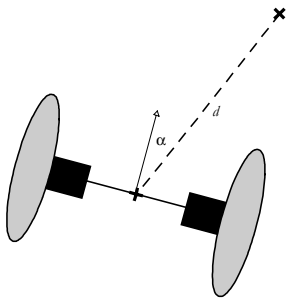

Figura 4. Robô em relação a posição desejada

A distância *d* e o ângulo α podem ser vistos na figura 4 e foram definidos como:

$$d = \sqrt{(x-x')^2 + (y-y')^2},$$

$$\alpha = \tan^{-1}\left(\frac{y'-y}{x'-x}\right) - \theta$$

Onde x e y são as coordenadas cartesianas do centro do eixo entre as rodas do robô, θ é a direção do robô em relação ao eixo x e x' e y' são as coordenadas da posição de destino.

Para simplificar o problema, as variáveis de entrada e saída foram tratadas por universos discretos e normalizados, apresentados abaixo:

$$U_d = \{0,1,2,3,4,5,6\},$$
$$U_\alpha = \{-180,-120,-60,0,60,120,180\},$$
$$U_{out} = \{-6,-4,-2,0,2,4,6\}$$

A distância d foi escalonada entre 0 e uma distância máxima. As saídas foram escalonadas entre velocidades mínimas e máximas para o caso das saídas serem as velocidades nas rodas, ou acelerações mínimas e máximas para o caso das saídas serem as acelerações.

Os conjuntos *fuzzy* foram definidos de forma aproximadamente triangular , simétricos, e com graus de pertinência {0,3; 0,7; 1; 0,7; 0,3}. Os conjuntos fuzzy podem ser vistos nas figuras 5,6 e 7.

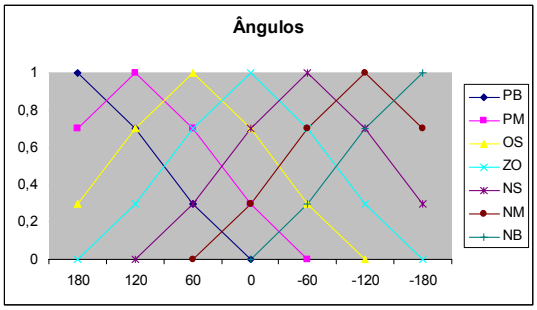

Figura 5. Conjuntos Fuzzy relacionados ao ângulo

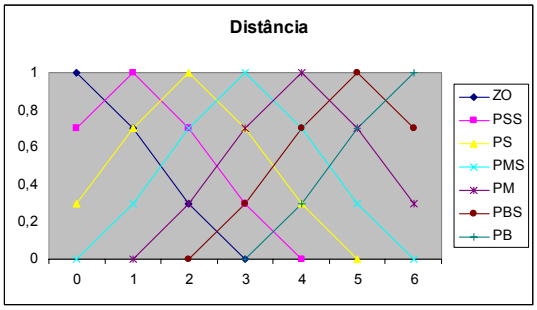

Figura 6. Conjuntos Fuzzy relacionados a distância

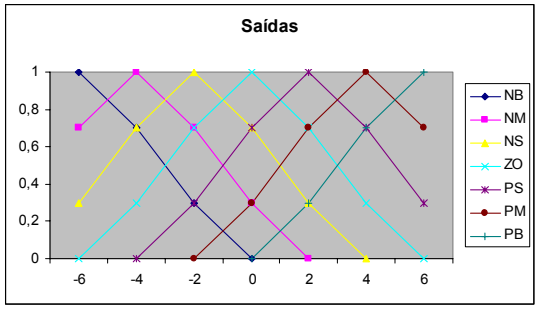

Figura 7. Conjuntos Fuzzy relacionados as saídas

O armazenamento das regras foi feito como proposto em (Mendel, J.M. [7]) utilizando um número para representar os conjuntos *fuzzy* associados as variáveis de saída do sistema.

O método de *defuzificação* utilizado foi o *CoM* (*Center of Maximums*) e para operador de agregação dos antecedentes foram utilizados tanto o operador *mínimo* como o operador *produto*.

A interface do simulador para definição das regras *fuzzy* é apresentada abaixo:

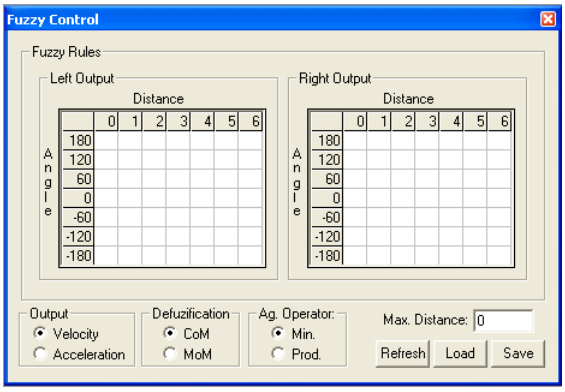

Figura 8. Interface para definição das regras

3.3 – Rede Neural

A Rede Neural é utilizada para o controle do robô de maneira similar à lógica *fuzzy*. As entradas da rede constituem-se de parâmetros da cinemática do robô no instante t e de sua posição em relação à posição desejada. As saídas são ou as acelerações ou as velocidades a serem aplicadas às rodas do robô.

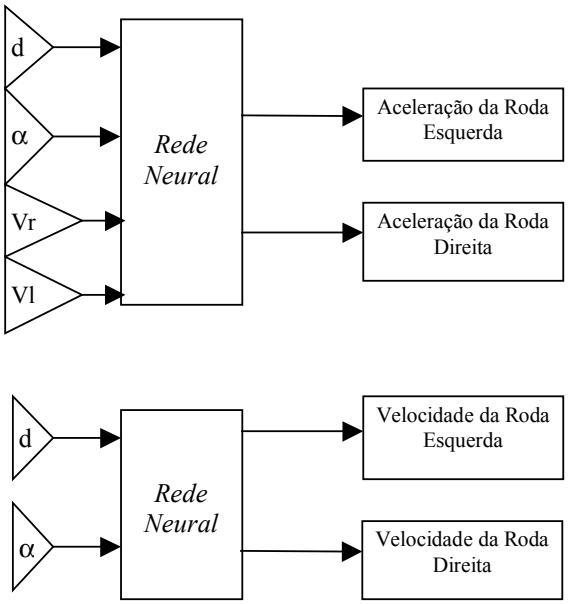

Figura 9. Redes Neurais propostas

Inicialmente, propôs-se trabalhar com quatro entradas, correspondentes à distância cartesiana *d* entre o robô e a posição destino, o ângulo α entre o robô e a posição destino e as velocidades das rodas direita e esquerda. Para esta configuração, as saídas utilizadas seriam as acelerações correspondentes às rodas. Como esta rede não estava apresentando bons resultados, a rede foi simplificada de modo a facilitar o seu ajuste na busca por melhores resultados. Posteriormente, verificou-se que os maus resultados obtidos não estavam atrelados à complexidade da rede, mas sim a outros fatores, portanto, fica proposto o uso da rede como inicialmente pensada para trabalhos futuros.

A segunda rede proposta foi a rede efetivamente testada. Sua configuração é semelhante ao Sistema *Fuzzy* utilizado. Suas entradas correspondem a *d* e a α, e suas saídas correspondem às velocidades das rodas do robô. Como para o sistema *fuzzy*, este modelo não representa a realidade fielmente, dado que as velocidades não podem ser aplicadas diretamente às rodas como já comentado. Porém, pode-se fazer como proposto para o Sistema *Fuzzy*, utilizando novas malhas de controle, para as quais as entradas seriam as velocidades inferidas pela Rede Neural. Estas malhas de controle também poderiam ser implementadas por Redes Neurais.

Nas simulações aqui descritas, admitiu-se o cenário como ideal, aplicando-se as velocidades inferidas diretamente sobre as rodas do robô. Fica proposta a simulação completa, com as malhas de controle sugeridas, para trabalhos futuros.

A rede utilizada foi do tipo *multi-layer perceptron* , com uma camada escondida. O algoritmo de treinamento empregado foi o *Back Propagation* , com treinamento incremental e utilizando-se o termo momento. Para implementação da rede, usou-se a biblioteca *Annie* [9], implementada em C++ e de código fonte aberto.

As entradas e saídas foram normalizadas entre {0,1} da seguinte maneira:

- Distância *d*: Normalizada de 0 a uma distância máxima. Valores acima desta distância foram computados como 1;
- Ângulo α: Normalizado entre 0 e 360 graus;
- Velocidades: Normalizadas entre –Max e Max, onde Max corresponde às velocidades máximas do robô;

Os padrões de treinamento foram obtidos através de corridas efetuadas e gravadas no simulador. O simulador desenvolvido permitia o controle do robô através do teclado, possibilitando o armazenamento de todos os parâmetros relacionados ao robô durante a corrida para uso posterior. Portanto, várias corridas foram gravadas, nelas, o robô foi movido para a posição destino a partir de diversas posições iniciais. Posteriormente, um grupo de padrões era selecionado a partir destes históricos e repassado para o treinamento da Rede Neural.

É interessante perceber que o método aplicado pode ser empregado em problemas mais complexos, envolvendo sensoriamento e atuadores diversos. Um problema proposto é a variação do problema aqui tratado com obstáculos. Neste problema, as entradas da rede neural poderiam incluir dados do ambiente obtidos através de sensores.

O mesmo método também pode ser aplicado em jogos no treinamento de agentes utilizando-se o histórico de partidas com jogadores reais. Em um jogo de corrida, por exemplo, os corredores virtuais (agentes) poderiam aprender com o histórico de partidas efetuadas por corredores reais.

As soluções encontradas por uma Rede Neural treinada da maneira proposta não buscam o resultado ótimo, e sim, aproximar as soluções humanas utilizadas para treinamento da rede.

A interface do simulador para trabalhar com a Rede Neural pode ser vista na figura 10.

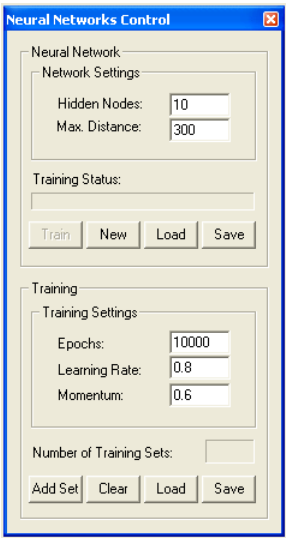

Figura 10. Interface para treinamento da Rede Neural

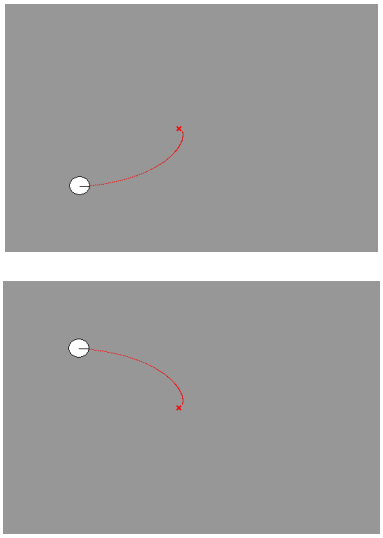

Figura 11. Trajetórias simétricas

4 – Resultados Experimentais

4.1 – Aplicando a Lógica Fuzzy

Foram experimentados diversos conjuntos de regras para o Controlador Fuzzy.

Em todos os experimentos, as regras de saída para um motor eram o espelho vertical das regras escolhidas para o outro motor. Isto é intuitivo e faz sentido pela própria cinemática do robô. O resultado desta prática é que dada uma mesma distância inicial, e para ângulos entre o robô e a posição objetivo simétricos, as saídas obtidas resultam em trajetórias também simétricas. Veja no exemplo abaixo:

Seguindo-se este critério e procurando-se definir o menor conjunto de regras fuzzy, são apresentados dois conjuntos de regras encontrados para duas situações distintas. Nestes dois experimentos, as saídas do controlador são referentes às velocidades das rodas do robô.

No primeiro conjunto de regras apresentado, buscou-se definir um controle onde o robô sempre se movesse para frente. Este conjunto é apresentado na figura 12.

Algumas trajetórias encontradas utilizando-se este conjunto de regras e o operador *mínimo* podem ser vistas na figura 13 . A distância máxima utilizada para este experimento foi de aproximadamente um quarto da tela do simulador.

Left Output

Angle \ Distance	0	1	2	3	4	5	6
180	0			-4			-6
120							
60							
0	0			4			6
-60							
-120							
-180	0			4			6

Right Output

Angle \ Distance	0	1	2	3	4	5	6
180	0			4			6
120							
60							
0	0			4			6
-60							
-120							
-180	0			-4			-6

Figura 12. Regras Fuzzy para Experiência 1

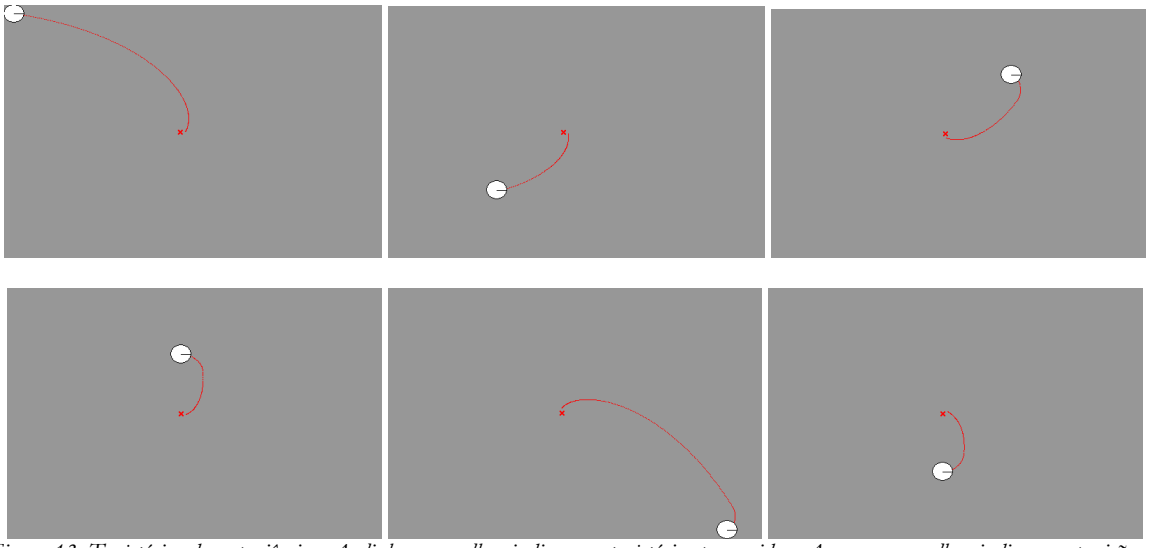

Figura 13. Trajetórias da experiência . As linhas vermelhas indicam as trajetórias percorridas. A cruzes vermelhas indicam as posições de destino.

Este conjunto de regras respondeu bem a todas as posições iniciais testadas. Os operadores *mínimo* e *produto* não levaram a diferenças significativas nas trajetórias encontradas.

Tentativas de se fazer o robô terminar mais próximo do objetivo acrescentando-se regras para os conjuntos PSS ou PS de distância não foram bem sucedidas. Estas regras faziam com que o sistema gerasse saídas diferentes de zero para distância zero, e portanto, o robô poderia não parar, muitas vezes chegando a oscilar quando próximo do objetivo. Um exemplo pode ser visto nas figuras 14 e 15.

Left Output

Angle \ Distance	0	1	2	3	4	5	6
180	0	-1		-4			-6
120							
60							
0	0	1		4			6
-60							
-120							
-180	0	1		4			6

Right Output

Angle \ Distance	0	1	2	3	4	5	6
180	0	1		4			6
120							
60							
0	0	1		4			6
-60							
-120							
-180	0	-1		-4			-6

Figura 14. Regras Fuzzy de exemplo

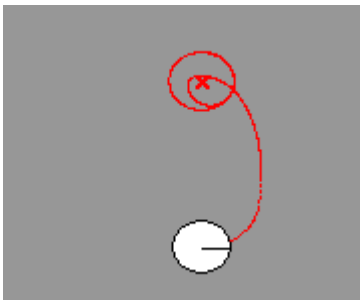

Figura 15. Trajetória de exemplo

Neste exemplo, o robô fica circulando ao redor da posição objetivo indefinidamente.

É interessante perceber que a maneira como os conjuntos fuzzy relacionados aos ângulos de entrada foram definidos favorece este controle específico.

São definidas regras diferentes para os conjuntos centrados em -180 e 180 graus, quando, na verdade, eles representam o mesmo ângulo. Do modo em que o sistema foi implementado, estas regras não se sobrepõem. Isto permite que as ações relativas aos ângulos negativos sejam desacopladas das ações relativas aos ângulos positivos.

Para que os conjuntos centrados em -180 e 180 fossem acionados ao mesmo tempo, os conjuntos Fuzzy teriam de ser definidos como demonstrado na figura 16.

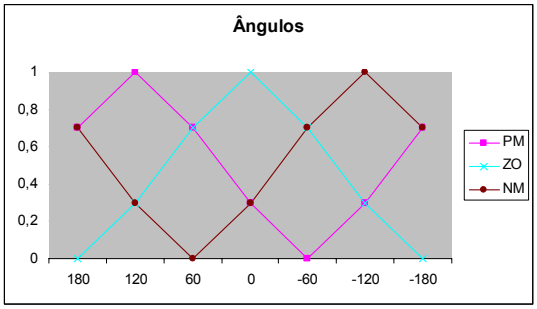

Figura 16. Proposta de Conjuntos Fuzzy para ângulo

Só estão representados 3 conjuntos para facilitar a visualização.

Na definição feita na figura 16, os conjuntos não seriam limitados pelos ângulos -180 e 180, mas sim, atravessariam de um ângulo ao outro, dado que são iguais.

Para fazer o robô andar sempre para frente, tendo-se tão poucos conjuntos relacionados ao ângulo de entrada, não seria adequado utilizar tal representação, pois não seria possível desacoplar ações relativas à ângulos positivos de ações relativas à ângulos negativos como foi feito.

Porém, para fazer o robô andar tanto para frente quanto para trás, a representação utilizada não se demonstrou ideal, sendo proposto fazer como na figura 16.

No segundo experimento se tenta fazer o robô andar tanto para frente quanto para trás. As regras escolhidas para este segundo experimento estão na figura 17 e algumas trajetórias encontradas utilizando-se o operador *produto* podem ser vistas na figura 17.

Left Output

Angle \ Distance	0	1	2	3	4	5	6
180	0			-4			-6
120				3			5
60				-4			-6
0	0			4			6
-60				4			6
-120				-3			-5
-180	0			-4			-6

Right Output

Angle \ Distance	0	1	2	3	4	5	6
180	0			-4			-6
120				-3			-5
60				4			6
0	0			4			6
-60				-4			-6
-120				3			5
-180	0			-4			-6

Figura 17. Regras Fuzzy para Experiência 2

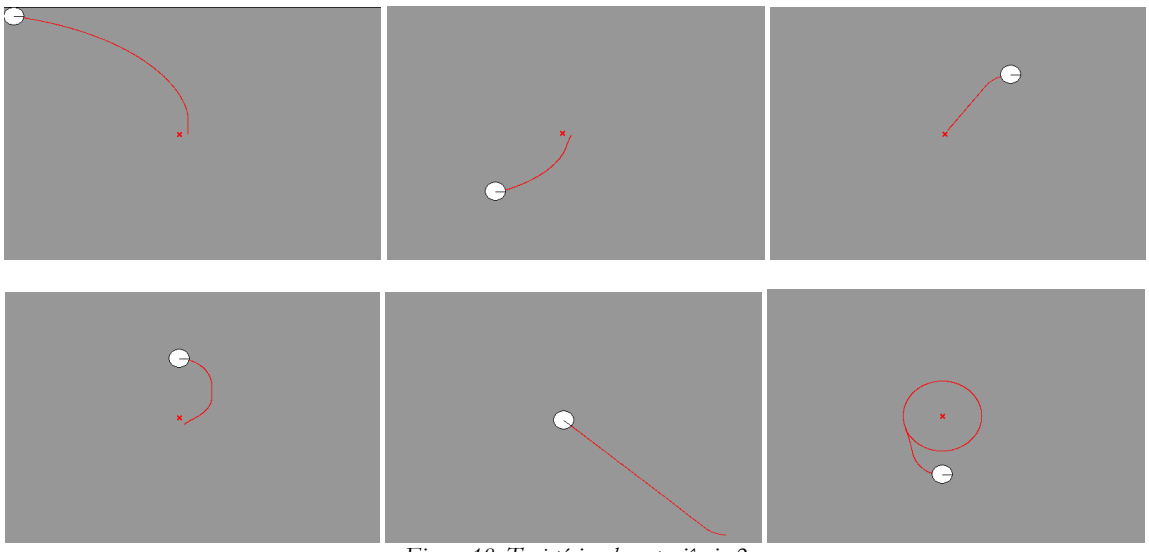

Figura 18. Trajetórias da experiência 2

Não se conseguiu trabalhar com o operador *mínimo* para obter os resultados desejados. Mesmo utilizando-se o operador *produto*, pequenas variações nas regras levavam a resultados completamente diferentes.

Este conjunto de regras respondeu bem para a grande maioria das posições iniciais testadas, mas não a todas. Além disto, o comportamento do robô não foi o mesmo para trajetórias em que este se movia para frente e trajetórias em que se movia para trás.

Além da proposta exemplificada na figura 16, ficam propostas as seguintes idéias:

- Definir um número maior de conjuntos relativos ao ângulo, melhorando a discretização do problema;
- Trabalhar com o problema não discretizado;
- Trabalhar com outros métodos de defuzificação;

Também foram feitos experimentos utilizando-se as acelerações como saídas do Sistema Fuzzy. Os resultados obtidos foram péssimos, pois o controlador gerava grandes oscilações. Para ilustrar o problema, são apresentadas duas experiências executadas:

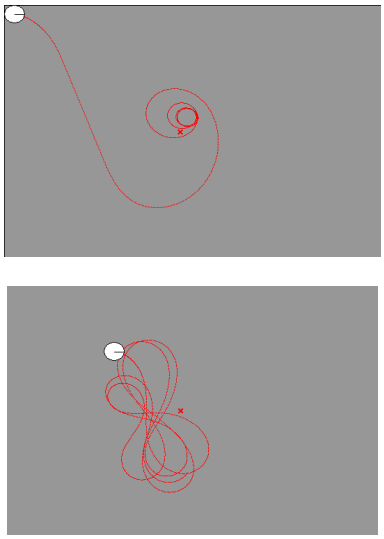

Figura 20. Trajetórias para acelerações como saídas

O controle é tão ruim porque o sistema não tem conhecimento sobre as velocidades do robô ou sobre a derivada da distância d e a derivada do ângulo α. Fica proposto para trabalhos futuros o desenvolvimento de um Controlador Fuzzy estes dados como entradas. Porém, a melhor opção deve ser trabalhar com um segundo controlador, como já proposto em *3-2 – Lógica Fuzzy*.

O Controlador Fuzzy que apresentou os melhores resultados foi o do primeiro teste. Porém, há sempre um erro entre a posição desejada e a posição obtida. Este erro é pequeno e da ordem de 5% da distância máxima. O erro pode ser diminuído ao diminuir-se a distância máxima, mas isto pode levar o robô a fazer trajetórias mais longas.

4.2 – Aplicando a Rede Neural

Foram feitos inúmeros testes para diferentes configurações de redes. Procurou-se variar também os padrões de treinamentos dados as redes, tentando entender como as redes vinham a se comportar dados padrões de treinamento com características diversas.

Foram treinadas redes com 5, 10, 15 e 20 processadores na camada escondida. Bons resultados foram obtidos com redes de 10 e 15 processadores. A rede que apresentou os melhores resultados possuía 15 processadores. A rede de 5 processadores não conseguiu modelar o problema para um número superior de padrões de treinamento (acima de 150 padrões). Já a rede de 20 processadores resultou em pior generalização.

O critério de parada do treinamento foi o número de épocas. Variou-se o número de épocas de 10000 a 20000. Valores inferiores a 10000 não produziram bons resultados. Valores superiores a 20000, além de piorarem a generalização das redes, levaram a piora da modelagem dos próprios padrões de entrada, pois a rede acabava modelando o ruído presente nos padrões.

A taxa de aprendizado foi variada entre 0.5 e 0.8. Porém, sem grandes implicações nos resultados.

A taxa de momento foi variada entre 0.4 e 0.6, também sem apresentar grandes diferenças nos resultados.

A grande dificuldade encontrada, não foi com o ajuste dos parâmetros da rede, mas sim com a escolha e apresentação dos padrões de treinamento.

Os padrões foram gerados a partir de históricos de corridas efetuadas no simulador. Estes históricos foram feitos a partir de amostragens realizadas a cada 20 ms. Portanto, uma corrida de alguns segundos resultava em centenas de padrões de treinamento. Para o treino da rede, eram feitas de 6 a 12 corridas, o que levava a alguns milhares de padrões de treinamento.

A apresentação desta enorme quantidade de padrões, além de fazer com que a rede necessitasse de um enorme tempo de treinamento, fazia com que redes pouco complexas não conseguissem aprender bem.

As redes modelavam algumas características dos padrões apresentados, mas não possuíam nenhuma capacidade de generalização.

As características aprendidas pela rede, eram aquelas relativas a padrões que se repetiam muito ao longo das corridas. Saídas muito presentes nos padrões apresentados, vinham a dominar a rede. Por exemplo, se nos padrões apresentados havia uma grande quantidade de padrões apresentando velocidade máxima nas saídas, a rede aprendia a responder com velocidade máxima para qualquer entrada apresentada.

Como inicialmente as redes utilizadas possuíam 4 entradas e acelerações como saídas, o problema era ainda mais sério. Os resultados encontrados muitas vezes não faziam sentido nenhum. Outras vezes, a rede relacionava as saídas somente as entradas relativas as velocidades, ignorando a distância d e o ângulo α.

Um agravante é o fato de que pelos padrões serem extraídos de corridas humanas, há ruído nos dados. Este ruído é inerente ao método que não é estático, e sim, dinâmico e altamente estocástico.

O problema da grande quantidade de padrões só foi constatado ao simplificar-se a rede.

Para resolver o problema foi feita uma filtragem nos padrões. A primeira medida tomada foi a de eliminar padrões com saídas repetidas, evitando que a rede fosse contaminada por saídas freqüentes. Esta medida levou a rede a começar a apresentar resultados mais coerentes. Porém, o número de padrões continuou alto.

A segunda medida tomada foi a de se escolher padrões em intervalos uniformes nos históricos. O intervalo escolhido foi de 10 padrões. O número de padrões caiu então de milhares para algumas centenas.

Os resultados aqui apresentados são referentes a uma rede de 15 processadores, 10000 épocas de treinamento, taxa de aprendizado de 0.7 e momento de 0.6. O conjunto de históricos foi feito de modo a cobrir de maneira uniforme o universo de entradas. Esta foi a rede que apresentou os melhores resultados.

Na figura 21 são apresentadas algumas trajetórias utilizadas para treinamento da rede. Foram 12 trajetórias ao todo, sempre simétricas ao redor da posição de destino que está marcada no centro da tela.

Na figura 22 são apresentadas trajetórias percorridas pela rede neural..

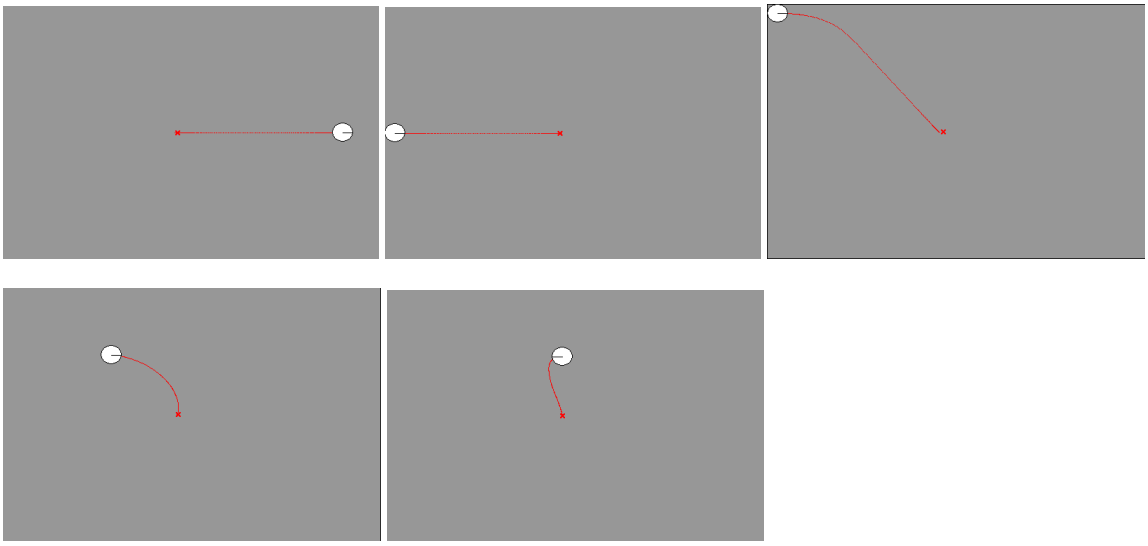

Figura 21: Algumas trajetórias de treinamento da redes. Os robô se encontram nas posições iniciais. As linhas vermelhas indicam as trajetórias percorridas. A cruzes vermelhas indicam as posições de destino.

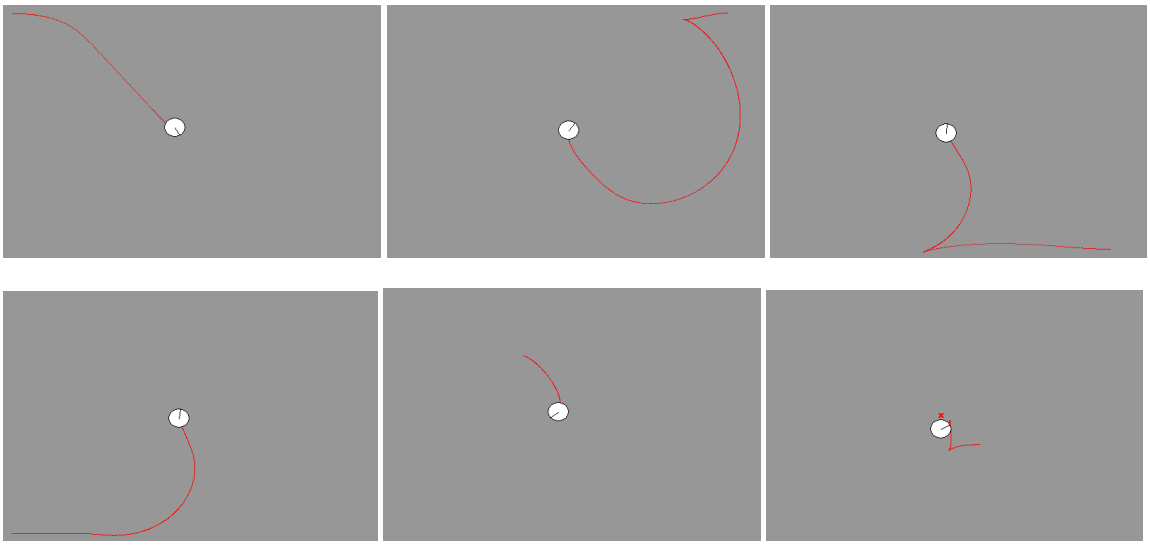

Figura 22: Algumas trajetórias efetuadas pela rede. Os robô se encontram nas posições finais.

De um modo geral, as redes geraram resultados bons, com dificuldades de realizar a tarefa para algumas áreas iniciais específicas. A rede comentada apresentou excelentes resultados, conseguindo executar a tarefa para quase toda a área coberta pela distância máxima escolhida. É interessante notar, que apesar da rede ter conseguido aprender a tarefa, ela não segue trajetórias sempre parecidas com as do conjunto de treinamento. Muitas vezes a rede encontra soluções esquisitas, não esperadas, mas consegue atingir a posição de destino. Para a grande maioria dos casos em que o robô não consegue atingir a posição de destino, ele ou passa a contorná-la, ou para próximo dela.

O motivo de a rede modelar soluções estranhas e que aparentemente não combinam com os padrões de treinamento é que diversos padrões podem apresentar dados conflitantes. Há muito ruído presente nos padrões pois é quase impossível gravar duas corridas idênticas. Cada corrida gravada possui características diferentes. Portanto, enquanto um padrão leva a rede a aprender determinado comportamento, este mesmo comportamento pode ser eliminado ao se apresentar para a rede outro padrão com características conflitantes com o primeiro.

Propõe-se para estudos futuros:

- Uma seleção mais fina dos padrões de treinamento;
- Treinar com um maior número de históricos, selecionados de modo a cobrir uniformemente o universo das variáveis de entrada;
- Tentar usar alguma variação da rede Back Propagation, pois o treinamento pela Back Propagation é muito lento;
- Usar redes RBF. Este tipo de rede provavelmente irá gerar resultados muitíssimo melhores que os obtidos com Back Propagation.

Apesar das redes não terem modelado o problema com exatidão, os resultados encontrados demonstram que o método aplicado possui validade, podendo vir a obter excelentes resultados se abordado de maneira um pouco diferente e tendo aplicações em diversos problemas similares ao proposto.

4.3 – Uma breve comparação entre o Controlador Fuzzy e o Controlador Neural

O Controlador Fuzzy obtido produz soluções mais consistentes dada qualquer posição inicial, a Rede Neural pode não prover soluções para alguns casos ou apresentar trajetórias não usuais.

Espera-se porém que a Rede Neural seja capaz de prover soluções tão boas, e até melhores, quanto as obtidas se utilizando o Controlador Fuzzy.

A Rede desenvolvida apresentou erros de distância finais menores que o Controlador Fuzzy, isto quando a Rede conseguia resolver o problema. Os erros relativos a Rede Neural são sempre próximos de zero.

Para uma comparação mais coerente, tanto a Rede Neural quanto o Sistema Fuzzy devem ser mais trabalhados, pois ambos podem apresentar melhores resultados.

O atrativo das Redes Neurais para a solução de problemas deste tipo está em não ser preciso o prévio conhecimento do problema. As regras do Controlador Fuzzy são arbitradas, já a Rede Neural aprende pela *observação* da ação humana.

Para problemas semelhantes, porém, mais complexos, Redes Neurais devem apresentar resultados melhores que Controladores Fuzzy.

5 – Conclusões

Neste trabalho, o controle de posição de um robô do tipo *differential drive* é feito através de duas técnicas distintas: Lógica Fuzzy e Redes Neurais.

Foram obtidos bons resultados com as duas técnicas, porém podem ser melhorados.

O melhor Controlador Fuzzy obtido demonstrou levar o robô para a posição desejada com erro pequeno para todas as posições iniciais testadas. Porém, tem a desvantagem de que o robô só se move para frente.

A melhor Rede Neural obtida demonstrou conseguir levar o robô para a posição desejada com distância de erro quase nula. Porém, a Rede não chegou à posição desejada para todas as posições iniciais testadas. Além disto, a rede apresentou algumas soluções estranhas e não usuais.

Apesar das redes não terem modelado o problema com exatidão, os resultados encontrados demonstram que o método aplicado possui validade, podendo vir a resultar em excelentes resultados se abordado com ressalvas.

As duas técnicas foram aplicadas em um cenário ideal com aplicação de velocidades sobre as rodas de modo direto. Propõe-se em trabalhos futuros a simulação do controle de velocidade.

Também ficam propostas para futuros trabalhos diversas modificações nas técnicas empregadas, e, o emprego destas técnicas em problemas semelhantes envolvendo complicações, tais como a simulação com obstáculos.

6 – Bibliografia

[1] Thomas Bräun (2003). *Embedded Robotics*, Springer-Verlag Berlin Heidelberg.

[2] Gregory Dudek, Michael Jenkin (2000). *Computational Principles of Mobiler Robotics*, Cambridge University Press.

[3] Vivek A. Sujan, Marco A. Meggiolaro. *Information Based Indoor Environment Robotic Exploration and Modeling Using 2-D Images and Graphs.*

[4] *Notas de Aula em Redes Neurais*, (www.ica.ele.puc-rio.br).

[5] *Notas de Aula em Lógica Fuzzy*, (www.ica.ele.puc-rio.br).

[6] *Khepera: The miniature móbile robot*, Khepera Support Team.

[7] Mendel, J.M., (1995). "Fuzzy logic systems for engineering: a tutorial". *Proc. IEEE*, Vol. 83(3): 345-377.

[8] Ricardo Tanscheit. *Sistemas Fuzzy*.

[9] Asim Shankar (2003). *Annie - Artificial Neural Network library*, (annie.sourceforge.net).

[10] Asim Shankar (2003). *Annie Reference Manual 0.51*, (annie.sourceforge.net).

[11] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[12] Marco Aurélio Cavalcanti Pacheco. *Algoritmos Genéticos: Princípios e Avaliações*.

[13] Pomerlau, D.A (1993). *Neural Networks for intelligent vehicles*. In *Proc. Intelligent Vehicles*, pp.19-24, Tokyo.