UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2569

Utilização de Modelos de Redes Neurais para Controle de Navegação de Robôs Móveis por um Chão de Fábrica

Debora Maria Rossi de Medeiros
Roseli Aparecida Francelin Romero

Nº 252

RELATÓRIOS TÉCNICOS

São Carlos – SP
Jan./2005

Resumo

A navegação autônoma de robôs móveis tem sido estudada por vários pesquisadores, e técnicas de Inteligência Artificial têm sido amplamente aplicadas a este problema. A navegação de robôs móveis pode ser utilizada em várias aplicações, como por exemplo, automação em ambientes de Chão de Fábrica onde os robôs podem exercer a função de transportar peças. Neste relatório será apresentado o desenvolvimento de um sistema que possibilita que o robô navegue autonomamente, ao longo de uma pista desenhada no chão, baseando-se em imagens, planejando sua trajetória para alcançar pontos pré-determinados. Redes Neurais Artificiais (RNAs), alimentadas por imagens, são utilizadas para decidir pelo movimento do robô em tempo real. Para o planejamento de trajetória é utilizada uma representação de mapa topológico do ambiente e para auxiliar na localização são utilizados marcos artificiais.

ÍNDICE

1. INTRODUÇÃO

A navegação autônoma de robôs móveis consiste na utilização de técnicas que possibilitem robôs móveis a navegar independentemente por caminhos como corredores, estradas, etc. Essas técnicas permitem que robôs realizem tarefas como, por exemplo, transporte de materiais.

A navegação pode ser baseada em informações de diversos tipos de sensores como, por exemplo, sonares, *lasers* e câmeras. A utilização de câmeras como fonte de informação possibilita que caminhos sejam desenhados no chão para que o robô navegue entre locais desejados. Por exemplo, em um ambiente de Chão de Fábrica[1] pode-se traçar caminhos entre máquinas para que o robô transporte peças entre elas.

Porém, informações provindas de câmeras normalmente estão sujeitas a presença de ruídos. Para tratar este problema, utilizar Redes Neurais Artificiais (RNAs) na classificação das informações de entrada tem sido uma ótima opção. Isso se deve à alta capacidade de generalização das RNAs, que é muito importante na análise de dados sujeitos a ruídos e variabilidade (Pomerleau, 1995).

Nesse texto é descrito o desenvolvimento de um sistema para controle de um robô móvel autônomo. A finalidade desse sistema é possibilitar que o robô navegue autonomamente ao longo de uma pista desenhada no chão baseando-se em imagens e planejando sua trajetória para alcançar pontos pré-determinados. Uma aplicação possível para este sistema é o transporte de peças entre máquinas em um ambiente de Chão-de-Fábrica. Redes Neurais Artificiais (RNAs) alimentadas por imagens são utilizadas para decidir pelo movimento do robô em tempo real. Para o planejamento de trajetória, é utilizada uma representação de mapa topológico do ambiente e para auxiliar na localização, são utilizados marcos artificiais.

Esse projeto está relacionado com um projeto maior, chamado ARMOSH[2], que contou com o apoio da FAPESP. O projeto ARMOSH tem como um de seus objetivos a implementação de diversos algoritmos para robôs móveis tanto em nível de software quanto de hardware.

Em (Lorena e Romero 2002) foi realizado um estudo sobre modelos de RNAs aplicados a navegação de robôs móveis por uma pista delimitada por duas faixas brancas. Um conjunto de imagens da pista foi coletado e, com ele, foram treinadas diversas topologias de

---

[1] O Chão-de-Fábrica consiste em um ambiente que contém um conjunto de máquinas, cada uma destinada a exercer uma ou mais funções em uma linha de produção.

[2] Aprendizado de Robôs Móveis via Software e Hardware.

RNAs do tipo *Multilayer Perceptron* (MLP) para se eleger a topologia mais adequada à situação. A finalidade da RNA é determinar a direção a ser seguida pelo robô a fim mantê-lo dentro dos limites da pista. A representação das saídas dessas redes é a 1-de-n onde, de n saídas, apenas uma é ativada (Pomerleau, 1992). Porém nesse estudo, não foi implementado um sistema para testar e validar a aplicação dessas RNAs em um robô em tempo real.

No presente trabalho foi desenvolvido um módulo que interliga o robô e a câmera, com a RNA (inicialmente foi utilizada a RNA resultante do trabalho de Lorena e Romero (2002)). Esse módulo captura as imagens do ambiente, aplica-lhes um pré-processamento e as submete a uma RNA que emite uma saída correspondente à direção que o robô deve seguir. Baseando-se na resposta da RNA, um comando é enviado para o robô (Medeiros e Romero, 2002).

Com os testes realizados, identificaram-se várias deficiências e possíveis aprimoramentos e, então, algumas melhorias e novas funcionalidades foram implementadas e adicionadas ao sistema. Entre eles está a adoção da representação gaussiana para a saída da RNA, (Pomerleau, 1992) (Waldherr, Romero e Thrun, 2000) que permite uma maior flexibilidade de movimentos ao robô e, conseqüentemente, maior suavidade na navegação. Outro exemplo é a adição de novas imagens ao conjunto de treinamento das RNAs a fim de possibilitar a navegação em curvas mais fechadas.

Depois disso as atenções se voltaram para o pré-processamento das imagens. Inicialmente as imagens eram convertidas para tons de cinza para evitar que a RNA tenha que tratar de informações de cor. Porém, mesmo variações de níveis de cinza não são informações relevantes para reconhecer a orientação da pista. O que importa é a distinção entre pista e chão. Então foi implementada uma maneira de binarizar as imagens de forma a converter os *pixels* das imagens em brancos (faixa) ou pretos (chão) pela análise das componentes de cor. Desta forma, o ruído é reduzido e a dependência da cor do piso e da faixa é eliminada.

Uma outra alternativa para binarizar as imagens foi implementada. Trata-se de uma técnica para separar os *pixels* das imagens em *clusters* (agrupamentos), utilizando um modelo de redes *self-organizing* chamado Fuzzy Art. Foi utilizada a implementação deste modelo realizada por Damiance Jr. e Liang (2001).

Em seguida foi implementada a funcionalidade de planejamento de trajetória, para determinar o caminho a ser seguido pelo robô para alcançar determinados pontos da pista. Foi utilizada uma representação de mapa topológico para a pista e o algoritmo de Dijkstra (Cormen, 2001) foi implementado para o cálculo do menor caminho entre dois pontos. Para auxiliar a localização do robô, foram utilizados marcos artificiais em pontos estratégicos da pista.

Por fim, foi desenvolvido um módulo de coleta e classificação automática de imagens. Este módulo possibilita a rápida criação de um novo conjunto de treinamento, permitindo, assim, a fácil adaptação do sistema a possíveis mudanças, por exemplo, ângulo de visão da câmera, tipo de pista, etc. Porém, este módulo ainda não foi testado.

Este relatório está organizado da seguinte maneira: a Seção 2 contém uma visão geral do sistema, apresentando suas principais partes a ligação entre elas. Na Seção 3 são explicadas as abordagens testadas para o pré-processamento das imagens. Na Seção 4 são apresentadas as topologias de RNAs treinadas e testadas para serem utilizadas no sistema, também são explicadas as técnicas utilizadas no treinamento. A Seção 5 contém uma descrição de como o movimento correto do robô é definido a partir da saída da RNA. Na Seção 6 é mostrado como é feito o planejamento da trajetória que o robô deve seguir, a fim de possibilita-lo a alcançar os pontos desejados da melhor maneira. Na Seção 7 é descrita a implementação de um módulo para coleta e classificação automática de imagens para treinamento de RNAs. A Seção 8 contém uma descrição sobre os testes em tempo real, resultados obtidos e dificuldades encontradas. Por fim a Seção 9 conclui o texto com algumas considerações sobre os resultados do trabalho.

2. VISÃO GERAL DO SISTEMA

Para a tarefa de seguir a pista, as imagens são capturadas seqüencialmente por meio de uma câmera conectada ao computador. Para cada imagem, a RNA gera uma saída, que é mapeada para um movimento, gerando um comando de direção que é enviado para o robô por conexão serial. A rede tem como entrada os *pixels* das imagens. Antes de serem submetidas à RNA, as imagens passam por um pré-processamento para que elas assumam características melhores para o desempenho da rede.

A captura das imagens é feita por meio de um componente baseado em *Video for Windows* que oferece funções para captura de vídeo (Huebler, 2001).

O sistema recebe os pontos do mapa para onde o robô deve seguir. O robô começa a seguir suas metas e em cada uma delas calcula o caminho para a próxima. No caminho entre uma meta e outra, o robô segue a pista normalmente até encontrar um ponto, então o sistema verifica o que fazer para seguir para o próximo ponto.

A estrutura do sistema está ilustrada na Figura 1.

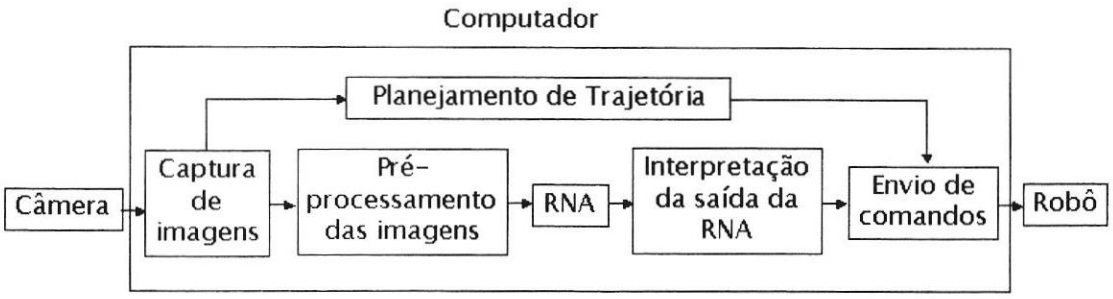


Figura 1 – Esquema do funcionamento do sistema

3. PRÉ-PROCESSAMENTO DAS IMAGENS

Submeter imagens em seu estado original a uma RNA significa exigir da RNA capacidade de lidar com grande quantidade de informação, muitas vezes desnecessária. Então, se um certo pré-processamento for, antes, aplicado às imagens, pode-se obter um ganho considerável no desempenho do sistema.

No trabalho em questão, como a pista é desenhada no chão, as imagens devem conservar apenas características que permitam distinguir a pista do chão.

Até o momento, foram treinadas RNAs com imagens de uma pista formada por duas faixas brancas, pertencentes ao banco de imagens citado anteriormente (Lorena e Romero, 2002). Um exemplo dessas imagens pode ser visto na Figura 2.

Figura 2 – Exemplo de imagem utilizada no treinamento das RNAs

Se essas imagens forem reduzidas para um certo tamanho, não perderão a informação da orientação das faixas. Isso torna o processamento da RNA mais rápido devido ao número reduzido de entradas. Assim, as imagens são reduzidas para 32x24 *pixels*.

As imagens originais contêm 320x240 *pixels* e, para serem reduzidas, é calculada a média de cada uma das três componentes (RGB – *Red* (vermelho), *Green* (verde) e *Blue* (azul)) de cada cem *pixels* (àreas de 10x10 *pixels* nas quais a imagem pode ser dividida), então os valores obtidos são atribuídos a um único *pixel*.

Para que variações de cor não tenham que ser tratadas pela RNA, dois métodos para pré-processamento das imagens foram verificados. O primeiro consiste na conversão das cores para escala de cinza e o segundo consiste em binarização, transformando *pixels* da faixa em branco e o resto em preto. Esses métodos serão descritos a seguir.

3.1 Níveis de Cinza

Para cada *pixel* da imagem, é calculada a média das três componentes (RGB), e o resultado é atribuído a cada uma das componentes. Assim, o chão pode até ser de uma cor diferente da cor utilizada nas imagens de treinamento (Lorena e Romero, 2002), contanto que seja escura.

Depois que esse processamento é aplicado, a imagem é normalizada para que seus *pixels* assumam valores entre 0 e 1. Isso é feito dividindo-se todos pelo maior deles (lembrando que nesta etapa cada *pixel* tem um valor único para R, G e B). Isso é feito para que variações de luminosidade não influenciem o funcionamento da RNA.

3.2 Binarização

Utilizando o método explicado acima, é necessário que o piso seja escuro para que as imagens fiquem parecidas com as imagens do treinamento, apenas com a faixa branca destacada. Outra desvantagem é que cada tipo de piso, mesmo que escuro, tem um padrão diferente, como tacos e pisos cerâmicos, que mesmo em níveis de cinza serão diferentes.

Então, se for possível reconhecer *pixels* pertencentes às faixas, pode-se deixar a imagem com apenas dois níveis de cor, um para as faixas e outro para o chão. Além de simplificar o processamento da RNA, isso elimina a necessidade de que o chão seja escuro e as faixas brancas. Só é, então, necessário que eles sejam de cores diferentes.

Constatou-se que, nas imagens do conjunto disponível para treinamento (Lorena e Romero, 2002), a componente azul (B) é mais intensa que a vermelha (R) nos *pixels* da faixa. Também nesses *pixels*, a componente verde (G) é mais intensa que nos outro *pixels* da imagem. Assim, todos os *pixels* que respeitassem essas regras e os que possuíssem todas as componentes próximas do nível máximo (255) foram transformados em branco, e o resto foi transformado em preto.

Uma outra possibilidade que surge com essa técnica é a de eliminar ruídos nas imagens como, por exemplo, reflexos luminosos no chão. Esses reflexos podem ser vistos na imagem da Figura 3.

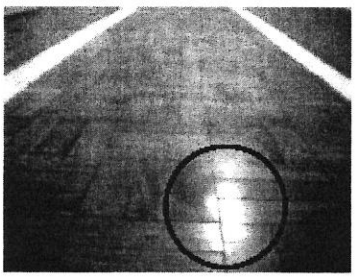


Figura 3 – Imagem com ruído (indicado com círculo).

Nas imagens de treinamento (Lorena e Romero, 2002) a cor da faixa é branca, se assemelhando muito com a cor dos reflexos, então eles não foram escondidos com a transformação explicada acima. Porém, se forem utilizadas faixas de outra cor, é possível esconder todo o resto, menos as faixas. Para isso, os ruídos foram retirados manualmente das imagens do conjunto de treinamento, visando utilizar uma cor diferente para a pista durante os testes, para alcançar o efeito citado.

Foram testadas duas maneiras para binarizar as imagens provindas da câmera. Uma consiste em o usuário analisar as imagens do ambiente e informar os limites das componentes de cor R, G e B a partir dos quais a imagem será binarizada. A outra, com a finalidade de automatizar o processo acima, consiste na "clusterização" dos *pixels* de uma imagem exemplo e utilização dessa "clusterização" para classificação dos *pixels* das imagens em tempo real.

Foi utilizada a rede Fuzzy ART pertencente à família de redes ART (Vicentini e Romero, 2003). Essas redes possuem a capacidade de auto-organização, pertencendo a um grupo de redes chamadas de *self-organizing* (Carvalho, Braga e Ludemir, 2000).

As redes *self-organizing* possuem a capacidade de aprender através de exemplos, sem que exista um supervisor externo, como no paradigma de aprendizado supervisionado que é o caso do algoritmo de treinamento *Back-propagation* utilizado para o treinar as RNAs MLP, onde a RNA é "ensinada" através de pares de entrada e resposta desejada. No caso das redes *self-organizing*, as únicas informações fornecidas são os exemplos de entrada. A rede agrupa esses exemplos em *clusters*, unindo exemplos que compartilham características comuns.

O modelo Fuzzy ART, em particular, aceita como entrada elementos assumindo valores entre 0 e 1, indicando o nível de presença da característica representada pelo elemento. No trabalho em questão a entrada é um *pixel* da imagem e as características medidas são suas componentes de cor R, G e B.

Foi utilizada a implementação deste modelo realizada por Damiance Jr. e Liang (2001) que consiste em um conjunto de métodos que permitem:

- Treinar a rede, informando um exemplo de entrada. Esse procedimento pode ser repetido com exemplos diferentes a qualquer momento sendo que a rede vai acumulando o conhecimento sem perder o treinamento anterior.
- Alterar o limiar de vigilância utilizado no treinamento. O valor deste parâmetro é proporcional a semelhança permitida para que exemplos sejam agrupados no mesmo *cluster*.
- Classificar um exemplo de entrada, ou seja, informar o *cluster* ao qual ele provavelmente pertence.

Depois que essa implementação foi adaptada ao sistema, tornou-se possível que o usuário ajuste, em tempo de execução, o limiar de vigilância visualizando o resultado da "clusterização". Para a visualização, cada *cluster* de *pixels* é colorido com uma cor diferente como na Figura 4. Depois de terminado o processo de "clusterização", o usuário deve informar a cor do *cluster* que contém os *pixels* da faixa da pista, assim, nas imagens futuramente capturadas, os *pixels* classificados pertencentes a esse *cluster* serão coloridos em branco e o resto em preto.

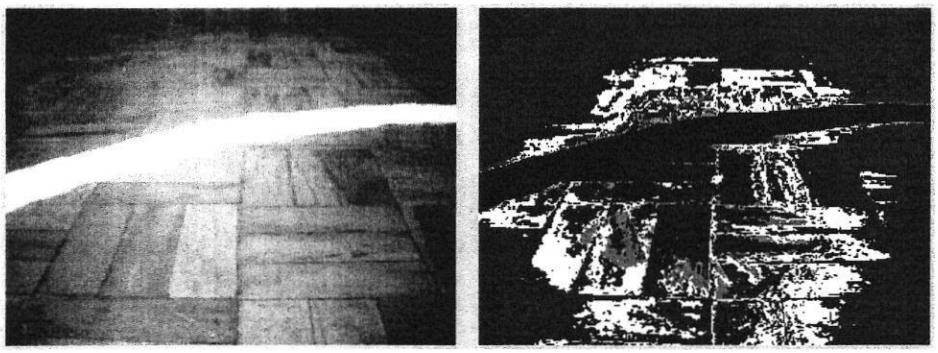


Figura 4 – Exemplo de "clusterização" de imagem da pista.

Para a "clusterização", o usuário deve utilizar uma imagem contendo um marco (auxiliadores da localização, citados anteriormente) como na Figura 5, assim, ele deve informar também a cor do *cluster* que contém os *pixels* do marco, para que eles possam ser identificados no processo de localização e planejamento de trajetória.

Utilizando a binarização como forma de pré-processamento, a conversão das cores é feita antes da redução e esta é feita da seguinte forma: se a porcentagem de *pixels* brancos dentro da área de 100 *pixels* que será reduzida para um for maior que um certo limite, aquela área será branca.

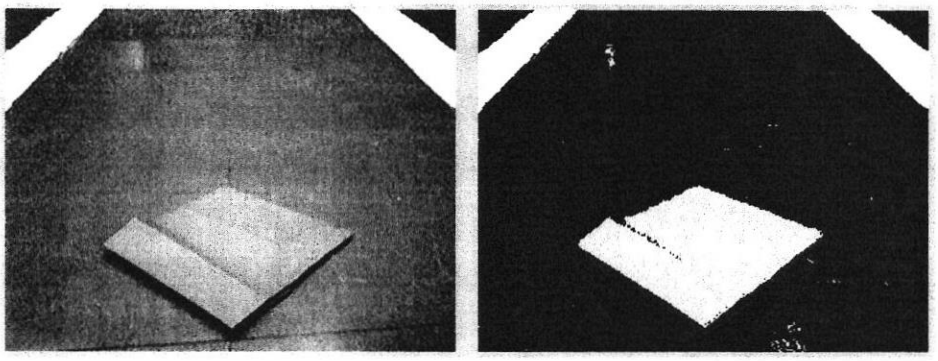


Figura 5 – Exemplo de "clusterização" de imagem da pista contendo marco.

4. REDES NEURAIS

RNAs são sistemas compostos por elementos de processamento, chamados neurônios artificiais, que são interconectados. Cada conexão (sinapse) entre neurônios está associada a um peso, para ponderar as entradas de cada neurônio. Os pesos são determinados na fase de treinamento da rede e neles está armazenado o conhecimento da RNA (Carvalho, Braga e Ludemir, 2000) (Carvalho et al, 1999). As RNAs possuem entrada e saída e, dado um certo exemplo de entrada, a RNA generaliza este exemplo de acordo com os exemplos vistos no treinamento. A saída emitida corresponde à classe a que esse exemplo pertence. O modelo de RNA utilizado neste trabalho é o *Multlayer Perceptron* (MLP), que é composto por várias camadas de neurônios, uma de entrada, uma de saída e uma ou mais camadas intermediárias. Isso possibilita o aprendizado de tarefas mais complexas como é o caso do projeto em questão. Uma ilustração deste modelo pode ser vista na Figura 6.

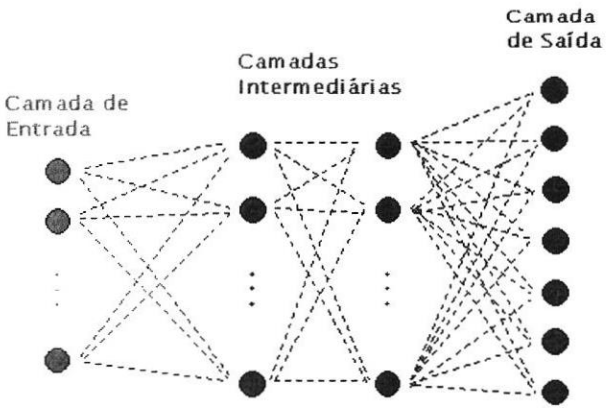


Figura 6 – Exemplo de *Multilayer Perceptron*

O algoritmo de treinamento geralmente utilizado para este modelo é o algoritmo "*Back Propagation*" (Carvalho, Braga e Ludemir, 2000). Este algoritmo pertence ao paradigma de aprendizado supervisionado, onde existe a figura de um "professor", isto é, a rede é "ensinada" através de um conjunto de pares com entrada e saída desejada.

Para o treinamento das redes foi utilizada a ferramenta SNNS (*Stuttgart Neural Network Simulator*) (University of Stuttgart, 1995). Essa ferramenta consiste em um simulador de RNAs que permite o treinamento de diversos modelos e topologias de redes por meio de diversos algoritmos de treinamento. O algoritmo de treinamento utilizado é o *Back Propagation*.

Os modelos de RNA utilizados neste trabalho foram treinados com um conjunto de imagens coletadas e classificadas por Lorena e Romero (2002). As imagens deste conjunto estão classificadas em oito diferentes classes, correspondentes a ação correta do robô diante de tais imagens. Essas classes são: curva brusca para esquerda, curva média para esquerda, curva leve para esquerda, ir em frente, curva leve para direita, curva média para direita, curva brusca para direita e parar. A cada intensidade de curva está associado um raio de curvatura. A classe parar contém as imagens para as quais é difícil determinar um movimento apropriado, por exemplo, se o robô está de frente para uma das faixas.

Algumas topologias de RNA MLP foram verificadas por Lorena e Romero (2002). A representação de saída utilizada nesses modelos é a 1-de-n. Nesta representação, cada elemento de saída corresponde a uma classe e apenas um deles é ativado, ou seja, o elemento correspondente à classe do exemplo de entrada deve possuir valor máximo e os demais, valor mínimo (Pomerleau, 1992) (Mitchell, 1997). A topologia que melhor se adequou a esse caso tem uma única camada intermediária com 200 elementos.

Porém, essa representação permite poucas possibilidades de movimento para o robô. Um desses modelos foi utilizado na primeira versão do sistema sendo descrito. Depois dos testes em tempo real, constatamos que uma pequena variação na entrada da rede pode provocar uma mudança na saída que comprometerá o restante da trajetória (Medeiros, e Romero, 2002).

Então partimos para outra representação, a gaussiana (Pomerleau, 1992) (Waldherr, Romero e Thrun, 2000). Nesta representação os valores das saídas apresentam-se na forma de uma função gaussiana (Figura 7), com seu pico centrado na saída correspondente a classe do exemplo de entrada.

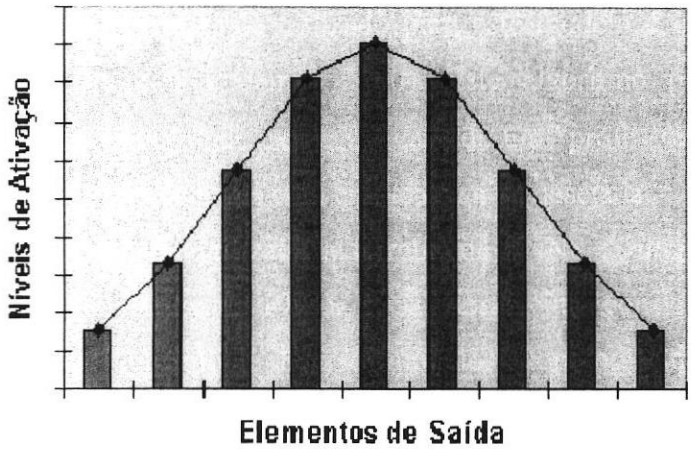


Figura 7 – Ilustração das saídas de uma RNA com representação gaussiana

Houve a necessidade do treinamento de novas RNAs. Foram utilizados os exemplos de treinamento de todas as classes citadas anteriormente, exceto da classe parar, pois, na representação gaussiana, elementos de saída consecutivos representam ações parecidas.

A saída desejada para cada exemplo de treinamento foi calculada utilizando a Equação (1). Nesta equação, xi é o valor da *i-ésima* saída, di é a distância entre a *i-ésima* saída e a saída correspondente à classe do exemplo de entrada (onde se situa o centro da gaussiana). O valor 5 é um parâmetro determinado empiricamente que controla a abertura da gaussiana.

$$x_i = e^{\frac{-d_i^2}{5}} \qquad (1)$$

No momento em que a RNA com saída gaussiana é utilizada, seus elementos de saída formam algo parecido com uma função gaussiana. Para determinar seu centro, que representa o movimento do robô, deve ser calculada a função gaussiana que melhor se ajusta às saídas (Medeiros e Romero, 2002) (Waldherr, Romero e Thrun, 2000).

O conjunto de exemplos de treinamento é dividido aleatoriamente em três subconjuntos: treinamento, validação e teste. O conjunto de treinamento contém por volta de 50% dos exemplos e é utilizado para ajustar os pesos da rede. O conjunto de validação contém por volta de 25% dos exemplos e é apresentado durante o ajuste de pesos para avaliar o desempenho da RNA em treinamento diante de novos exemplos, visando selecionar o conjunto de pesos com melhor generalização. O conjunto de teste contém os 25% restantes dos exemplos e é apresentado no final para avaliar a capacidade de generalização da rede. Trata-se de uma técnica chamada de *Cross-Validation* (Haykin, 1998), cujo objetivo é testar a performance de generalização com um conjunto diferente do conjunto de validação, para evitar o *overfitting*, isto é, evitar que a rede "vicie" nos conjuntos de treinamento e validação.

Foi implementado um módulo que gera esses três subconjuntos aleatoriamente três vezes, resultando em três particionamentos diferentes. Assim, três redes de cada topologia são geradas, e a performance da topologia é medida pela média das performances de cada rede (Prechelt, 1994). A ferramenta de simulação SNNS anota os erros MSE de cada etapa (treinamento, validação e teste). Este erro é igual a soma dos erros quadráticos (SSE) dividida pelo número de exemplos apresentados à rede. A topologia adequada é escolhida através da média dos erros MSE de generalização obtidos nas três vezes que essa topologia é treinada. O desvio padrão entre esses três valores também é analisado. Esta técnica se chama *Random Subsampling* (Lorena e Romero, 2002; Bao, 2001) e visa estimar a taxa verdadeira de erro.

Assim, as primeiras RNAs com representação gaussiana foram treinadas para imagens reduzidas e em tons de cinza. Foram testadas as topologias listadas na Tabela 1.

Tabela 1 – Topologias verificadas.

Topologia	Nº de camadas intermediárias	Nº de elementos na 1ª camada intermediária	Nº de elementos na 2ª camada intermediária
1	1	5	-
2	1	50	-
3	1	100	-
4	1	200	-
5	2	50	20
6	2	100	30

A topologia que melhor se adequou foi a de número 6 na Tabela 1. Abaixo, na Tabela 2 são apresentadas as taxas de erro MSE observadas durante seu treinamento.

Tabela 2 – Taxas de erro MSE do treinamento da topologia escolhida (P1, P2 e P3 correspondem aos diferentes particionamentos)

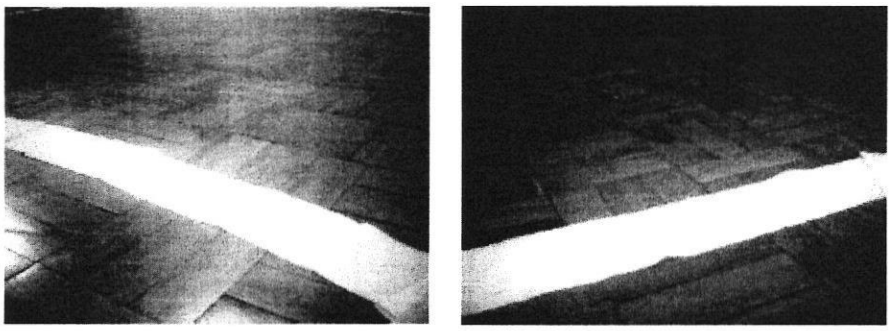
Topologia	Etapa	P 1	P 2	P 3	Média	Desvio Padrão
6	Treinamento	0,009515	0,009889	0,009527	0,009644	0,000213
	Validação	0,141449	0,130582	0,122457	0,131496	0,009529
	Teste	0,159289	0,174140	0,137129	0,156853	0,018625

É importante ressaltar que esta topologia foi escolhida com uma média de erro de generalização igual a 0,131. Esse erro é relativamente baixo se comparado ao erro da melhor topologia, com saída 1-de-n, que é igual a 0,329 (Lorena e Romero, 2002).

O próximo passo foi dado em direção a tratar o problema dos ruídos na imagem, exemplificados na Figura 3. Então, o pré-processamento explicado na Seção 3.2 foi aplicado as imagens de treinamento. Os ruídos foram retirados manualmente das imagens, pois, se uma cor de faixa diferente de branco fosse utilizada, ela poderia ser diferenciada dos reflexos e outros ruídos de cor branca, e após o pré-processamento eles seriam eliminados.

Nas entradas dos pares de treinamento os *pixels* brancos são representados por 1 e os pretos por –1.

Também foram adicionadas duas classes de exemplos de treinamento para solucionar dificuldade da RNA em processar imagens de curvas muito fechadas. Essas classes representam os movimentos: curva mais brusca para a esquerda e para a direita. Foram utilizadas as imagens originalmente coletadas para representar a classe parar (Lorena e Romero, 2002; Medeiros e Romero, 2002). Dois exemplos dessas imagens podem ser vistos na Figura 8.



(a) **(b)**

Figura 8 – Exemplos de figuras adicionadas ao conjunto de treinamento para formar as classes: esquerda mais brusca (a) e direita mais brusca (b).

A representação de saída utilizada para este novo conjunto de pares de treinamento também foi à gaussiana. Também foi utilizada a Equação (1), porém o parâmetro que controla a abertura da gaussiana agora é igual a 9, devido ao maior número de saídas na RNA.

Assim, com esse novo conjunto, foram treinadas as mesmas topologias listadas na Tabela 1. Os erros resultantes do treinamento dessas redes estão organizados na Tabela 3.

Podemos perceber que a melhor topologia é a de número 5, cuja média de erro de generalização é igual a 0,089. É importante salientar que este resultado é melhor que o obtido do treinamento com imagens em níveis de cinza, encontrado na Tabela 2. Também é importante observar que essa topologia é mais simples que a antiga, contendo menos elementos por camada. Topologias mais simples são mais interessantes, quando aplicadas a sistemas de tempo real.

Tabela 3 – Erros MSE resultantes do treinamento das RNAs com imagens em preto e branco (P1, P2 e P3 correspondem aos diferentes particionamentos)

Topologia	Etapa	P 1	P 2	P 3	Média	Desvio Padrão
1	Treinamento	0,032	0,046	0,044	0,041	0,00750
	Validação	0,122	0,152	0,126	0,133	0,01633
	Teste	0,119	0,148	0,135	0,134	0,01476
2	Treinamento	0,004	0,004	0,004	0,004	0,00002
	Validação	0,094	0,110	0,126	0,110	0,01615
	Teste	0,103	0,110	0,104	0,106	0,00382
3	Treinamento	0,004	0,004	0,004	0,004	0,00036
	Validação	0,148	0,163	0,119	0,143	0,02226
	Teste	0,134	0,131	0,114	0,126	0,01095
4	Treinamento	0,004	0,004	0,004	0,004	0,00018
	Validação	0,161	0,163	0,142	0,155	0,01160
	Teste	0,162	0,159	0,162	0,161	0,00199
5	Treinamento	0,004	0,004	0,004	0,004	0,00016
	Validação	0,095	0,092	0,080	0,089	0,00784
	Teste	0,098	0,089	0,070	0,086	0,01463
6	Treinamento	0,004	0,004	0,004	0,004	0,00008
	Validação	0,085	0,109	0,105	0,100	0,01321
	Teste	0,080	0,098	0,101	0,093	0,01121

5. INTERPRETAÇÃO DA SAÍDA DA RNA

O controle do robô é feito variando-se sua velocidade angular, enquanto a velocidade escalar permanece constante. Então, a saída da RNA deve ser mapeada para a velocidade angular correspondente.

Como dito anteriormente, a cada classe foi associado um valor de raio de curvatura (Lorena e Romero, 2002). Então, sabendo o valor do raio é possível determinar a velocidade angular através da Equação (2), onde v é a velocidade escalar atual, ω é a velocidade angular e R é o raio de curvatura desejado.

$$v = \omega \times R \tag{2}$$

Utilizando a RNA com saída 1-de-n, a determinação do raio é relativamente simples, pois sua saída é normalmente apresentada com um de seus elementos possuindo um alto valor e os restantes com valores baixos, então, quando ela está em operação, seus elementos de saída são analisados para se encontrar o de maior valor. Se o maior valor for maior que um certo limite determinado empiricamente e for suficientemente maior do que os valores dos outros elementos, a saída é considerada confiável. Um exemplo de uma situação em que a saída da RNA é confiável está ilustrado na Figura 9, onde o movimento correto é uma curva suave para a direita (neste caso, ainda eram utilizados oito elementos de saída).

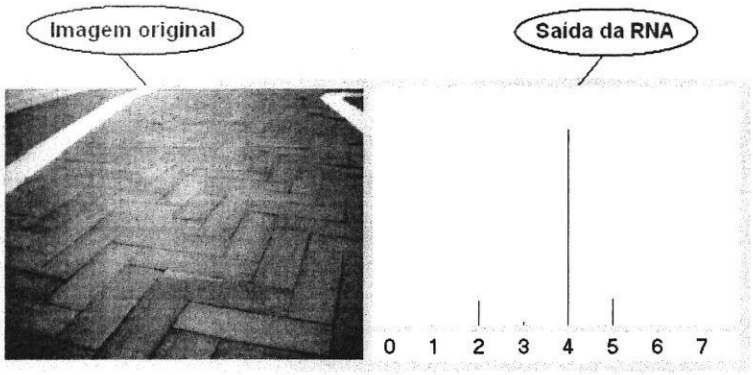


Figura 9 – Ilustração de uma saída satisfatória

Quando as duas condições citadas, para que uma saída seja confiável, não são satisfeitas, é concluído que a rede está confusa. Neste caso, o robô rotaciona para buscar um melhor ângulo de visão.

O raio associado à classe correspondente ao elemento de maior valor é utilizado na Equação (2). Quando esse elemento corresponde à classe parar, o procedimento é o mesmo de quando a rede está confusa.

Quando RNAs com representação gaussiana são utilizadas, existem diversas possibilidades de direção para o robô que variam de acordo com a posição do centro da gaussiana, que é utilizado para calcular o raio de curvatura.

Primeiro é necessário determinar o centro da função gaussiana que melhor se ajusta às saídas emitidas pela RNA.

Isso é feito testando diversas posições para o centro da gaussiana. Para cada posição, é calculada a soma dos quadrados das diferenças entre as saídas verdadeiras da rede e valor da função gaussiana correta naquele ponto. Um resultado deste procedimento é mostrado na Figura 10.

Uma vez determinado o centro da gaussiana, este deve ser mapeado para o raio de curvatura correspondente. Anteriormente isto era feito utilizando a Equação (3) (quando esta

equação era utilizada, o elemento central era 3 pois existiam sete classes), onde R é o raio e c é o centro da gaussiana. O gráfico desta equação é mostrado na Figura 11. Essa é uma maneira intuitiva de transformar a posição do centro da gaussiana em raio, pois o raio tende ao infinito quando se aproxima do centro, que representa o movimento "ir em frente", e tende a zero quando a curva vai se tornando mais fechada.

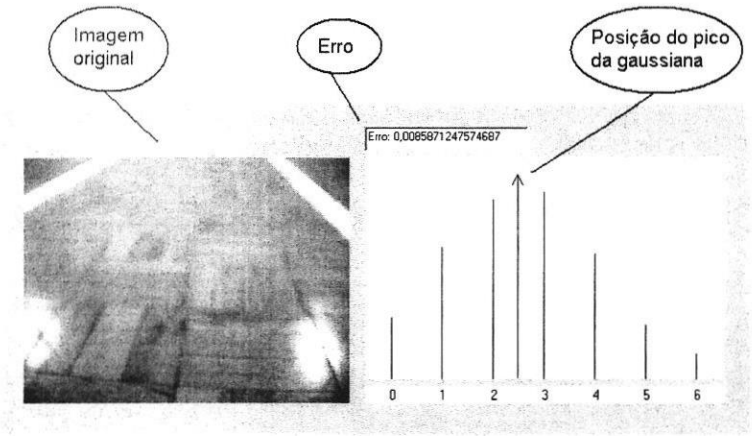


Figura 10 - Ilustração do método utilizado para estimar o centro da função gaussiana ajustada à saída da RNA (quando este resultado foi produzido, ainda eram utilizados sete elementos de saída).

$$R = \frac{1}{|c-3|}, \text{ se } c \neq 3 \tag{3}$$

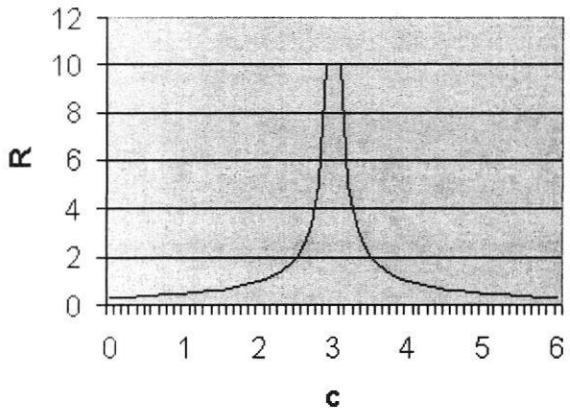


Figura 11 - Gráfico da Equação (3)

Porém, com esta equação, os movimentos do robô estavam um pouco bruscos. Então, a alternativa foi partir para um mapeamento linear entre o centro da gaussiana e o raio de curvatura, utilizando a Equação (4), cujo gráfico é apresentado na Figura 12.

$$R = ((0{,}4 * c) + 0{,}3), \text{ se } c < 4$$
$$R = ((-0{,}4 * c) + 3{,}5), \text{ se } c > 4 \qquad (4)$$

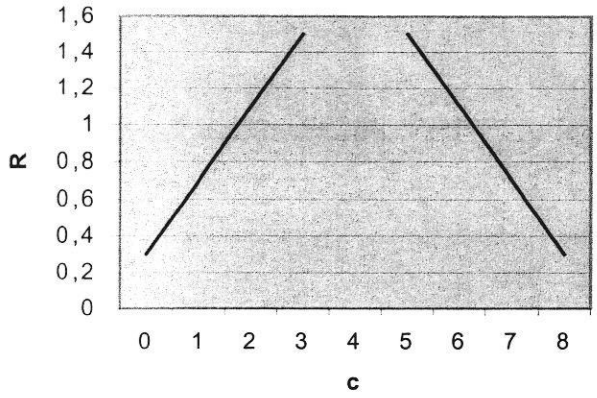


Figura 12 - Gráfico da Equação (4)

6. PLANEJAMENTO DE TRAJETÓRIA

A navegação de robôs móveis por caminhos desenhados no chão permite que o robô navegue entre os locais desejados. Por exemplo, em um ambiente de Chão-de-Fábrica pode-se traçar caminhos entre máquinas para que o robô transporte peças entre elas. Um exemplo de tal ambiente encontra-se na Figura 13. Porém, de acordo com a sofisticação do ambiente, pode haver a necessidade de o caminho ser complexo, contendo bifurcações e vários trajetos possíveis para alcançar um determinado ponto. Um caminho de tal tipo pode ser representado como um grafo, cujos vértices seriam pontos estratégicos, como as máquinas onde o robô deve parar e as bifurcações.

Dessa forma, foi implementado o planejamento de trajetória utilizando a teoria sobre grafos.

O mapa é representado como um conjunto de listas, uma para cada vértice, como está ilustrado na Figura 14. Nesta representação, cada lista contém os vértices adjacentes ao vértice a que se refere e cada elemento da lista contém o vértice e a distância até ele.

Quando uma lista contém mais do que um elemento, significa que o vértice de referência corresponde a uma ramificação. Este sistema permite até três ramos em uma ramificação e, em cada lista, os vértices estão ordenados de acordo com qual ramo seguir para chegar até eles, da esquerda para direita. O elemento da lista contendo valor –1 e –1 existe para o caso de naquele vértice existir ramo para frente e para direita.

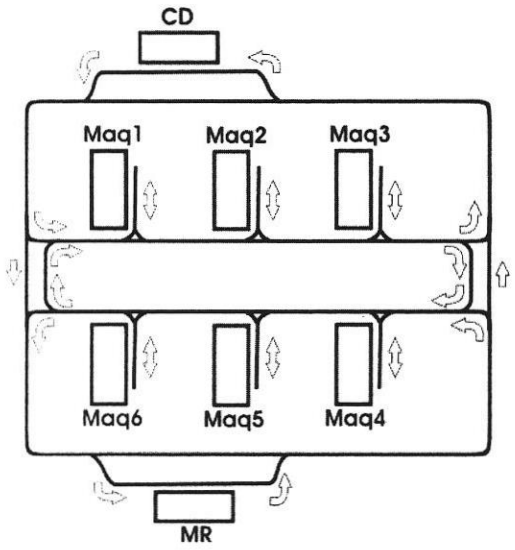


Figura 13 (Morandin et al, 2000)

Maq# = Máquinas que depositarão ou receberão as peças

CD = Local de carga e descarga

MR = Local de manutenção do robô

As setas indicam as direções que o robô pode seguir

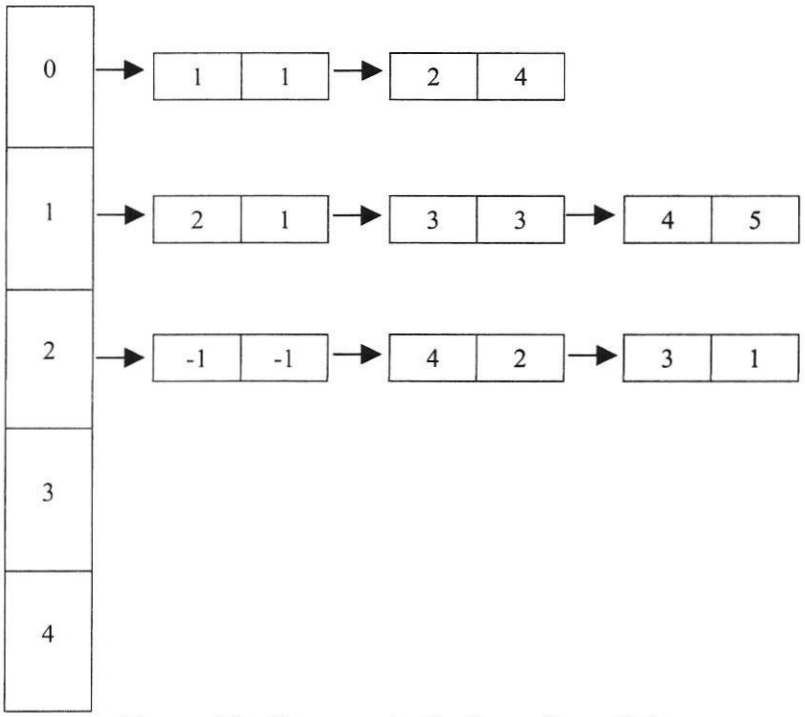


Figura 14 – Representação de grafo em listas

O algoritmo de Dijkstra (Cormen, 2001) foi implementado para buscar, em uma estrutura como a da Figura 14, o menor caminho entre dois vértices.

O sistema recebe os pontos do mapa para onde o robô deve seguir e começa a alcançar as metas na ordem em que forem recebidas. A implementação do algoritmo de Dijkstra retorna uma lista de vértices a serem visitados até a meta. O robô segue a pista até chegar a um vértice. O sistema reconhece que está em um vértice com o auxílio de marcos artificiais posicionados no ambiente, como está ilustrado na Figura 15. Esse tipo de marco é detectado pela presença, na imagem, de *pixels*, pertencentes ao *cluster* de *pixels* do marco, resultante do processo de "clusterização".

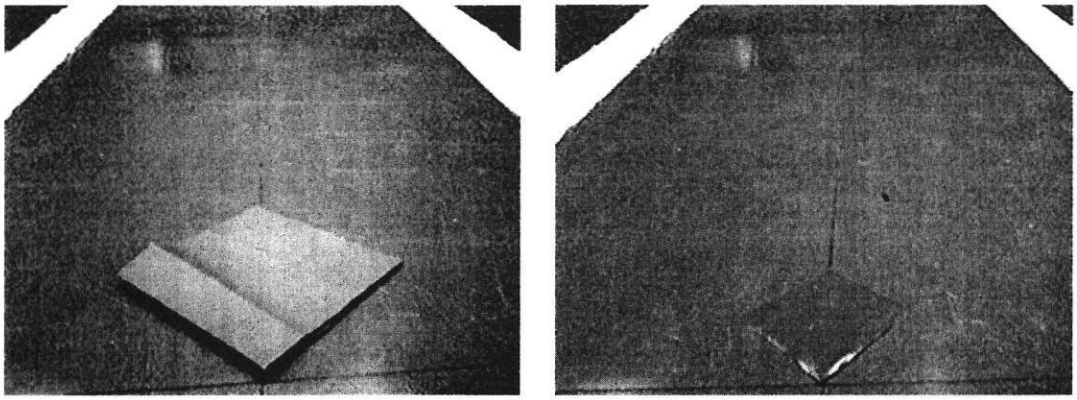


Figura 15 – Exemplo de marcos artificiais para auxiliar a localização do robô

Quando um vértice é alcançado, o sistema verifica o que fazer para seguir para o próximo vértice da lista resultante do procedimento de busca no grafo. Pode ser simplesmente seguir em frente, no caso de o vértice atual conter apenas um adjacente, ou escolher uma de algumas direções (esquerda ou direita), no caso de ramificações. No segundo caso, busca-se a posição do próximo vértice na lista de adjacências do vértice atual, o que determinará qual das pistas, originárias da ramificação, o robô deverá seguir. Sendo esquerda, o robô realizará uma curva de 90 graus para a esquerda, sendo direita, procedimento análogo para direita.

Assim que o vértice destino é alcançado, o procedimento é repetido para o próximo, e assim em diante até que todas as metas tenham sido cumpridas, então o robô para e espera por novas instruções.

7. MÓDULO DE COLETA E CLASSIFICAÇÃO AUTOMÁTICA DE IMAGENS

O conjunto de imagens classificadas para treinamento das RNAs (Lorena, 2002) foi coletado com a câmera posicionada logo em cima do robô, a uma distância de 21 cm do chão e, portando, é nesta posição que a câmera dever estar durante a execução do sistema.

Acreditava-se que este ângulo de visão estivesse prejudicando o desempenho do sistema, pois o robô "enxerga" um pedaço da pista que está bem à frente. Então ele pode estar "enxergando" tanto uma pista reta como uma curva estando em uma posição em que deve continuar para frente, ou seja, duas situações diferentes para respostar iguais.

O software de comunicação entre o robô e o computador, Saphira, permite que o robô seja controlado pelas teclas direcionais (setinhas) do teclado. Este software também possui uma estrutura, sfRobot, que armazena informações de movimento e posição do robô, entre outras.

Isso possibilitou o desenvolvimento de um módulo que, enquanto o robô é controlado pelo usuário, captura uma seqüência de imagens e no momento em que cada imagem é coletada, extrai informações do movimento que o robô estava executando.

Essas informações são extraídas da seguinte forma: no momento em que a imagem é coletada, a orientação (em graus) do robô é armazenada e após um pequeno espaço de tempo é verificado quantos graus o robô variou da posição inicial. Assim pode-se calcular sua velocidade angular (em graus/segundo) para aquela imagem.

Assim é possível criar, facilmente, novos conjuntos de treinamento, coletados com diferentes ângulos de visão da câmera ou modificando o tipo de pista, por exemplo, com apenas uma faixa no centro. Porém este módulo ainda não foi testado.

8. TESTES E DIFICULDADES ENCONTRADAS

Os testes em tempo real são feitos com o robô móvel Pioneer I (Figura 16) da companhia *ActivMedia*, presente no LABIC[3]. O robô é conectado a um *notebook*, onde o sistema é executado. O *notebook,* por sua vez, é conectado a uma *webcam,* de onde as imagens são capturadas. O sistema montado é mostrado na Figura 16.

Os comandos são enviados para o robô por meio da ferramenta Saphira (Konolige, 1997) (Medeiros e Romero, 2002), que constitui a interface de comunicação com o computador, desenvolvida para o robô Pioneer. Essa ferramenta possui uma biblioteca (API) que contém diversas funções como comandos de locomoção, leitura de sensores, etc.

[3] Laboratório de Inteligência Computacional do Instituto de Ciências Matemáticas e de Computação (ICMC) da USP.

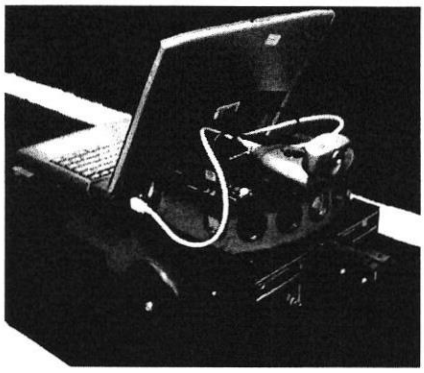


Figura 16 – Sistema montado para testes em tempo real.

Ao longo do desenvolvimento do projeto foram feitos vários testes em tempo real e os resultados têm sido cada vez mais satisfatórios.

No início, quando era utilizada a RNA com saída 1-de-n, o robô saía muitas vezes da pista, apesar de apresentar a tendência de permanecer dentro dela.

A partir da utilização da representação gaussiana, a performance do sistema apresentou uma melhora muito significativa. Foram gravados dois vídeos que mostram o sistema em funcionamento, eles estão disponíveis em http://grad.icmc.usp.br/~debora.

A performance do sistema aumentou ainda mais depois das últimas modificações explicadas anteriormente: utilização das imagens em branco e preto, adição das duas classes para curvas mais pesadas e mapeamento linear entre o centro da gaussiana e o raio de curvatura, pois, antes, o robô apresentava dificuldades, como centralizar sua posição em algumas curvas, passando por cima de uma das faixas. Então, outros dois vídeos foram gravados e disponibilizados no mesmo *site*.

Quando eram utilizadas imagens em tons de cinza, o tempo de processamento de uma imagem, desde sua captura até o envio de comandos para o robô (sem o planejamento de trajetória), é, em média, 300 milisegundos e, somente o tempo gasto pelo processamento da Rede Neural, é, em média, 20 milisegundos. Utilizando o processo de binarização através da comparação com limiares das componentes, o tempo total de processamento é por volta de 500 milisegundos e somente a Rede Neural consome em média 5 milisegundos de tempo de processamento. Utilizando a rede Fuzzy ART para binarizar as imagens, o tempo médio total de processamento aumenta para 750 milisegundos.

Não foi possível testar o sistema, em tempo real, após a implementação do planejamento de trajetória e da "clusterização", devido à falta de equipamento, pois o *notebook* utilizado nos testes esteve em conserto durante a implementação desta parte do projeto.

9. CONCLUSÕES

Observando o comportamento do robô nos vídeos, é interessante notar que ele faz um leve movimento de vai e volta. Isto pode significar que rede tenha aprendido a manter o robô nos limites da pista, e não a orientação das curvas.

A representação gaussiana apresenta-se mais apropriada para esse tipo de aplicação, comparada à saída 1-se-n, devido ao fato de permitir maior flexibilidade de movimento e, conseqüentemente, evitar que mudanças bruscas entre as direções tomadas pelo robô prejudicam o restante da trajetória.

Com a modificação no pré-processamento das imagens, o processamento da RNA ficou mais simples, porém, o restante ficou mais custoso, devido à binarização. Entretanto, o aumento na performance do sistema e a possibilidade de melhor adaptação a novas condições do ambiente compensam essa desvantagem.

Comparando os dois métodos de binarização percebe-se que, utilizando a rede Fuzzy ART, o processamento é mais custoso, porém, ela possibilita o ajuste automático do sistema a cor do chão e das faixas que compõem a pista. Outra vantagem é que a resposta emitida pela RNA, relativa a uma imagem de entrada pré-processada com essa técnica, apresenta-se mais próxima a uma gaussiana perfeita, com um erro de até 75% menor do que a gaussiana obtida utilizando a outra técnica. A performance em tempo real ainda deve ser testada.

Pretende-se, também aplicar a "clusterização" ao conjunto de imagens de treinamento e treinar novas RNAs. Verificar a performance ao clusterizar, não *pixels* isolados, e sim áreas de *pixels*, reduzindo a imagem nesta etapa. E, por fim, testar o módulo de coleta e classificação automática de imagens, que pode possibilitar uma adaptação mais rápida do sistema a novas configurações de pista.

O conhecimento adquirido durante o curso de graduação influenciou bastante no sucesso do projeto, principalmente em termos de técnicas de programação.

O desenvolvimento deste projeto, sem dúvida nenhuma, foi fundamental para a experiência e segurança que a aluna atualmente possui. A convivência com outras pessoas envolvidas com pesquisa no LABIC e os desafios encontrados durante o desenvolvimento do projeto contribuiam para o desenvolvimento de uma afinidade com a pesquisa e de uma maior capacidade de resolver problemas por parte da aluna.

REFERÊNCIAS

BAO, H. T. "Knowledge Discovery and Data Mining Techniques and Practice", Department of Pattern Recognition and Knowledge Engineering, Institute of Information Technology, Hanoi, Vietnam Japan Advanced Institute of Science and Technology.

CARVALHO, A. C. P. L. F.; BRAGA, A. P.; LUDEMIR, T. B. **Redes Neurais Artificiais:** Teoria e aplicações. Rio de Janeiro: Livros técnicos e Científicos Editora S.A., 2000. 262p.

CARVALHO, A. C. P. L. F., et al. **Sistemas Inteligentes:** Aplicações a Recursos Híbridos e Ciências Ambientais. 1. ed. Porto Alegre: Editora da Universidade – Universidade Federal do Rio Grande do Sul, 1999. 246p.

CORMEN, T. H., et al. **Introduction to Algorithms**. 2nd. ed. MIT Press, 2001.

DAMIANCE Jr., A. P. G., LIANG, Z. Reconhecimento de Imagens utilizando Redes Neurais In: 9º SIICUSP - SIMPÓSIO INTERNACIONAL DE INICIAÇÃO CIENTÍFICA DA UNIVERSIDADE DE SÃO PAULO, 2001, São Paulo.

HAYKIN, S. **Neural Networks**: A Compreensive Foundation. 2nd. ed. USA: Prentice-Hall Inc., 1998.

HUEBLER, J, "*TVideoCap Version 2.3*", Manual de Referência, January 2001.

LORENA, A. C. e R. A. F. ROMERO (2002). Utilização de Modelos de Redes Neurais Associados a Imagens para Navegação de Robôs Móveis. **SBC – Revista Eletrônica de Iniciação Científica** (Mar.), Vol. 2, No. 1.

MEDEIROS, D. M. R.; ROMERO, R. A. F. Sistema de Controle Autônomo de Robôs Móveis com Utilização de Redes Neurais Artificiais. In: Workcomp'2002 – V Workshop de Computação, 2002, São José dos Campos. Trabalho Completo, **Workcomp'2002 - Workshop de Computação**. São José dos Campos: Papercrom, 2002. p.131 – 136.

MITCHELL, T. M. **Machine Learning**. USA: McGraw-Hill, 1997. 414p.

MORANDIN Jr, O., KATO, E.R.R., POLITANO, P.R., CAMARGO, H.A., PORTO, A. J. V., INAMASU, R.Y., A Modular Modeling Approach for Automated Manufacturing Systems Based on Shared Resources and Process Planning Using Petri Nets. In: 2000 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 4, 2000. Nashville, TN, USA. 5p. P 3057-3062.

POMERLEAU, D. **Neural Network Perception for Mobile Robot Guidance.** 1992. 104f. Phd Thesis - Carnegie Mellon University.

POMERLEAU, D. Neural Network Vision For Robot Driving. Carnegie Mellon University, 1995.

PRECHELT, L. Proben 1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical Report 21/94, Universität Karlsruhe – Fakultät für Informatik, 1994.

UNIVERSITY OF STUTTGART. SNNS – Stuttgart Neural Network Simulator User Manual version 4.1, Institute for Parallel and Distributed High Performance Systems (IPVR), Technical Report N° 6/95, 1995.

VICENTINI, J. F., ROMERO, R. A. F. **Indexação e Recuperação de Informações Utilizando Redes Neurais da Família ART**. 2003. Dissertação de Mestrado – ICMC (Instituto de Ciências Matemáticas e de Computação), Universidade de São Paulo.

WALDHERR, S., ROMERO, R. A. F., THRUN, S., "Gesture-Based Interface for Human-Robot Interaction", **Journal Autonomous Robots**, Vol. 9, no. 2, pp. 151-173, 2000.