Navegação Autônoma de Robôs: Uma Implementação Utilizando o Kit Lego Mindstorms

Sidnei Alves de Araújo[1,2], **André Felipe H. Librantz**[1,3], **Oswaldo Flório Filho**[1]

[1]Departamento de Ciências Exatas – Centro Universitário Nove de Julho (UNINOVE)
São Paulo – SP – Brasil

[2]Laboratório de Processamento de Sinais – Escola Politécnica – USP
São Paulo – SP – Brasil

[3]IPEN – Instituto de Pesquisas Energéticas e Nucleares
São Paulo – SP – Brasil

{saraujo,librantz,oswaldo}@uninove.br

Abstract. *The necessity of improving the products quality and to optimize the process time regarding the manufacturing of them, researches related to the automation and robotics have been intensified. In this work we describe an autonomous navigation system using Lego Mindstorms kit. The proposed method of navigation is based on computational vision that describes the environment allowing the robot to make decisions. This robot, since obtained informations about its localization, target and the obstacles positions, must decide which path to travel in order to get the target. Implementation and applied techniques details and results, as well, are discussed in this work.*

Resumo. *Diante da necessidade de melhorar a qualidade dos produtos e otimizar o tempo dos processos envolvidos na fabricação dos mesmos, as pesquisas relativas a automação e robótica têm sido intensificadas. Neste trabalho é apresentado um sistema de navegação autônoma de robôs utilizando o Kit Lego Mindstorms. O método de navegação proposto é baseado em mecanismos de visão computacional que descrevem o ambiente para que o robô possa tomar decisões. O robô, a partir das informações recebidas sobre o seu posicionamento, a posição do alvo e dos obstáculos, deve decidir qual trajeto fazer para atingir o objetivo. Detalhes da implementação, das técnicas utilizadas bem como os resultados experimentais são discutidos nesse trabalho.*

1. Introdução

Nos últimos anos, tem havido um grande interesse em pesquisas voltadas para automação de processos por meio de sistemas robóticos com intuito de promover a melhoria da qualidade dos produtos e otimização do tempo. Estes sistemas, em sua maioria, utilizam técnicas de Inteligência Artificial que são empregadas na construção dos algoritmos propostos para solucionar os problemas. Por isso, sistemas autônomos de navegação robótica que permitem a tomada de decisões a partir de informações extraídas do ambiente, que proporcionam a cooperação de agentes ou que possuem visão computacional, têm sido largamente explorados em pesquisas nas áreas de Automação, Robótica e Inteligência Artificial e muitas aplicações têm sido propostas.

Robôs de navegação autônoma geralmente são objetos de grande admiração dada a sua "inteligência" para se deslocarem de maneira independente. Associar características

da inteligência humana à máquinas é, no mínimo, interessante. Entretanto, vale lembrar que os robôs móveis, mesmo com toda tecnologia, ainda apresentam muitas limitações quanto à sua capacidade de navegação (Cazangi & Figueiredo, 2000). A navegação autônoma de robôs requer, entre outras coisas, aprender estratégias de navegação, adaptar-se a novas situações e construir conhecimento a partir de informações obtidas do seu ambiente (Sun et al., 2002). Tais capacidades são responsáveis pela caracterização do sistema de navegação autônomo e são essenciais para se encontrar trajetórias eficientes e seguras em ambientes desconhecidos. O desenvolvimento de uma teoria relacionada ao projeto de robôs móveis autônomos traz conseqüências práticas importantes considerando-se as diversas aplicações a que podem estar empenhados (Cohen et al., 2002). O estudo e desenvolvimento de um projeto de robôs autônomos pode estar ligado a diversas aplicações práticas e têm suscitado grandes desafios dado que as dificuldades se multiplicam à medida que os ambientes de navegação tornam-se mais imprevisíveis e diversificados. Diversos mecanismos para o controle de robôs móveis têm sido utilizados tais como sonares, lasers e visão (Quiles & Romero, 2004; Pomerleau, 1995).

O objetivo deste trabalho é explorar o desenvolvimento de um sistema de navegação autônoma de robôs utilizando o Kit Lego Mindstorms, baseado em mecanismos de visão computacional. A partir do sistema de visão, informações do ambiente são transmitidas ao robô para que o mesmo possa tomar decisões. As informações recebidas pelo robô referem-se à sua localização, localização do alvo e dos obstáculos dispostos aleatoriamente no ambiente. Usando tais informações, o trajeto é calculado por um algoritmo de busca que, a partir de funções heurísticas, tenta escolher o melhor caminho, considerando as particularidades do ambiente.

2. Metodologia

Para possibilitar a navegação autônoma do robô, uma câmera de vídeo é posicionada sobre o ambiente, provendo uma visão panorâmica do mesmo. A partir das imagens capturadas pela câmera, o sistema de visão computacional "mapeia" o ambiente no qual o robô tem que se mover. Este mapeamento provê a identificação do robô e seu posicionamento, a localização dos obstáculos e a identificação do local de destino (alvo), por meio de coordenadas cartesianas. O mapa do ambiente é descrito como uma matriz M onde as dimensões de cada célula $M(i, j)$ são baseadas nas dimensões físicas do robô. Assim, a partir do mapeamento feito pelo sistema de visão, o trajeto é calculado por um algoritmo de busca usando 3 diferentes funções heurísticas. As técnicas utilizadas na construção do robô bem como as implementações do sistema de visão e do cálculo do trajeto são descritas com mais detalhes nesta seção.

Para implementação de todos os algoritmos utilizou-se a linguagem C. Nas etapas que envolvem processamento de imagens foi utilizada também uma biblioteca denominada IMG que consiste de um conjunto de rotinas para processamento de imagens escrito em C++, desenvolvida por Kim (2005).

2.1. Construção do robô utilizando o Kit Lego Mindstorms

O robô utilizado neste trabalho foi construído com o Kit Lego Mindstorms no formato retangular com dimensões de aproximadamente 10x12x15 cm (Figura 1). Visando dar maior manobrabilidade em espaços limitados o robô foi dotado de esteiras com 2 eixos. Embora o Kit Lego Mindstorms possua vários sensores e outros acessórios, o único sensor

utilizado é uma câmera de vídeo que provê as imagens para que o sistema de visão possa fornecer os dados para a tomada de decisão do robô.

Figura 1. Fotos do protótipo do robô montado. O formato e a cor do robô facilitam distinguí-lo dos demais objetos do cenário.

2.2. O Sistema de visão

Nesta seção são apresentados de forma sucinta alguns conceitos teóricos além dos procedimentos empregados no sistema de visão do robô. As etapas envolvidas no processo vão desde a aquisição das imagens até a fase de análise das mesmas onde se reconhece o robô e os demais objetos (obstáculos e o alvo).

2.2.1. Imagens digitais

Uma imagem digital pode ser vista como uma matriz cujos índices de linha e de coluna identificam um ponto (*pixel*) na imagem e o valor do elemento da matriz identifica o nível de cinza do *pixel* (Gonzalez & Woods, 2002; Pratt, 1991). Normalmente, uma imagem digital é definida como uma função bidimensional $f(i, j)$, com $i, j \in \mathrm{Z}^2$. Uma imagem em níveis de cinza G^y pode ser descrita como uma função na qual o contra-domínio representa diferentes níveis de cinza. Para denotar os níveis de cinza, geralmente utiliza-se o intervalo discreto [0,255] (Araújo & Kim, 2005). Uma imagem colorida $C^{\mathbf{y}}$, é uma imagem multibanda, onde a cor de cada *pixel* é representada de acordo com um modelo de cor (Shiba et al., 2005). Os modelos de cores mais comuns em processamento de imagens são o RGB (*Red, Green, Blue*) e HSI (*Hue, Saturation, Intensity*). Segundo Gonzalez & Woods (2002), no modelo de cor RGB, uma imagem colorida pode ser vista como um conjunto de três imagens em níveis de cinza independentes ($G^y_{R,G,B}$), cada uma delas representando uma das componentes de cor (vermelho, verde e azul). No modelo HSI, as três imagens em níveis de cinza ($G^y_{H,S,I}$) representam a matiz, a saturação e a intensidade. A matiz está relacionada à cor do pixel, a saturação diz respeito à pureza da cor e a intensidade denota a quantidade de luz.

$$G^y, G^y_{R,G,B}\ ,\ G^y_{H,S,I} : \mathbf{Z}^2 \rightarrow [0, 255] \qquad (1)$$

Neste trabalho foram empregadas as propriedades do modelo RGB para distinção do robô e do alvo dos demais objetos do ambiente.

2.2.2. Aquisição de imagens

Na primeira etapa é capturada a imagem panorâmica do cenário, constituído basicamente por um robô de cor amarela, o alvo (indicando o local de destino) na cor azul e alguns obstáculos na cor preta dispostos em um piso de tonalidade cinza. Para esta finalidade,

utilizou-se uma webcam perpendicularmente situada a ~2,60m acima do cenário, conforme mostrado de forma esquemática na Figura 2.

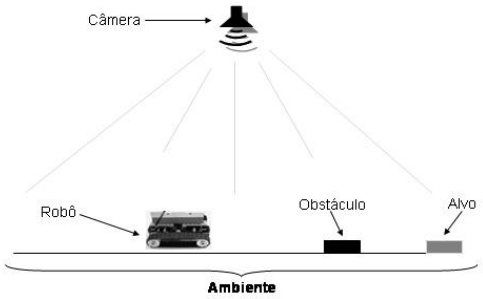
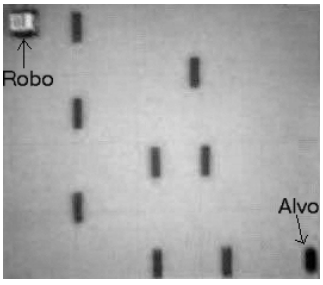


Figura 2. Arranjo esquemático do cenário.

A altura escolhida para a captura das imagens mostrou-se satisfatória, pois todos os componentes do cenário ficaram bem definidos. A partir das imagens capturadas, o sistema de visão computacional deve identificar, usando coordenadas cartesianas, a posição dos obstáculos, a posição do robô e o local de destino.

2.2.3. Processamento de imagens

Quando o sistema de visão é acionado, uma imagem colorida com resolução de 320x240 é capturada e armazenada. Esta imagem é denotada por C^y. A partir de C^y, primeiramente, detecta-se as posições do robô e do alvo dado a facilidade de localizá-los por meio dos componentes RGB dos *pixels* que o compõem (Quiles & Romero, 2004; Cohen et al., 2002). Este procedimento para localização do robô e do alvo é descrito na seção 2.2.4. Cabe ressaltar que, embora o modelo RGB tenha apresentado facilidades na operação, há o inconveniente da sensibilidade à iluminação, o que talvez possa ser contornado usando o modelo HSI. A etapa seguinte consiste na localização dos demais objetos (obstáculos). Para isso, a imagem em níveis de cinza relativa à componente R da imagem C^y, denotada por G_R^y, é extraída. Para eliminação de ruídos em G_R^y é feita uma filtragem gaussiana na mesma usando σ=1,2 (o valor do parâmetro σ foi escolhido após diversos experimentos). A imagem G_R^y filtrada é então binarizada utilizando um limiar (*threshold*)=80. A próxima etapa do processamento é a transformação da imagem binária, denotada por B^y, na matriz $M(i, j)$. Para cada célula da matriz é atribuído um valor inteiro V que assume o valor 0 quando a célula está livre, 1 quando a célula possui obstáculo, 2 para célula onde está o robô e 3 para célula que contém o alvo. Essa matriz é posteriormente utilizada pelo algoritmo de roteamento na determinação do trajeto do robô.



3a. Imagem G_R^y.

3b. Imagem B^y.

2	**1**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**1**	**0**	**0**
0	**0**	**0**	**1**	**1**	**1**	**3**

3c. A matriz M obtida a partir de B^y.

Figura 3. Etapas do processamento de uma imagem do ambiente.

2.2.4. Reconhecimento e localização do robô e do alvo

A tarefa de reconhecimento do robô e do alvo é feita após uma "varredura" na imagem C^y observando os valores RGB dos *pixels*. Para localizar o robô, os componentes RGB de cada *pixel* $P(i, j) \in C^y$ são analisados de acordo com a equação 2 para verificar se $P(i, j)$ faz parte da região da imagem que representa o robô. A idéia é que há grande probabilidade de $P(i, j)$ pertencer a esta região quando os valores de R e G são altos e o valor de B é baixo. Desta forma, para cada $P(i, j)$ lido em C^y, é atribuído um valor (0 ou 1) ao *pixel* situado na mesma posição numa imagem binária temporária denotada por B_t^y. Ao final da varredura de C^y, a imagem B_t^y é analisada e os componentes conexos com poucos pixels são descartados. O centro do componente conexo restante é utilizado para indicar a localização do robô. A localização do alvo é feita de maneira análoga, porém utilizando a equação 3.

$$B_t^y(i, j) = \begin{cases} 0, se \frac{R+G}{2}) > 2 * B \\ 1, caso\ contrário \end{cases} \quad \forall P(i, j) \in C^y \tag{2}$$

$$B_t^y(i, j) = \begin{cases} 0, se\ B > R + G \\ 1, caso\ contrário \end{cases} \quad \forall P(i, j) \in C^y \tag{3}$$

A partir deste processamento, são definidas as coordenadas cartesianas que indicam o posicionamento do robô e do alvo e os valores 2 e 3 indicando suas posições são atribuídos aos elementos correspondentes da matriz M.

2.3. Cálculo da trajetória do robô

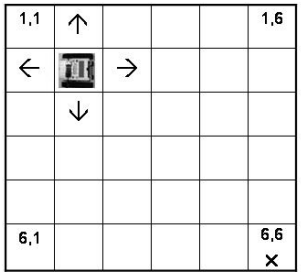
O trajeto do robô é calculado por meio de algoritmos de busca tradicionais usando "árvores" de busca no espaço de estados. Como o ambiente é descrito pelo sistema de visão como uma matriz, definiu-se que o robô poderia movimentar-se nas direções 0º, 90º, 180º e 270º (Figura 4a). A escolha destas direções visou diminuir o espaço de estados o que reduz o tempo para o cálculo do trajeto. Utilizou-se como solução o algoritmo de busca pela melhor estimativa (Figura 4b) que embora não seja ótimo, é adequado para o problema visto que combina as vantagens das buscas em amplitude e profundidade (Russel & Norvig, 2002) e que o custo de cada passo do robô é unitário. Além disso, a função de avaliação heurística utilizada pelo algoritmo de busca $f(n) = h(n)$, na qual $h(n)$ é o custo estimado do caminho a ser percorrido desde a célula da matriz onde se encontra o robô até a célula definida como estado meta, pode ser facilmente calculada pelo sistema de visão, a partir da descrição do ambiente. Diante desta facilidade, foram utilizadas para $h(n)$, 3 diferentes medidas de distância conhecidas como City-Block (d_b), Chessboard (d_c) e Distância Euclidiana (d_e) (Gonzalez & Woods, 2002), as quais são dadas, respectivamente, pelas equações (4), (5) e (6).

$$d_b(p, q) = |x - u| + |y - v| \tag{4}$$

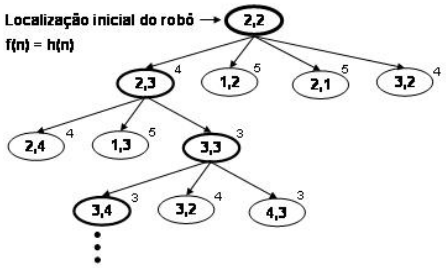
$$d_c(p, q) = \max(|x - u|, |y - v|) \tag{5}$$

$$d_e(p, q) = \sqrt{(x - u)^2 + (y - v)^2} \tag{6}$$

onde $p(x, y)$ e $q(u, v)$ são dois pontos no espaço cartesiano.



4a. Descrição do ambiente do robô



4b. Árvore de busca. Cada nó da árvore de busca corresponde a uma célula da matriz.

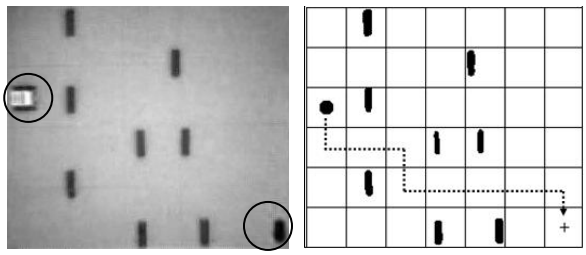
Figura 4. Descrição do ambiente do robô ilustrando suas possibilidades de movimentação (4a) e exemplo da árvore de busca (4b) usando a heurística Chessboard para cálculo do trajeto do robô considerando o ambiente descrito em 4a.

2.4. Acionamento do robô

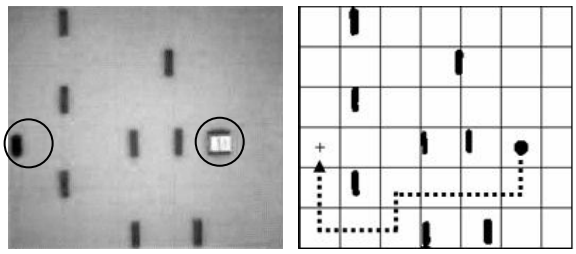
Uma vez encontrada uma rota a partir das heurísticas propostas, as coordenadas deste roteamento são transformadas em um pseudocódigo de comando do robô denominado NQC (*Not Quiet C*) e transmitidas por um transmissor infravermelho. Estas informações são captadas por um receptor embarcado no robô. A transmissão das informações é realizada em ~500ms e após isso, o robô executa a seqüência de movimentos prevista na solução encontrada.

3. Resultados e discussões

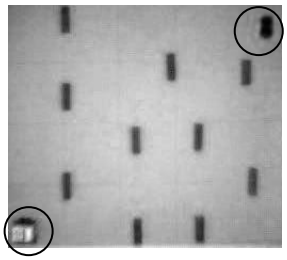
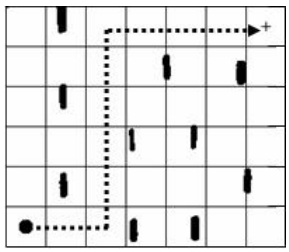
Os experimentos foram realizados variando-se a disposição dos objetos no cenário e utilizando-se as três funções heurísticas descritas na seção anterior. O tamanho do cenário utilizado foi fixado em 2x2m e as condições de iluminação do ambiente foram controladas. Para determinar o número de células da matriz M, fixou-se em número de *pixels* os valores do comprimento e da largura de uma área na imagem C^y a ser processada respeitando as dimensões físicas do robô na imagem. Assim, em todos os experimentos, M possui o tamanho 6x7. A Figura 5 a seguir ilustra os resultados obtidos para diferentes cenários utilizando a distância City-Block (Eq. 4) como função heurística.



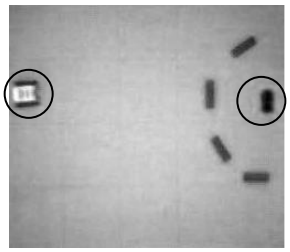
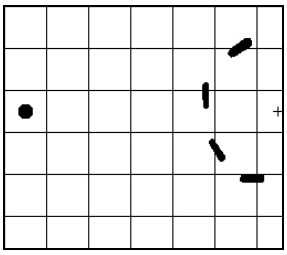
5a. Cenário com 8 obstáculos. O robô está posicionado na parte esquerda e o alvo na parte inferior direita.



5b. Cenário com 8 obstáculos. O robô está posicionado na parte direita e o alvo na parte esquerda.

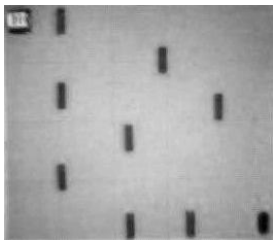
5c. Cenário com 10 obstáculos. O robô está posicionado na parte inferior esquerda e o alvo na parte superior direita.

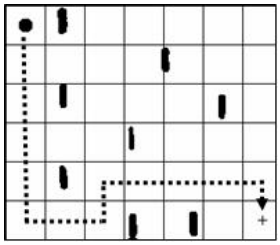
5d. Cenário com 4 obstáculos cercando o alvo. Neste caso a impossibilidade do robô atingir o alvo é detectada pelo algoritmo de roteamento.

Figura 5. Soluções encontradas para diferentes cenários usando a heurística $h(n) = d_b$.

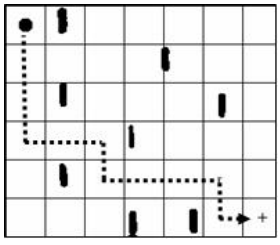
Com intuito de verificar as melhores soluções considerando as limitações do sistema, foram realizados testes empregando as três funções heurísticas no algoritmo de roteamento do robô em cada cenário. Nos experimentos realizados, cujos resultados são descritos na Tabela 1, verificou-se que os melhores resultados foram obtidos com a função heurística baseada na distância City-Block. Observou-se também que as três heurísticas apresentaram resultados muito próximos, entretanto as heurísticas baseadas nas distâncias City-Block e Euclidiana, em geral, convergiram para soluções melhores. Um exemplo disto pode ser visto Figura 6.



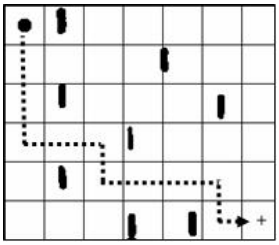
6a. Cenário com 8 obstáculos.



6b. Solução usando a heurística $h(n) = d_b$.



6c. Solução usando a heurística $h(n) = d_c$.



6d. Solução usando a heurística $h(n) = d_e$.

Figura 6. Comparativo entre as soluções encontradas usando as três heurísticas.

Tabela 1. Comparativo entre as soluções encontradas usando as 3 heurísticas.

Cenário		**Custo da solução para as diferentes funções heurísticas**			**Cenário**		**Custo da solução para as diferentes funções heurísticas**		
Nº.	Qtd. de Obstáculos	d_b	d_c	d_e	Nº.	Qtd. de Obstáculos	d_b	d_c	d_e
1	4	8	10	10	6	10	11	11	11
2	8	13	11	11	7	10	11	11	11
3	8	11	9	9	8	10	8	8	8
4	8	13	11	11	9	11	13	11	11
5	8	7	9	9	10	12	11	9	9
					Custo médio		**10,6**	**10**	**10**

Observando as Figuras 5 e 6 é possível inferir que as rotas encontradas foram satisfatórias. O tempo total gasto pelo sistema para o processamento da imagem e apresentação de uma solução para um dado cenário foi de ~1s, evidenciando que pode ser empregado em aplicações de navegação autônoma em tempo real. Vale ressaltar que sistemas de navegação autônoma de robôs por meio de visão requerem cuidado com o tempo de processamento, controle de ruídos na imagem além do tratamento da iluminação diferenciada em partes diferentes do ambiente.

4. Conclusões

Neste estudo foi desenvolvido um sistema de navegação autônoma utilizando o Kit Lego Mindstorms, o qual apresentou resultados significativos. O algoritmo de roteamento usando como função heurística as medidas de distância City-Block e Euclidiana convergiu para soluções melhores quando comparadas com as soluções obtidas usando a distância Chessboard. O tempo de resposta do sistema (~1s) mostrou-se adequado para a aplicação visto que sistemas de navegação autônoma de robôs requerem rapidez na tomada de decisões. Para trabalhos futuros pretende-se avaliar outras formas de reconhecimento dos objetos visando diminuir a necessidade do controle sobre as condições do ambiente. Além disso, pretende-se implementar um algoritmo de roteamento híbrido para melhorar a qualidade das rotas usando os algoritmos de busca tradicionais e os algoritmos genéticos.

Agradecimentos

Os autores agradecem a Uninove pelo apoio financeiro.

Referências

Araújo, S. A. & Kim, H. Y. (2005) “Meio-Tom Inverso Usando Redes Neurais Artificiais”. In: Anais do XXII Simpósio Brasileiro de Telecomunicações (SBrT'05), Campinas–SP, vol. 1. p. 226-231.

Cazangi, R. R. & Figueiredo, M. F. (2000) “Sistema de Navegação Autônoma de Robôs Baseado em Sistema de Classificação com Aprendizado e Algoritmos Genéticos”. In: Anais do II Fórum de Informática e Tecnologia de Maringá e V Mostra de Trabalhos em Informática da UEM, Maringá-PR, p. 29–29.

Cohen H. et al. (2002) “Vision based pursuing of moving vehicle from bird's view – PART I and PART II”. The Vision and Image Sciences Laboratoy– TECHNION – Israel Institute Technology.

Gonzalez, R. C. & Woods, R. E. (2002) “Digital Image Processing”. 2. nd., Prentice Hall, New Jersey.

Kim, H. Y. (2005) “Sistema IMG”. LPS/POLI/USP. <http://www.lps.usp.br/~hae/>. Acesso em 12/07/2005.

Pomerleau, D. (1995) “Neural Network Vision for Robot Driving”. In: The Handbook of Brain Theory and Neural Networks.

Pratt, W. K. (1991) “Digital image processing”, John Wiley, New York, 1991.

Quiles, M. G. & Romero, R. A. F. (2004) “Um Sistema de Visão Computacional Baseado em Cores Aplicado ao Controle de um Robô Móvel”. In: Anais do IV Congresso Brasileiro de Computação, Itajaí-SC, p. 379–383.

Russel, S. & Norvig, P. (1995) “Artificial intelligence a modern approach”, Prentice Hall, New Jersey.

Shiba, M. H. et al. (2005) “Classificação de imagens de sensoriamento remoto pela aprendizagem por árvore de decisão: uma avaliação de desempenho”, In: Anais do XII Simpósio Brasileiro de Sensoriamento Remoto, Goiânia-GO, p. 4319-4326.

Sun, Z. et al. (2002) “A Real-time Precrash Vehicle Detection System”. In: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision (WACV’02), Orlando-FL, p. 01-06.