Visão e inteligência computacionais aplicadas a navegação autônoma de robôs

Sidnei Alves de Araújo[1, 3],
André Felipe Henriques Librantz[2, 3]
[1]EP-USP/[2]Ipen-USP/[3]Uninove
saraujo@uninove.br

Diante da necessidade de melhorar a qualidade dos produtos e otimizar o tempo dos processos de sua fabricação, as pesquisas relativas à automação e robótica têm sido intensificadas. Neste trabalho, apresenta-se 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, com base nas informações recebidas sobre seu posicionamento, posição do alvo e dos obstáculos, deve decidir o trajeto a fazer para atingir o objetivo. Neste estudo, são discutidos detalhes da implementação e das técnicas utilizadas bem como os resultados experimentais.

Palavras-chave: Inteligência computacional. Navegação autônoma. Robótica. Visão computacional.

1 Introdução

Nos últimos anos, tem havido 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 a otimização do tempo. Esses sistemas, em sua maioria, utilizam técnicas de inteligência artificial (IA) empregadas na construção dos algoritmos propostos para solucionar os problemas. Por isso, sistemas autônomos de navegação robótica que permitem a tomada de decisão com base em informações extraídas do ambiente, que proporcionam a cooperação de agentes ou possuem visão computacional, têm sido largamente explorados em pesquisas nas áreas de automação, robótica e IA, o que gera muitas propostas de aplicações em vários segmentos.

Robôs de navegação autônoma são objetos de grande admiração em face de sua "inteligência" para se deslocar de maneira independente. Associar características da inteligência humana a máquinas é, no mínimo, interessante. Entretanto, vale lembrar que os robôs móveis, mesmo com toda a tecnologia, ainda apresentam muitas limitações quanto à sua capacidade de navegação (CAZANGI; FIGUEIREDO, 2000). A navegação autônoma 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 autônoma, além de essenciais para achar 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 em que podem estar empenhados (HACOHEN; COHEN, 2002a; 2002b). O estudo e desenvolvimento de um projeto de robôs autônomos podem estar ligados a diversas aplicações práticas e têm suscitado grandes desafios, em razão de as dificuldades se multiplicarem à medida que os ambientes de navegação vão se tornando mais imprevisíveis e diversificados. Diversos mecanismos para o controle de robôs móveis, tais como sonares, *lasers* e visão, têm sido utilizados (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. Por meio do sistema de visão, informações do ambiente, referentes tanto à sua localização quanto à do alvo e dos obstáculos dispostos aleatoriamente, são transmitidas ao robô para que ele possa tomar decisões. Com 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, uma câmera de vídeo é posicionada sobre o ambiente, o que propicia que se tenha uma visão panorâmica do local. A partir das imagens da câmera, o sistema de visão computacional "mapeia" o ambiente no qual o robô tem de mover-se. Esse 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, em que 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 três diferentes

funções heurísticas. Tanto as técnicas utilizadas na construção do robô quanto as implementações do sistema de visão e do cálculo do trajeto são descritas a seguir.

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 (2004)[1].

2.1 Construção do robô utilizando o kit Lego Mindstorms

O robô foi construído com o *kit Lego Mindstorms*, no formato retangular, com dimensões de, aproximadamente, 10 x 12 x 15 centímetros (cm) (Figura 1). Visando a dar maior manobrabilidade em espaços limitados, o robô foi dotado de esteiras com dois eixos. Embora o *kit Lego Mindstorms* possua vários sensores e outros acessórios, o único utilizado é uma câmera de vídeo que gera as imagens para que o sistema de visão possa fornecer os dados que servem de base para tomadas de decisão.

2.2 O sistema de visão

Nessa seção, são apresentados, de forma sucinta, alguns conceitos teóricos, além dos procedimentos empregados no sistema de visão. As etapas do processo compreendem tanto a aquisição quanto a análise das imagens que permitem não só reconhecer o robô, mas também 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, enquanto o valor do elemento da matriz determinar o nível de cinza do pixel (GONZALEZ; WOODS, 2002; PRATT, 2001). Normalmente, uma imagem digital é definida como uma função bidimensional f(i, j), com i, j e z2. Uma imagem em níveis de cinza Gy pode ser descrita como uma função em que o contradomínio representa diferentes níveis de cinza. Para denotar os níveis de cinza, geralmente se utiliza o intervalo discreto [0,255] (ARAÚJO; KIM, 2005). Uma imagem colorida Cy é multibanda, na qual a cor de cada pixel é representada de acordo com um modelo de cor (SHIBA et al., 2005). Os modelos mais comuns em processamento de imagens são o RGB (do red, green, blue – em português, vermelho,

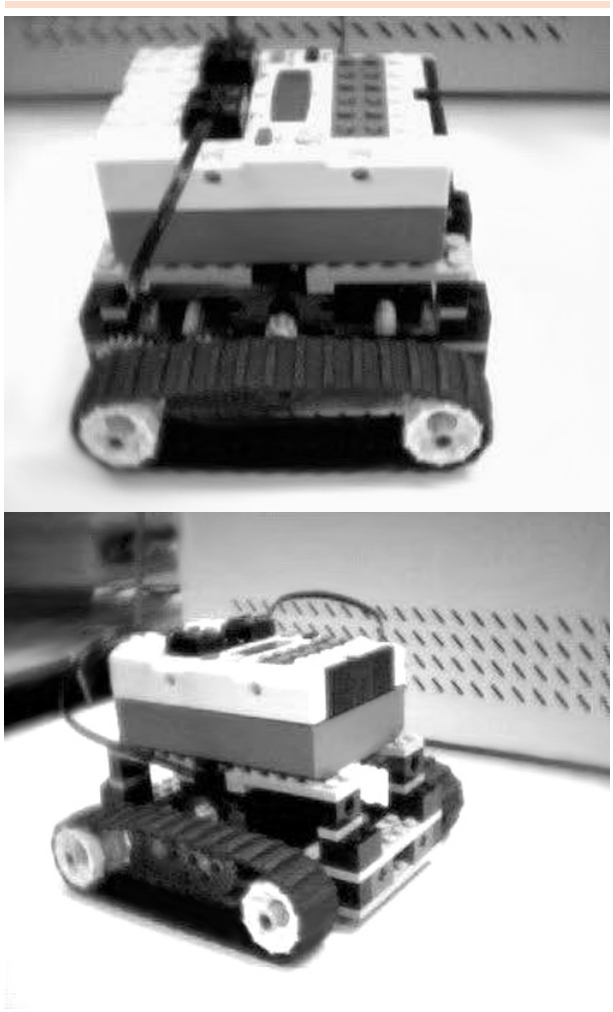


Figura 1: Fotos do protótipo do robô montado. O formato e a cor do robô facilitam distingui-lo dos demais objetos do cenário

Fonte: O autor.

verde, azul) e o HSI2 (em inglês hue, saturation, intensity – em português, tonalidade, saturação, intensidade). Segundo Gonzalez e Woods (2002), no modelo de cor RGB, uma imagem colorida pode ser vista como um conjunto de três imagens em níveis de cinza independentes (GyRGB), cada uma delas representando uma cor. No modelo HSI, as três imagens em níveis de cinza (GyHSI) 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.

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

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 isso, posicionou-se, perpendicularmente, uma webcam, a 2,6 metros (m) acima do cenário (Figura 2).

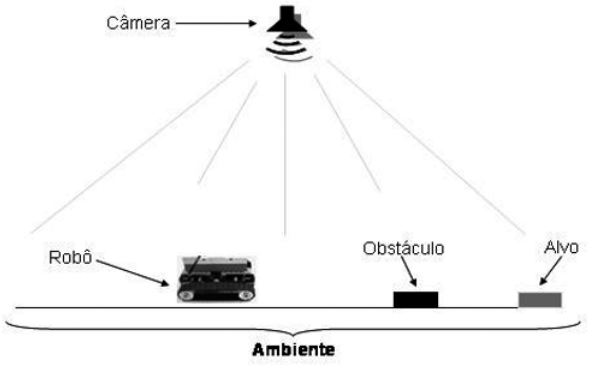
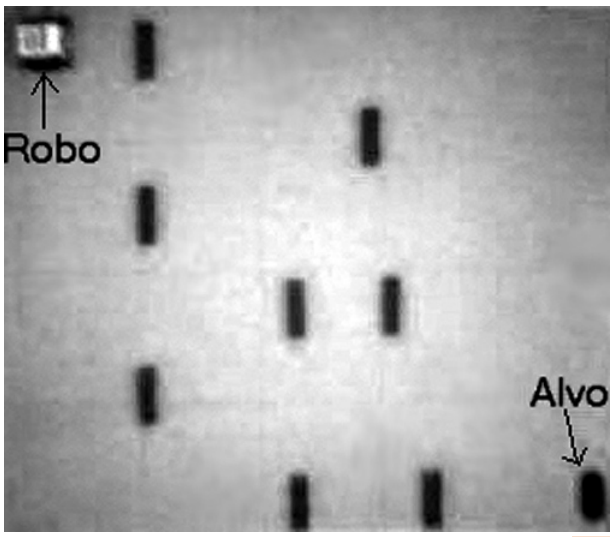


Figura 2: Arranjo esquemático do cenário
Fonte: O autor.

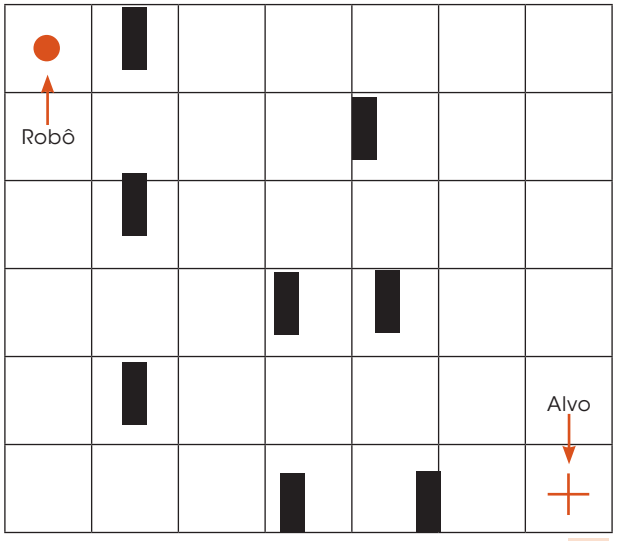
A altura escolhida para a captura das imagens mostrou-se satisfatória, pois todos os componentes do cenário apresentaram boa definição. A partir das imagens capturadas, o sistema de visão computacional deve identificar, usando coordenadas cartesianas, a posição tanto dos obstáculos quanto do robô, além do local de destino.

2.2.3 Processamento de imagens

Quando o sistema de visão é acionado, uma imagem colorida com resolução de 320 x 240 pixels é capturada e armazenada. Essa imagem é denotada por C^{y}. A partir de C^{y}, num primeiro momento, detectam-se as posições do robô e do alvo, em razão da facilidade de localizá-los por meio dos componentes RGB dos pixels que o compõem (QUILES; ROMERO, 2004; HACOHEN; COHEN, 2002a; 2002b). 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 com o uso do 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 C^{y}_{R}, é extraída. Para eliminação de ruídos em G^{y}_{R}, é feita uma filtragem gaussiana da imagem, usando σ = 1,2 (o valor do parâmetro σ foi escolhido após diversos experimentos). A imagem G^{y}_{R} 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 "situação" da célula da matriz, é atribuído um valor inteiro V, que assume o valor 0 quando está livre; 1 quando está com o obstáculo; 2 quando está com o robô e 3 quando está com o alvo. Essa matriz é, posteriormente, utilizada pelo algoritmo de roteamento na determinação do trajeto do robô.



(a)



(b)

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

(c)

Figura 3: Etapas do processamento de uma imagem do ambiente
a) Imagem GyR;
b) Imagem By;
c) Matriz M obtida a partir de By.
Fonte: O autor.

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-se 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 essa região quando os valores de R e G forem altos, e o valor de B, baixo. Dessa 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^y_t. Ao final da varredura de C^y, a imagem B^y_t é analisada, e os componentes conexos com poucos *pixels*, descartados. O centro do componente conexo restante é utilizado para indicar tanto a localização do robô quanto a do alvo que, por sua vez, é feita utilizando a Equação 3.

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

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

A partir desse processamento, são definidas as coordenadas cartesianas que indicam o posicionamento do robô e do alvo, e os valores 2 e 3 que determinam suas posições, 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 movimen-

tar-se nas direções 0°, 90°, 180° e 270° (Figura 4a). A escolha dessas direções visou a 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 ao problema, pois combina as vantagens das buscas em amplitude e profundidade (RUSSEL; NORVIG, 1995). Ressalte-se 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) compreende o custo estimado do caminho a ser percorrido desde a célula da matriz em que 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 dessa facilidade, foram utilizadas para h(n) três diferentes medidas de distância, conhecidas como *city-block* (d_b), *chessboard* (d_c) e distância euclidiana (d_e) (GONZALEZ; WOODS, 2002), dadas, respectivamente, pelas Equações 4, 5 e 6.

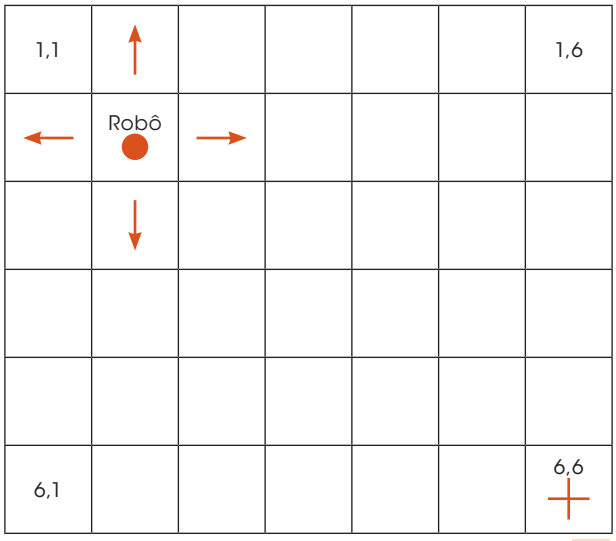


Figura 4. Descrição do ambiente do robô ilustrando suas possibilidades de movimentação

Obs.: 4a) e exemplo da árvore de busca; 4b) usando a heurística *chessboard* para cálculo do trajeto do robô, considerando o ambiente descrito em 4a

Fonte: O autor.

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

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

$$d_e(p, q) = \sqrt{(x - u)^2 + (y - v)^2} \quad (5)$$

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

2.4 Acionamento do robô

Uma vez encontrada uma rota a partir das heurísticas propostas, as coordenadas desse roteamento são transformadas em um pseudocódigo de comando do robô, denominado *Not Quiet C* (NQC), e transmitidas por um transmissor infravermelho. Essas informações são captadas por um receptor embarcado no robô e transmitidas em, aproximadamente, 500 microssegundos (ms). Em seguida, 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 usado foi fixado em 2 x 2 m e as condições de iluminação do ambiente foram controladas. Para determinar o número de células da matriz M, fixaram-se em

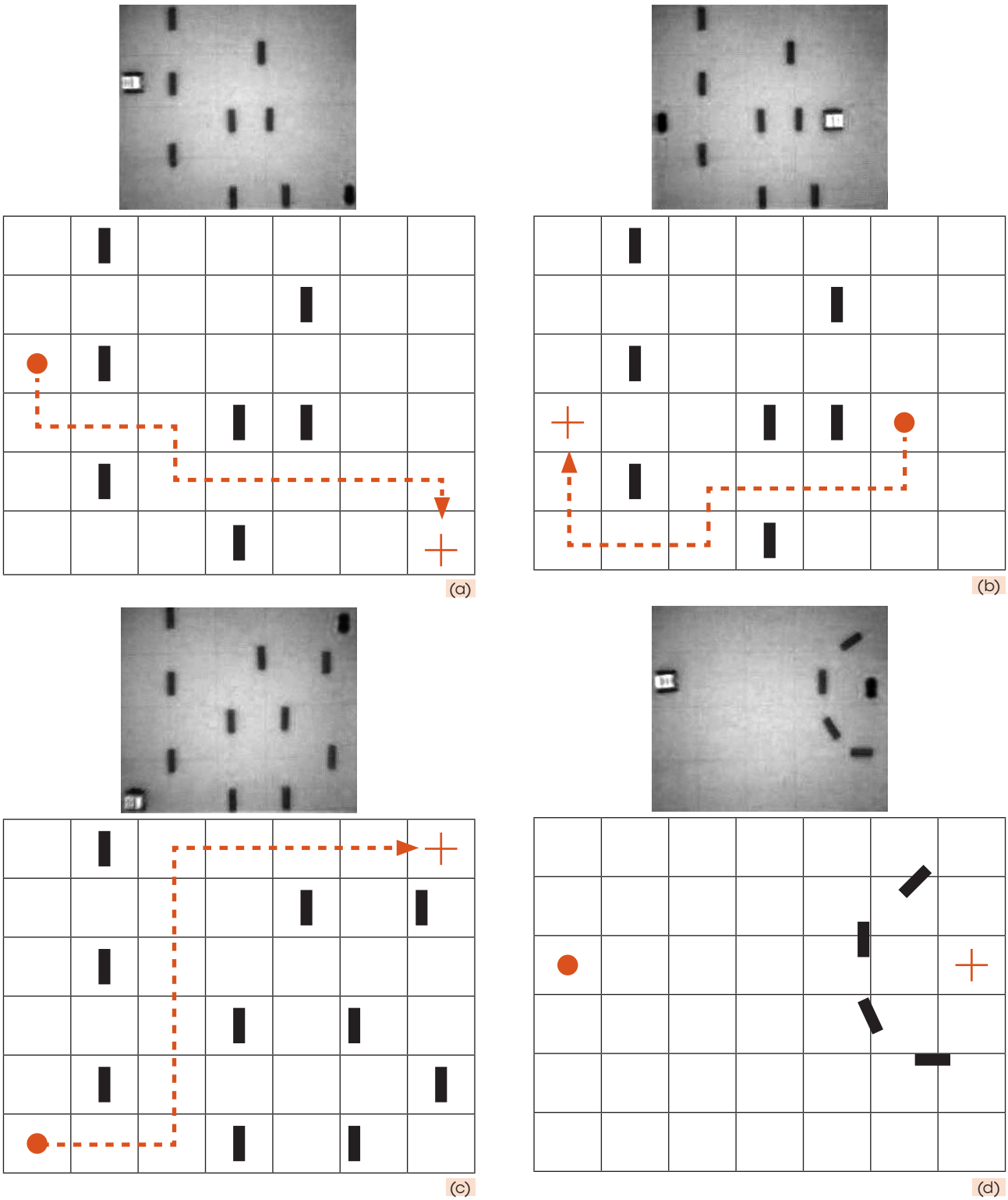


Figura 5: Soluções encontradas para diferentes cenários usando a heurística h(n) = d_b

Obs.: 5a) Cenário com oito obstáculos. O robô está posicionado na parte esquerda, e o alvo, na parte inferior direita; 5b) Cenário com oito obstáculos. O robô está posicionado na parte direita, e o alvo, na parte esquerda; 5c) Cenário com dez obstáculos. O robô está posicionado na parte inferior esquerda, e o alvo, na parte superior direita; 5d) Cenário com quatro obstáculos cercando o alvo. Nesse caso, a impossibilidade de o robô atingir o alvo é detectada pelo algoritmo de roteamento.

Fonte: O autor.

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 6 x 7 células. A Figura 5 a seguir ilustra os resultados obtidos para diferentes cenários, utilizando a distância *city-block* (Equação 4) como função heurística.

Com intuito de verificar as melhores soluções, considerando as limitações do sistema, foram realizados testes empregando, em cada cenário, as três funções heurísticas no algoritmo de roteamento do robô. 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; no entanto, as baseadas nas distâncias *city-block* e euclidiana, em geral, convergiram para soluções melhores (Figura 6).

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 a apresentação de uma solução para um dado cenário foi de aproximadamente 1

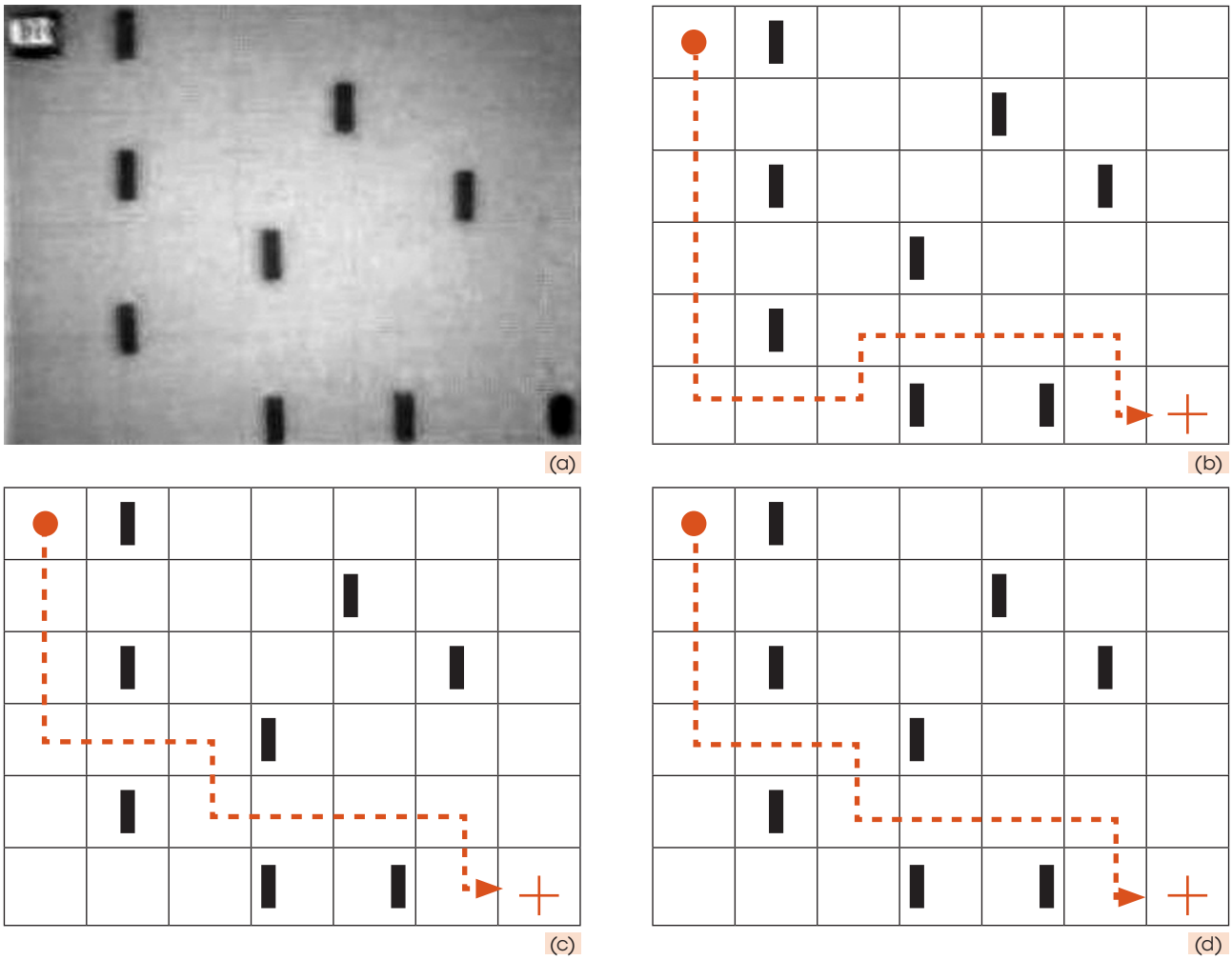


Figura 6: Comparativo entre as soluções encontradas usando as três heurísticas

Obs.: (a) Cenário com oito obstáculos; (b) Solução usando a heurística h(n) = d_b; (c) Solução usando a heurística h(n) = d_c; (d) Solução usando a heurística h(n) = d_e.

Fonte: O autor.

Tabela 1: Comparativo entre as soluções encontradas usando as três heurísticas

Custo da solução para as diferentes funções heurísticas				
Cenário	Obstáculos	db	dc	de
1	4	8	10	10
2	8	13	11	11
3	8	11	9	9
4	8	13	11	11
5	8	7	9	9
6	10	11	11	11
7	10	11	11	11
8	10	8	8	8
9	11	13	11	11
10	12	11	9	9
Custo médio		10, 6	10	10

Fonte: O autor.

segundo (s). Esse resultado evidenciou 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 Considerações finais

Neste estudo, foi desenvolvido um sistema de navegação autônoma de robôs utilizando o *kit* Lego Mindstorms, que 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 às obtidas com o uso da distância *chessboard*. O tempo de resposta do sistema (aproximadamente 1 s) mostrou-se adequado à aplicação, visto que esse tipo de sistema de navegação requer rapidez nas tomadas de decisão. Para trabalhos futuros, pretende-se avaliar outras formas de reconhecimento dos objetos, com o intuito de diminuir a necessidade do controle sobre as condições do ambiente. Além disso, intenta-se implementar um algoritmo de roteamento híbrido para melhorar a qualidade das rotas, usando os algoritmos de busca tradicionais e os genéticos.

Vision and computational intelligence applied to autonomous navigation of robots

The necessity of improving the products quality and to optimize the processing time regarding the manufacturing of them, led to the growing of researches related to the automation and robotics. 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 one robot to make decisions. This robot, once having information about its localization, target and the obstacles positions, must decide which path to choose in order to get the target. Implementation, applied techniques details and results, as well, are discussed in this work.

Key words: Autonomous navigation. Computational intelligence. Computational vision. Robotics.

Notas

1 N. Ed.: Hoje, o pesquisador desenvolve o ProEikon (com rotinas e programas em C++ para processamento de imagens e visão computacional), uma versão mais atualizada do IMG. Para saber mais, acesse http://www.lps.usp.br/~hae/software.

2 N. Ed.: Método de exibição de cores conforme a sua intensidade e variações de cores.

Referências

ARAÚJO, S. A.; KIM, H. Y. Meio-tom inverso usando redes neurais artificiais. In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, 22., 2005. Campinas. *Proceedings*... Campinas: SBrT. v. 1. p. 226-231, 2005.

CAZANGI, R. R.; FIGUEIREDO, M. F. Sistema de navegação autônoma de robôs baseado em sistema de classificação com aprendizado e algoritmos genéticos. In: ENCONTRO ANUAL DE INICIAÇÃO CIENTÍFICA, 9., 2000, Londrina. *Anais*... Londrina: p. 29, 2000.

GONZALEZ, R. C.; WOODS, R. E. *Digital image processing*. 2. ed. Nova Jersey: Prentice Hall, 2002.

HACOHEN, A.; COHEN, H. *Vision based pursuing of moving vehicle from bird's view: part I*. Haifa: Visl-Technion, 2002a. Disponível em: <http://visl.technion.ac.il/projects/2002w07/>. Acesso em: 4 set. 2006.

HACOHEN, A.; COHEN, H. *Vision based pursuing of moving vehicle from bird's view: part II*. Haifa: Visl-Technion, 2002b. Disponível em: <http://visl.technion.ac.il/projects/2002s07/>. Acesso em: 4 set. 2006.

KIM, H. Y. *Sistema IMG. Biblioteca de programas e rotinas para processamento e análise de imagens*. São Paulo: LPS-EP-USP, 2004.

POMERLEAU, D. A. Neural network vision for robot driving. In: ARBID, M. *The handbook of brain theory and neural networks*. 1. ed. Cambridge: The MIT Press, 1995. p. 1.008-1.009.

PRATT, W. K. *Digital image processing: piks inside*. 3. ed. Nova York: John Wiley & Sons, 2001.

QUILES, M. G.; ROMERO, R. A. F. Um sistema de visão computacional baseado em cores aplicado ao controle de um robô móvel. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 4., 2004. Itajaí. *Anais*... Itajaí: CBComp, p. 379–383, 2004.

RUSSEL, S.; NORVIG, P. *Artificial intelligence a modern approach*. 1. ed. Nova Jersey: Prentice Hall, 1995.

SHIBA, M. H. et al. Classificação de imagens de sensoriamento remoto pela aprendizagem por árvore de decisão: uma avaliação de desempenho. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 12., 2005, Goiânia. *Anais*... Goiânia: SBSR, p. 4.319-4.326, 2005. Disponível em: <http://www.lps.usp.br/~hae/sbsr2005.pdf>. Acesso em: 4 set. 2006.

SUN, Z. et al. A real-time precrash vehicle detection system. In: WORKSHOP ON APPLICATIONS OF COMPUTER VISION, 6., 2002. Orlando. *Proceedings*... Orlando: WACV-IEEE, p. 171-176, 2002.

Recebido em 4 set. 2006 / aprovado em 10 nov. 2006

Para referenciar este texto

ARAÚJO, S. A. de; LIBRANTZ, A. F. H.; FLÓRIO FILHO, O. Navegação autônoma de robôs: uma implementação utilizando o kit Lego Mindstorms. *Exacta*, São Paulo, v. 4, n. 2, p. 343-352, jul./dez. 2006.