UTILIZANDO A ROBÓTICA COMO RECURSO DIDÁTICO NO ENSINO DE ENGENHARIA E COMPUTAÇÃO

Rafael Gomes Mantovani[1], Valter Takahiro Kamiji[2], Pedro Paulo da Silva Ayrosa[3]

Abstract *— The task of mobile robot navigation can be understood as a problem of determining the trajectories with the aim of providing conditions that allow the execution of tasks. One type of problem that has attracted attention from scientific community is the autonomous navigation: the problems where the agent is who determines its trajectory in an unknown environment. The interdisciplinary, from areas such as computer graphics, artificial intelligence and related ones, assists in learning and development of knowledge in technology. In this paper we describe an experiment involving autonomous navigation of robots, using the model NXT robotics kit from Lego Mindstorms, widely used in educational robotics. Through mechanisms of computer vision, environmental characteristics are transmitted to the robot so that decision can be taken. This paper explores the possibility of using teaching kits for the treatment of problems of engineering and computing that students may encounter in their future professional life.*

Index Terms — Autonomous navigation, Lego Mindstorms, Teaching Engineering and computation.

INTRODUÇÃO

O campo científico de robótica móvel é relativamente jovem. Suas raízes incluem uma variedade de disciplinas, compartilhando conceitos de mecânica, engenharia elétrica e eletrônica, computação, ciências cognitivas e até mesmo sociais. A aplicação de tais modelos é extremamente vasta. Agentes móveis podem ser designados para exploração de ambientes inóspitos, perigosos e de difícil acesso aos seres humanos, como também podem dividir espaço com os seres humanos em ambientes tradicionais ([1]).

O projeto de desenvolvimento de um robô móvel envolve a integração de muitos campos do conhecimento. Para resolver problemas de locomoção, o projetista precisa conhecer conceitos de cinemática, dinâmica e teoria de controle. Para criar sistemas de percepção robustos, ele precisa demonstrar conhecimento de análise de sinais e campos específicos como visão computacional para empregar corretamente uma vastidão de sensores tecnológicos. Já a localização e navegação demandam compreensão de conhecimento de algoritmos computacionais, teoria da informação, inteligência artificial e teoria de probabilidade ([1]).

Um tipo de problema que tem atraído atenção da comunidade cientifica é a navegação autônoma: navegação onde o próprio agente, sem ajuda externa, determina a sua trajetória em um ambiente desconhecido ([2]). Um agente ser autônomo significa que, além de agir por conta própria, ele tem a capacidade de se auto-regular gerando as próprias regras que regem sua atuação. Pode-se dizer também, que um agente autônomo é automático, pois ele tem a capacidade de operar em um ambiente, percebê-lo e impactá-lo visando o cumprimento de tarefas definidas ([3]). Além disso, ele deve se autodirigir, com base em sua capacidade de aprender e adaptar seus comportamentos. Os processos de aprendizagem e adaptação devem ocorrer quando o agente está operando no ambiente.

Agentes autônomos artificiais são ideais para se estudar os princípios de inteligência. Segundo [4], uma das motivações para o uso destes agentes envolve a idéia de emergência. Agentes autônomos apresentam comportamentos emergentes, que surgem pela interação do agente com o ambiente sem que tenham sido programados a *priori* pelo projetista. A pesquisa de sistemas inteligentes autônomos é classificada como uma metodologia sintética, cuja idéia se resume em "construir para entender" ([3]).

O objetivo deste trabalho é realizar um experimento de navegação autônoma de robôs, baseado no trabalho descrito em [5]. O experimento consiste do desenvolvimento de um agente de navegação autônoma utilizando o *kit* de robótica Lego Mindstorms. Através de mecanismos de visão computacional, informações do ambiente são transmitidas ao robô para que o mesmo possa tomar decisões. As informações passadas ao robô correspondem à sua localização, localização do objetivo final e de possíveis obstáculos dispostos aleatoriamente no ambiente. Usando tais informações, o trajeto é calculado por um algoritmo de busca que, utilizando de funções heurísticas, tenta escolher o melhor percurso dentro do ambiente previamente mapeado.

1 Rafael Gomes Mantovani, LABRE - Laboratório de Robótica Educacional, Departamento de Computação - Universidade Estadual de Londrina, Caixa Postal 6041 – CEP 86057-980, Londrina, PR, Brasil, rgmantovani@gmail.com
2 Valter Takahiro Kamiji, LABRE - Laboratório de Robótica Educacional, Departamento de Computação - Universidade Estadual de Londrina, Caixa Postal 6041 – CEP 86057-980, Londrina, PR, Brasil, vtkamiji@gmail.com
3 Pedro Paulo da Silva Ayrosa, LABRE - Laboratório de Robótica Educacional, Departamento de Computação - Universidade Estadual de Londrina, Caixa Postal 6041 – CEP 86057-980, Londrina, PR, Brasil, ayrosa@uel.br

METODOLOGIA

Para execução do experimento desenvolvido, baseado nos trabalhos de [5], dividimos a aplicação em dois módulos principais: o Módulo de Navegação Autônoma (MNA) e o Módulo de Processamento Visual (MPV).

O MPV é responsável por elaborar uma representação do ambiente, através de sensoriamento prévio. Uma câmera localizada acima do ambiente disponibiliza uma visão panorâmica do ambiente por completo. A partir de imagens capturadas pela câmera, o MPV "mapeia" o ambiente no qual o agente desenvolvido terá de se locomover. Este mapeamento fornece a posição do agente, a posição do objetivo final, e os eventuais obstáculos que possam atrapalhar o movimento do agente dentro do ambiente. As posições do ambiente são tratadas como coordenadas do plano cartesiano. Desta forma, o mapa do ambiente é descrito através de uma matriz *M* onde as dimensões de cada célula *M(i,j)* são baseadas nas dimensões físicas do agente ([5]). Após todo esse processo, o fluxo da aplicação passa a ser trabalhado pelo MNA.

O MNA é responsável pelo cálculo da trajetória tomada pelo agente, e consequentemente pela tomada de decisão que ocorre a cada iteração do algoritmo de busca. Este módulo também gerencia todas as funções e estruturas responsáveis pela movimentação do robô dentro do ambiente. O percurso é obtido através de um algoritmo de busca guloso, utilizando como auxílio três diferentes funções heurísticas ([5]-[7]). Ao final da execução, uma seqüência de instruções é elaborada e transmitida para o robô realizar a movimentação.

A implementação dos algoritmos foi feita através da linguagem *Python* ([8] e [9]). Foi utilizada a versão 2.5.4 do *Python*. Nas etapas que envolveram processamento visual utilizou-se o módulo *PIL* para manipulação de imagens ([10]), a versão utilizada é compatível com Python 2.5.4. Já o programa que executa as instruções do robô é descrito em *NXT Python* ([11]). As técnicas e processos utilizados ao longo do trabalho serão descritos a seguir.

Robô Utilizado

Para confecção do agente robótico utilizou-se o *kit* Lego Mindstorms. Inicialmente a idéia era utilizar o modelo RCX, mas após alguns problemas, tanto de sensores como de atuadores, optou-se pela versão mais nova do *kit*: o modelo NXT, lançado pela Lego no ano de 2006. Características adicionais de como manipular estes *kits* podem ser encontrados em [12] e [13]. A figura 1 nos identifica o agente utilizado.

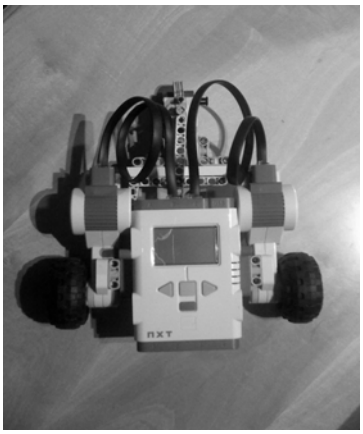

FIGURA. 1
ROBÔ UTILIZADO NO EXPERIMENTO.

Processamento Visual

Na primeira etapa do algoritmo de processamento visual, captura-se uma imagem panorâmica do cenário, constituído por um robô (de cor cinza), o alvo (na cor azul), e alguns obstáculos na cor preta dispostos em um piso de cor laranja. Para a captura da imagem seguiu-se às instruções descritas em [5].

A partir das imagens capturadas, o MPV deve identificar, usando coordenadas cartesianas, a posição do robô e o local do destino. No início do processamento no MPV uma imagem colorida é armazenada, com dimensões de 320x240 ([5]). A partir desta imagem o algoritmo funciona em duas etapas:

- **Localização do Robô e Posição Objetivo:** é realizada uma varredura pela imagem tentando identificar pixels que compõem a região do agente e do objetivo. Estas coordenadas são identificadas e armazenadas para cálculo do centro de massa da região. Com as coordenadas cartesianas identificadas na imagem (através do centro de massa), o procedimento as converte para uma coordenada matricial com o respectivo valor de dígito (discutido nas seções a seguir);
- **Localização de Obstáculos:** a imagem do ambiente é segmentada, aplicando-se processos para a conversão em níveis de cinza, realiza-se uma filtragem gaussiana da imagem seguida de uma etapa de limiarização. O objetivo aqui é identificar as áreas que compõem obstáculos na imagem. Estes processos (filtragem, limiarização, etc.) auxiliam na redução de ruídos existentes na cena capturada ([5]). Abordagens convencionais para segmentação de imagens são normalmente baseadas nas propriedades básicas dos níveis de cinza da imagem, buscando detectar *descontinuidades* ou *similaridades* na imagem ([14]). Já a existência de ruídos pode levar os métodos a distorcer as formas dos objetos, comprometendo seu reconhecimento ([14]). Conseqüentemente, obtém-se

uma imagem auxiliar contendo os obstáculos segmentados. Uma vez identificado se um pixel é pertencente a um obstáculo, ele é expandido recursivamente através da vizinhança-de-4, marcando cada pixel já percorrido com uma determinada tonalidade de cinza, de maneira que o algoritmo não visite dentro das recursões um pixel já percorrido. Além disso, é armazenado o número de pixels componentes de uma região segmentada. Regiões com uma determinada quantidades de pixels são identificadas como obstáculos, enquanto as demais são consideradas como ruído. Ao final da recursão, as dimensões máximas do obstáculo são identificadas e convertidas para a representação matricial (discutida a seguir).

Após esse processamento, uma representação matricial do ambiente é construída.

Ambiente

Após a etapa de processamento visual pelo MPV, o ambiente é convertido em uma representação em malha, com seis linhas e sete colunas (*matriz de ordem 6x7*). Estático, completamente observável e de um único agente, o ambiente é de fácil manipulação e tratamento de eventos ([7]). Dentro do ambiente podem existir obstáculos, representando possíveis "interferências naturais" ao deslocamento do agente. Sendo assim, a configuração inicial do problema é composta por:

- **Estado Inicial:** posição inicial do agente dentro do ambiente;
- **Estado Final:** posição final (ou objetivo) do agente;
- **Obstáculos:** que representam as dificuldades naturais ao movimento do agente.

A figura 2 apresenta um exemplo de configuração de ambiente capturada pelo MPV e sua respectiva representação matricial.

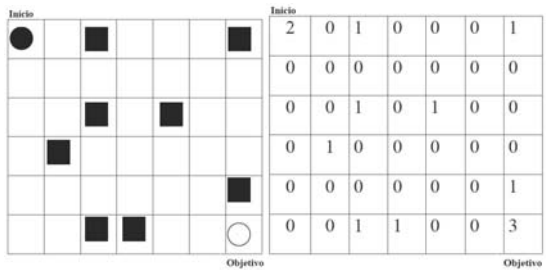

FIGURA. 2
AMBIENTE COM OBSTÁCULOS DISPOSTOS E MATRIZ DIGITALIZADA CORRESPONDENTE.

Cada dígito existente na matriz digitalizada da figura 2 apresenta um significado próprio:

- **Dígito 0:** representa coordenada livre para o ação do agente;
- **Dígito 1:** representa coordenada onde há um obstáculo situado;
- **Dígito 2:** representa a coordenada onde o agente se encontra inicialmente;
- **Dígito 3:** representa a coordenada do objetivo final do agente.

Com base em tais critérios de representação do ambiente, podemos então calcular a trajetória.

Cálculo da Trajetória

Para o cálculo da trajetória do agente foi utilizado um algoritmo de busca baseado em informação. A abordagem utilizada é o que se chama de **busca pela melhor escolha,** que consiste do processo onde um nó é selecionado para expansão com base em uma **função de avaliação** *f(n)*. Desta maneira o nó com a avaliação *mais baixa* é selecionado para expansão, porque a avaliação mede a distância até o objetivo ([7]). A busca gulosa tenta expandir o nó mais próximo à meta, na suposição de que isso provavelmente levará a uma solução rápida. O algoritmo guloso usado, embora não seja ótimo, é adequado para o problema visto que combina vantagens das buscas em amplitude e profundidade e que o custo de cada passo do robô é unitário ([7]).

A movimentação do robô pode ser feita nas direções 0º, 90º, 180º e 270º como pode ser observado na figura 3. Diante da facilidade de representação do ambiente foram utilizadas para *f(n)*, três diferentes medidas de distância conhecidas como: *City-Block* (CB), *Chessboard* (CS) e *Distância Euclidiana* (DE), representadas pelas seguintes equações:

$$\begin{aligned} CB(p,q) &= |x-u| + |y-v| \\ CS(p,q) &= \max(|x-u|,|y-v|) \\ DE(p,q) &= \sqrt{(x-u)^2 + (y-v)^2} \end{aligned} \qquad (1)$$

onde *p(x,y)* e *q(u,v)* denotam coordenadas do plano cartesiano, estados que compõem o ambiente ([5]).

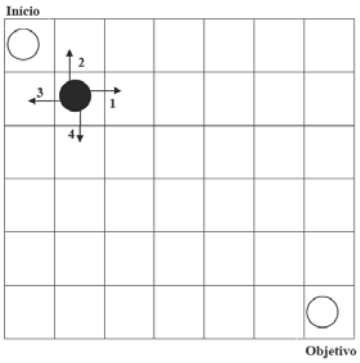

FIGURA. 3
ESQUEMA DE MOVIMENTAÇÃO DO AGENTE

Acionamento do Robô

Após etapa de processamento visual da imagem, e cálculo da rota de acordo com as heurísticas propostas, as coordenadas deste roteamento são transformadas em pseudocódigo e enviadas ao robô por intermédio de um transmissor sem fio. Tais dados são obtidos pelo agente robótico e então o mesmo executa a seqüência de instruções prevista na solução encontrada.

RESULTADOS E DISCUSSÕES

Durante as experimentações efetuamos análises de duas naturezas: 1 – ambientes com soluções, onde existiam rotas até a posição objetivo; e 2 – ambientes sem solução, ou seja, quando qualquer rota até a posição final é obstruída pela existência de obstáculos. Em um primeiro momento fixamos as posições de origem e destino, para analisar as trajetórias obtidas com as diferentes funções já citadas acima. Uma amostra deste comportamento observado está representada graficamente nas figuras 4 e 5.

A figura 4 mostra um ambiente *default*, ou seja, um ambiente que não contém obstáculos, explorado aqui no intuito de captar e perceber o comportamento natural das funções conforme a aproximação ao estado objetivo.

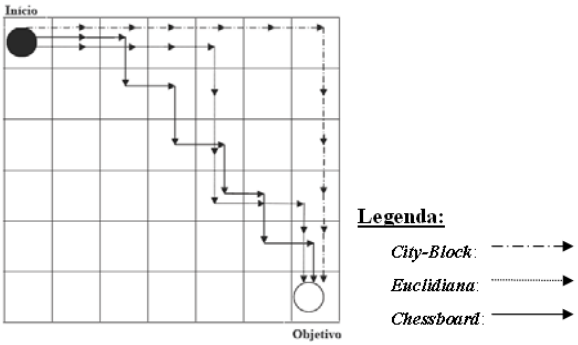

FIGURA. 4
COMPORTAMENTO DA TRAJETÓRIA DO AGENTE EM UM AMBIENTE SEM OBSTÁCULOS

Já na figura 5, podemos visualizar que as funções descritas adotam escolhas distintas de caminho conforme a própria análise do problema local, isto a cada iteração do algoritmo de busca. Às vezes, tais escolhas podem coincidir para certos tipos de funções, em outros casos, não.

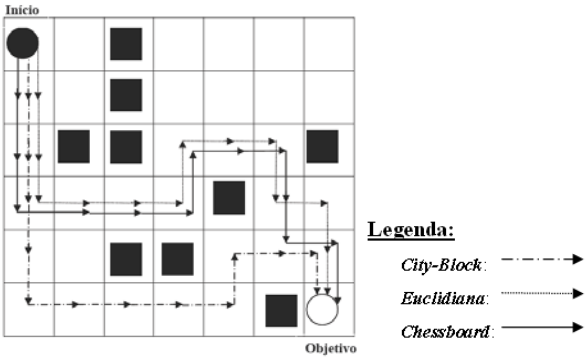

FIGURA. 5
COMPORTAMENTO DA TRAJETÓRIA DO AGENTE EM UM AMBIENTE COM OBSTÁCULOS DISPERSOS

Em um segundo momento, os testes passaram a adotar posições inicial e final variáveis dentro do ambiente, e não mais fixa e pré-determinada como nos exemplos anteriores. Uma análise de tal natureza pode ser vista na figura 6.

Além da comparação visual, podemos explicitar os resultados obtidos do desempenho de cada função por meio de tabelas, denotando o desempenho obtido para um número fixo de testes. Cada linha da tabela I contém o número do teste realizado (um identificador), a quantidade de obstáculos inseridos no ambiente, e o número de posições percorridas para cada função heurística aplicada: *City-Block, Chessboard e Distância Euclidiana*, respectivamente.

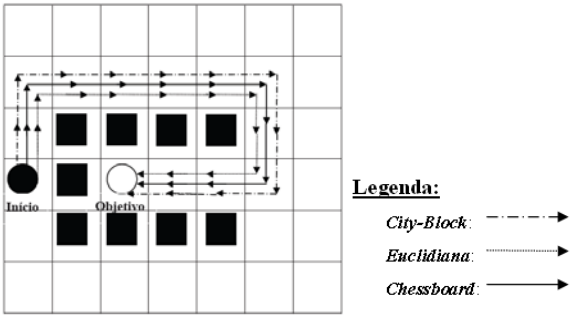

FIGURA. 6
COMPORTAMENTO DA TRAJETÓRIA DO AGENTE EM UM AMBIENTE COM OBSTÁCULOS DISPERSOS E POSIÇÕES DE ORIGEM E DESTINO NÃO-FIXAS

De acordo com tais valores podemos observar que a distância *Chessboard* perde em desempenho quando comparada com as demais, já que percorre um número maior de posições no percurso entre o estado inicial e o estado final. Já as funções *City-Block* e *Distância Euclidiana* apresentaram resultados semelhantes.

TABELA I
COMPARATIVOS ENTRE AS SOLUÇÕES ENCONTRADAS USANDO AS 3 FUNÇÕES DE HEURÍSTICA.

Nº Teste	Nº. Obstáculos	CB	CS	DE
1	0	11	11	11
2	5	11	11	11
3	6	11	11	11
4[1]	3	0	0	0
5	6	11	11	11
6	11	14	11	11
7[1]	6	0	0	0
8[1]	7	0	0	0
9	19	19	19	19
10	9	16	15	15
	Custo Médio	9.3	8.9	8.9

[1] – são ambientes onde o objetivo encontra-se cercado por obstáculos, ou seja, não há uma trajetória qualquer que chegue à posição final desejada.

Outro fator a ser destacado é que embora os custos obtidos sejam quase os mesmos, as trajetórias obtidas, em sua maioria, divergem muito quanto à semelhança de trajeto, diferença explicada pela natureza de cada função. Além disso, durante as experimentações constatou-se que o tempo de execução de uma instrução de mudança de direção para direita/esquerda (*turn left*/*right*) é maior do que o tempo de execução de uma instrução para avançar à frente (*forward*), quando utilizados os *kits* de robótica Lego Mindstorms. Sendo assim, embora os resultados mostrem que a quantidade de posições percorridas para cada função é próxima, o desempenho real é influenciado diretamente pelo tipo de trajetória obtida. Desta premissa, e avaliando o desempenho em relação ao tempo real de execução observamos uma ligeira vantagem para a função de Distância Euclidiana, já que ela proporcionou trajetórias mais "uniformes", sem muitas instruções de mudança de direção, resultando em rotas mais "eficientes".

CONCLUSÃO

O presente trabalho, além de ferramenta para o aprendizado e desenvolvimento de atividades ligadas à tecnologia, serviu como trabalho introdutório a diversas disciplinas da grade curricular do curso de Ciências da Computação. Explorando processamento gráfico, técnicas de inteligência artificial e afins, o experimento foi um fator agregador multidisciplinar. Além disso, foi desenvolvida uma aplicação de navegação autônoma utilizando os *kits* de robótica Lego Mindstorms. O algoritmo de roteamento quando utilizando da função heurística da *Distância Euclidiana* (DE) convergiu para soluções mais satisfatórias do que quando usadas as funções de distância *Chessboard* (CS) e *City-Block* (CB). Tal fato se deve principalmente à natureza do trajeto, que contém comportamentos mais "regulares", evitando mudanças de orientação desnecessárias, e reduzindo eventuais atrasos de execução das instruções. Possíveis trabalhos futuros poderiam ser elaborados visando explorar melhor as características de um ambiente mais complexo. Outro fator a ser pensado é melhorar a parte de segmentação e representação do ambiente, além de otimizar o algoritmo de elaborado, avaliando outras possíveis maneiras de roteamento, visando diminuir problemas em relação às condições do ambiente.

REFERÊNCIAS

[1] Siegwart, R.; Nourbakhsh, I. R. "Introduction to Autonomous Mobile Robots", *MIT Press*, 2004.

[2] Araújo, S. A.; Librantz, A. F. H., "Navegação Autônoma de Robôs", *Exacta*, São Paulo, v. 4, No especial, 2006, PP 81-83.

[3] Cazangi, R. R., "Uma proposta evolutiva para Controle Inteligente em Navegação Autônoma de Robôs", *Dissertação Mestrado*, Universidade Estadual de Campinas – Faculdade de Engenharia Elétrica e Computação, 2004.

[4] Pfeifer, R; Sheier, C., "Understanding Intelligence", *MIT Press*, 1999.

[5] Araújo, S. A; Librantz, A. F. H.; Flório Filho, O., "Navegação Autônoma de Robôs: uma implementação utilizando o kit Lego Mindstorms", In: *Congresso Sul Catarinense de Computação*, Criciúma: SulComp, 2006.

[6] Librantz, A. F. H., Araújo, S. A., Koyama, C., "Sistema de Navegação Autônoma de Robôs: uma proposta de uso como instrumento pedagógico multidisciplinar no curso de Ciência da Computação", *Exacta*, São Paulo, v.4, No especial, PP. 123-125, 2006.

[7] Russel, S; Norvig, P., "Inteligência Artificial", *Campus*, Rio de Janeiro, 2004.

[8] Python. "Python Programming Language: Official Website". Disponível em: http://www.python.org/. Acesso em 25 de março de 2009.

[9] Lutz M., "Pyhton: Guia de Bolso", *Atla Books Ltda*, Rio de Janeiro, 2006.

[10] NXTPython."NXT Python". Disponível em: http://home.comcast.net/~dplau/nxt_python/. Acesso em 04 de abril de 2009.

[11] PIL. "Python Imaging Library". Disponível em: http://www.pythonware.com/products/pil/. Acesso em 25 de março de 2009.

[12] Sartori, B., "Estudo da Linguagem Legolog", *Trabalho de Conclusão de Curso*, Univesidade Estadual de Londrina – Departamento de Computação, 2005.

[13] LEGO. "Lego.com: Mindstorms NXT Home". Disponível em http://mindstorms.lego.com/. Acesso em 04 de abril de 2009.

[14] Pedrini, H.; Schwartz, W. R. "Análise de Imagens Digitais: Princípios e Aplicações", *Thomson Learning*, São Paulo, 2008.