Aprendendo por reforço a evitar colisões em um robô móvel

Leandro Toss Hoffmann[1] e José Demisio S. da Silva[2]

Programa de Pós-graduação em Computação Aplicada (CAP)

Instituto Nacional de Pesquisas Espaciais (INPE)

[1]*hoffmann@dss.inpe.br* [2]*demisio@lac.inpe.br*

Resumo

Este trabalho apresenta a modelagem de um robô móvel sobre a óptica de um agente de aprendizagem, com o objetivo de navegar em um ambiente desconhecido, desviando de obstáculos. A aprendizagem da tarefa de navegação, bem como o processo de tomada de decisão, é realizada através do algoritmo Q-learning de aprendizagem por reforço. O sistema sensorial do agente é baseado em visão computacional, detectando informações sobre os objetos do ambiente, que serão utilizadas como variáveis de estado. Experimentos foram realizados com um robô móvel real, equipado com sensores e atuadores de baixo custo e de baixa precisão. Os resultados mostraram que o robô foi capaz de aprender a navegar num ambiente não estruturado, adaptando-se a imprecisões de modelagem, otimizando continuamente a escolha de suas ações.

Palavras-chave: robótica móvel, aprendizado por reforço, algoritmo *Q-learning*.

1. Introdução

A navegação autônoma está relacionada à habilidade de um veículo mover-se sem intervenção humana até alcançar o seu objetivo, em um ambiente no qual nenhuma informação, em princípio, está disponível. Em geral, para realizar essa tarefa, encontra-se uma função de mapeamento que recebe como entrada os dados dos sensores e produz comandos que serão enviados aos atuadores do veículo, podendo inclusive ser abordado como um problema inverso (Surmann et al. (1995)).

Mesmo com o aumento do poder computacional dos processadores, dependendo da complexidade do problema, torna-se impraticável o uso de modelos matemáticos precisos. Uma das alternativas então é o uso de técnicas emergentes, como computação inteligente (*soft computing*), que requerem menor poder computacional para processamento (Yager and Zadeth (1994)). Em especial, técnicas de *aprendizagem por reforço*(RL - do inglês *Reinforcement Learning* - Sutton and Barto (1998)) tem atraído a atenção dos pesquisadores em aplicações de navegação e controle de robôs. Em RL, um agente é estimulado ou inibido a realizar determinadas ações, a medida que interage com o ambiente e recebe recompensas. Após um determinado número de iterações, o agente aprende a otimizar suas ações, sendo então particularmente útil em aplicações nas quais não se tem um modelo preciso do ambiente, e deseja-se realizar um aprendizado *on-line* (Singh et al. (1997)).

Vários trabalhos na literatura têm aplicado aprendizado por reforço na área de robótica e navegação. Nos trabalhos de Gaskett et al. (2000), Smart and Kaelbling (2001) eZhu and Levinson (2001) é feito o uso de informações provenientes de sensores de visão computacional, incorporados a algoritmos de RL para controlar a navegação de robôs móveis. Aranibar and Alsina (2004) aplicam o algoritmo *Q-learning* no planejamento de rotas de robôs autônomos. Bir Bhanu Pat Leang and Patterson (2001) apresentam uma implementação física do ambiente de labirintos, descritos no clássico trabalho de Sutton and Barto (1998). Yen and Hickey (2002) propõem uma arquitetura de aprendizagem de reforço, que incorpora um mecanismo de esquecimento, permitindo assim exploração, e um aprendizado baseado em características do ambiente, para mapeamento dos estados. Michels et al. (2005) usam RL com sensores de visão computacional para determinar a direção de um robô navegando em alta

velocidade. Martínez-Marín and Duckett (2005) apresentam um robô que aprende a tarefa de ancoragem (*docking*), também baseado em visão computacional.

Um dos algoritmos mais populares em aplicações de RL é o *Q-learning* de Watkin, que já vem sendo utilizado de forma híbrida. Faria and Romero (2000) propõem uma modificação no algoritmo de aprendizagem por reforço *R-learning*, através da inclusão de informações nebulosas, provenientes dos sensores, no cálculo de atualização dos valores-R. O uso de redes neurais e RL pode ser visto no trabalho de Macek et al. (2002) para tratar o problema de representação estrutural de estados contínuos, em aplicação de desvio de obstáculos em robôs móveis.

Neste trabalho, um agente de aprendizagem é modelado para interagir com um ambiente não estruturado, aprendendo a navegar de forma segura, evitando colisões. O robô móvel é equipado com sensores e atuadores de baixa precisão e treinamento é realizado num ambiente físico real. Para tanto, utilizou-se uma abordagem de aprendizado por reforço, descrita na Secção 2. Em seguida, a modelagem da arquitetura de agente é apresentada na Secção 3. Posteriormente, a Seção 4 traz alguns resultados de experimentos realizados; e finalmente a Seção 5 conclui o trabalho e discute trabalhos futuros.

2. Aprendizado por reforço

Um sistema de *aprendizagem por reforço* é um modelo de aprendizagem não supervisionado, em que um agente aprende a tomar suas decisões, sem que haja a presença de um tutor externo. Inicialmente, não se sabe qual a melhor ação a ser tomada a cada instante de tempo, mas por ser um sistema de aprendizado orientado a objetivos, o agente vai tomando consciência de que ações deve tomar para atingir os objetivos.

A cada ação tomada, o agente recebe uma *recompensa*, que pode ser positiva, na forma de um prêmio, ou negativa, na forma de uma punição. Muitas vezes, essa recompensa só é dada quando se atinge um certo *estado*, após a execução de várias ações. Esse atraso na entrega da recompensa, conhecido como recompensa com atraso (*delayed reward*), é muito comum em modelagens de problemas reais, e dificulta a percepção de quais ações no passado levaram a uma situação ruim ou boa. Um exemplo disto é o jogo da velha, no qual o agente escolhe as posições livres para posicionar sua peça, e no final do jogo recebe um prêmio se ganhou, ou uma punição de perdeu.

Assim, em um sistema de RL, o aprendizado é realizado gradativamente a medida que o agente interage com o ambiente, e melhora a escolha de suas ações.

Formalmente, a cada instante de tempo t, o agente percebe o ambiente através de seus sensores e identifica um estado s, pertencente a um conjunto de estados S. Para cada estado s existe um conjunto finito de ações $a \in A(s)$ que poderão ser tomadas. A escolha das ações é feita por meio de uma política π que representará a probabilidade de escolha de uma ação, num dado estado: $\pi = (a, s)$.

Para cada ação tomada, uma recompensa r poderá ser fornecida, na forma de um sinal de reforço. Utilizando o sinal de reforço, o problema de aprendizado consiste em encontrar uma política ótima π^* que maximize o recebimento de recompensas.

Dada uma política π é possível determinar uma função de valor $V^\pi(s)$, que mapeia um estado $s \in S$ num valor que representa a expectativa de recebimento de recompensas. Essa esperança de recompensas é expressa pela Equação 1, onde γ entre $[0, 1]$ é um fator de desconto para as recompensas que serão recebidas no futuro. Quando γ é 0, apenas o reforço imediato é considerado; e quando é 1, todas as recompensas futuras terão o mesmo peso do que a recompensa imediata. O fator de desconto é especialmente importante em modelagens em que o horizonte é infinito, ou seja, não existem estados terminais. Nestes casos, γ deve ser menor que um, caso contrário a esperança do valor do estado tenderá a infinito.

$$V^\pi(s) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s\right] \tag{1}$$

Existe então um relacionamento direto entre o valor de um estado e o valor dos estados vizinhos s', podendo o valor de um estado ser determinado pela *Equação de Bellman* (Equação 2).

$$V(s) = r + \gamma \max_{s'} (V(s')) \tag{2}$$

Se existir um modelo de transição de estados $T(s, a, s')$, que define a probabilidade de mudar para o estado s', tomando a ação a a partir do estado s, estão a Equação de Bellman pode ser expressa pela Equação 3.

$$V(s) = r + \gamma \max_{a} \sum_{s'} T(s, a, s') V(s') \tag{3}$$

Da mesma forma que conhecendo-se a política ótima π^* pode-se determinar os valores dos estados, também é possível efetuar o inverso: sabendo-se o valor ótimo dos estados V^* é possível determinar a política ótima, pois será possível escolher a melhor ação a ser tomar em cada estado (Equação 4). Trata-se então de um problema dual de otimização (ten Hagen and Kröse (1997)).

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s')V(s') \quad (4)$$

Vários algoritmos foram desenvolvidos a fim de se aproximar os valores de π^* e V^*. Os mais clássicos são o algoritmo de Iteração de Valor e Iteração de Política (descritos em Sutton and Barto (1998)), todavia esses métodos necessitam de um modelo apropriado do ambiente. Sendo assim, neste trabalho optou-se pelo *Algoritmo Q-learning*, que é baseado no método por *Diferença Temporal*, descritos a seguir.

2.1. Diferença Temporal

Na aprendizagem por Diferença Temporal (TD - *Temporal Difference*), dada uma política π várias iterações completas com o ambiente são realizadas (episódios), experimentando-se vários caminho pelo espaço de estados. Este método não precisa de testar várias ações viáveis a partir de um estado, mas apenas atualizar os valores dos estados, referente a ação tomada durante o episódio (ten Hagen and Kröse (1997); Russell and Norvig (2004)). Essa idéia é descrita pela Equação (5):

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\left(r + \gamma V_t(s_{t+1}) - V_t(s_t)\right) \quad (5)$$

onde $\alpha > 0$ é a taxa de aprendizado. Uma taxa de aprendizado muito grande, dificulta a aproximação dos valores, pois ficarão oscilando. Por outro lado, uma taxa de aprendizado muito pequena, acarretará uma demora na convergência do aprendizado.

2.2. Algoritmo *Q-learning*

O algoritmo *Q-learning* é uma técnica popularmente utilizada em aprendizagem por reforço, combinada com o método de aprendizagem TD. A vantagem do *Q-learning* é realizar apenas um mapeamento do par estados/ações para Q-valores, substituindo a função de valores e a função de transição de estados. Esse mapeamento é conhecido como Q-função, que define um Q-valores para cada ação possível a partir de um dado estado (ten Hagen and Kröse (1997); Harmon and Harmon (1996); Russell and Norvig (2004)).

Assim, um Q-valor é definido como sendo a soma das recompensas recebidas ao realizar uma dada ação, seguindo uma política fornecida. Ao assumir que o valor de um estado é dado pelo maior Q-valor desse estado, pode-se dizer que um Q-valor é definido pela Equação (6):

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\Delta q \quad (6)$$

onde Δq é dado pela Equação (7).

$$\Delta q = r + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \quad (7)$$

Uma das principais vantagens do *Q-learning* é viabilizar o aprendizado *online* do agente, embora a convergência ótima da escolha das ações possa ser lenta, dependendo da modelagem utilizada (Singh et al. (1997)).

3. Arquitetura do agente

O robô utilizado neste trabalho foi modelado sobre a óptica de um agente de aprendizagem, ou seja, dotado com um tipo especial de programa de agente, com capacidade de aprender durante sua iteração com o ambiente (Russell and Norvig (2004)). A Figura 1 ilustra a arquitetura interna do agente, dividida em quatro elementos: **elemento de aprendizado**, responsável pela adaptação do agente, através do aperfeiçoamento de suas ações; **elemento de desempenho** que realiza a tomada de decisão, definindo as ações que serão tomadas pelo agente; **crítico** avalia as ações do agente e determina como o *elemento de desempenho* deverá ser modificado; e **gerador de problemas** que cria novas situações para que o agente aprenda novas experiências. Percebe-se na Figura 1, que sua interação com o mundo externo é realizada através dos seus sensores e atuadores, mas também pode receber informações realimentadas através do *crítico*.

Neste trabalho, o agente de aprendizagem é implementado na forma de um robô móvel, composto pelos seguintes sub-sistemas:

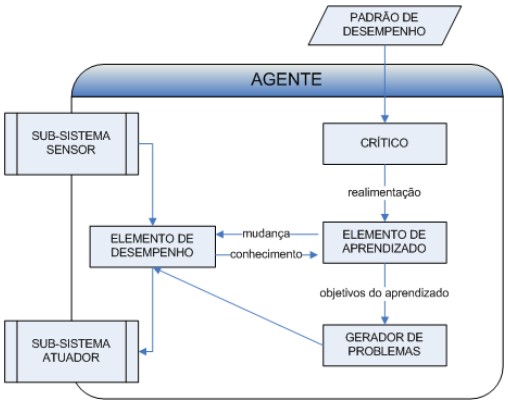

Figura 1: Arquitetura do agente de aprendizagem.

- **Sub-sistema sensor**: extrai as informações do ambiente, através de uma câmera de captura de imagens. Operadores de visão computacional baseados em Redes Neurais detectam objetos no ambiente, dos quais são calculadas posições e distâncias aproximadas. (Informações sobre Redes Neurais podem ser vista em Haykin (2001).)As posições são representadas em valores discretos *à esquerda*, *ao centro* e *à direita*. De forma semelhante, as distâncias são limiarizadas em *longe*, *médio* e *perto*. Essas informações, adicionadas à detecção de colisão, mapeiam o estado do agente, que é enviado ao sub-sistema de aprendizagem.

- **Sub-sistema de aprendizagem**: é implementado na forma do algoritmo *Q-learning*, que tem como objetivo navegar o robô, evitando objetos pelo caminho. O agente pode decidir entre 3 ações a serem tomadas (*andar pra frente*, *girar à esquerda* e *girar à direita*), que são enviadas ao sub-sistema atuador. A cada ação *à frente* tomada, é enviado um sinal de reforço $+1$. Quando o robô colidir com algum obstáculo, sofre uma punição na forma de um sinal -1.

- **Sub-sistema atuador**: converte as ações escolhidas pelo agente em sinais que serão enviados aos motores do robô, via rádio-freqüência.

4. Experimentos e Resultados

A Figura 2-a mostra o robô utilizado nos experimentos com a câmera para captura de imagens posicionada sobre o seu capô dianteiro. O ambiente no qual o robô deve navegar (ilustrado na Figura 2-b) consiste num terreno plano, não estruturado, com dimensões de 310 cm por 260 cm, no qual são posicionados objetos imóveis de vários tamanhos e formas. As extremidades do ambiente interno é delimitado com objetos planos coloridos.

Figura 2: (a) Robô móvel com câmera posicionada a frente; (b) Exemplo de ambiente com obstáculos na forma de objetos imóveis, por onde navega o robô.

Uma série de episódios de navegação foram realizados com o robô móvel, no ambiente real, em que o agente foi gradativamente otimizando suas ações, partindo do conhecimento zero. Cada episódio consistiu num determinado número de ações (passos) executadas pelo robô, totalizando 477 passos em todo o experimento. No último episódio, o agente efetuou 48 ações, no qual se utilizou os seguintes parâmetros do algoritmo *Q-learning*: $\alpha = 0,1$, $\gamma = 0,8$ e ϵ-*greedy* = 0.

A trajetória realizada pelo robô no quarto episódio está ilustrada na Figura 3, ao longo de 48 ações. Percebe-se que o agente pôde, na maioria das vezes, evitar as paredes e os objetos do ambiente, tomando o rumo a frente sempre que possível (ação esta estimulada). Por volta da ação número 40, o robô encontrou dificuldades em sair da região, pois ficou isolado entre uma parede e um objeto.

Das 48 ações tomadas, 7 levaram o robô a sofrer colisões, sendo que 6 foram no momento de manobras. Essas colisões ocorrem porque o atuador do robô não permite girar em torno do seu próprio eixo, necessitando de muito espaço para virar a esquerda ou a direita, aumentando assim as chances de bater em um obstáculo.

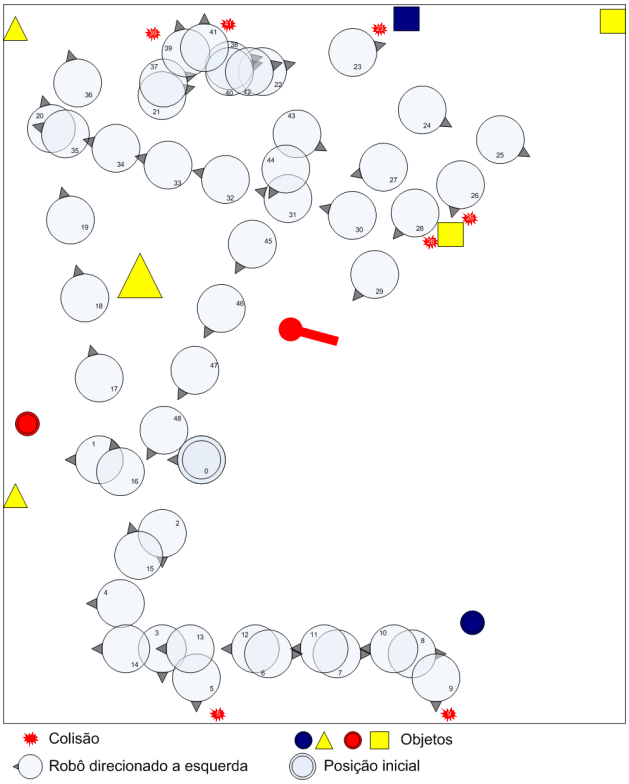

Figura 3: Descrição das trajetórias executadas pelo robô, num ambiente de 310 x 260 cm. A direção do robô é indicada por uma seta preta.

Uma colisão ocorreu devido a tomada de decisão incorreta, em uma situação que o agente não havia ainda experimentado (todos os Q-valores eram zero) (Figura 4). No instante número 22 (Figura 4-a) o agente percebe o estado 121 e sorteia a ação *frente*. Esta ação leva o robô a uma colisão (Figura 4-b). Mais tarde, no instante número 42 (Figura 5-a) o robô percebe novamente o estado 121, mas desta vez sabe que a ação frente não é a mais indicada (Q-valor$(121, frente) = -0.0468$), então escolhe a ação virar a *direita* (Figura 5-b).

A seqüência de ações ilustradas pelas Figuras 4 e 5 mostram a capacidade de aprendizado e adaptação do agente, frente a situações nunca enfrentadas.

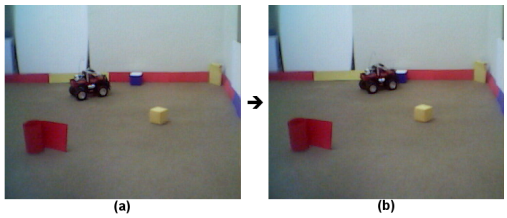

Figura 4: (a) O robô no instante 22 percebe o estado 121, até então desconhecido, e sorteia a ação *frente*, que o leva a colisão. (b) robô colide com o objeto

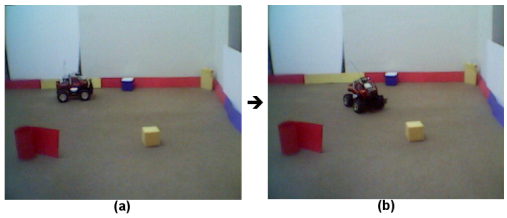

Figura 5: (a) O robô no instante 42 percebe o estado 121, que desta vez é conhecido, e então toma a ação *direita*. (b) robô evita a colisão, desviando do objeto a sua frente.

5. Conclusão

Neste trabalho foi apresentando um robô móvel que aprendeu a navegar num ambiente não estruturado, utilizando técnicas de aprendizado por reforço. Os experimentos realizados não foram exaustivos, cobrindo todo o espaço de estados, mas os resultados indicaram a capacidade de navegação do robô e o aprendizado contínuo de suas ações.

O fato de utilizar a política ϵ-*greedy*$= 0$ deixou o agente mais cauteloso, não experimentando ações novas e evitando ações que o levariam a punição já experimentadas. Isto foi particularmente interessante para viabilizar a navegação robô, com pouco tempo de treinamento, uma vez que se minimizava as chances do robô colidir gravemente com algum obstáculo. Por outro lado, essa política impediu o agente de otimizar ainda mais suas ações, pois não varreu suficientemente o espaço de busca. Esse é o conhecido dilema de "'*exploration x explotation*"', no qual se tem um compromisso entre a exploração de novas

ações em busca de otimização, em detrimento daquilo que já se aprendeu.

Uma solução para amenizar esse dilema seria organizar o sistema de aprendizagem em uma arquitetura hierárquica de navegação, proposta em Hoffmann and Silva (2005). Um módulo de baixo nível, cuidaria para que o robô não colidisse, sendo mais cauteloso com obstáculos próximos. Outro módulo aprenderia a navegar pelo ambiente, objetivando uma exploração homogênia do terreno.

Referências

Aranibar, D. B. and Alsina, P. J. (2004). Reinforcement learning-based path planning for autonomous robots. In *EnRI - XXIV Congresso da Sociedade Brasileira de Computação*, page 10, Salvador.

Bir Bhanu Pat Leang, Chris Cowden, Y. L. and Patterson, M. (2001). Real-time robot learning. In *IEEE International Conference on Robotics and Automation*.

Faria, G. and Romero, R. (2000). Incorporating fuzzy logic to reinforcement learning. In *Proc. IEEE Internat. Conf. on Fuzzy Systems*, pages 847–852.

Gaskett, C., Fletcher, L., and Zelinsky, A. (2000). Reinforcement learning for a vision based mobile robot. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Harmon, M. E. and Harmon, S. S. (1996). Reinforcement learning: a tutorial.

Haykin, S. (2001). *Redes neurais: princípios e prática*. Bookman, Porto Alegre.

Hoffmann, L. T. and Silva, J. D. S. (2005). Modelagem de um agente móvel de aprendizagem para vagueio em ambientes inexplorados. In *Anais do XXV Congresso da Sociedade Brasileira de Computação / V Encontro Nacional de Inteligência Artificial*, pages 882–891, São Leopoldo.

Macek, K., Petrovic, I., and Peric, N. (2002). A reinforcement learning approach to obstacle avoidance of mobile robots. In *International Workshop on Advanced Motion Control*, pages 462 – 466.

Martínez-Marín, T. and Duckett, T. (2005). Fast reinforcement learning for vision-guided mobile robots. In *Proc. ICRA-2005, IEEE International Conference on Robotics and Automation*, page 6, Barcelona.

Michels, J., Saxena, A., and Ng, A. Y. (2005). High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proc. XXII International Conference on Machine Learning*, page 8, Bonn.

Russell, S. and Norvig, P. (2004). *Inteligência Artificial*. Elsevier, Rio de Janeiro.

Singh, S., Norvig, P., and Cohn, D. (1997). How to make software agents do the right thing: An introduction to rl. In *Dr. Dobbs journal*.

Smart, W. D. and Kaelbling, L. P. (2001). Reinforcement learning for robot control. In *Mobile Robots XVI (Proc. SPIE 4573)*.

Surmann, H., Huser, J., and Peters, L. (1995). A fuzzy system for indoor mobile robot navigation. In *IV IEEE Internacional Conference on Fuzzy Systems*, pages 83–86, Yokohama.

Sutton, R. and Barto, A. (1998). *Reinforcement learning: an introduction*. MIT Press.

ten Hagen, S. and Kröse, B. (1997). A short introduction to reinforcement learning. In *Proc. of the 7th Belgian-Dutch Conf. on Machine Learning*, pages 7–12, Tilburg.

Yager, R. and Zadeth, L. A. (1994). *Fuzzy, Sets, Neural Networks, and Soft computing*. Van Nostrand Reinhold, New York.

Yen, G. and Hickey, T. (2002). Reinforcement learning algorithms for robotic navigation in dynamic environments. In *IJCNN International Joint Conference on Neural Networks*, volume 2, pages 1444 – 1449.

Zhu, W. and Levinson, S. (2001). Vision-based reinforcement learning for robot navigation. In *International Joint Conference on Neural Networks*, Washington DC.