Um sistema de visão robótica baseada em aprendizagem por reforço

Leandro Toss Hoffmann e José Demísio S. da Silva

Instituto Nacional de Pesquisas Espaciais – INPE
Programa de Pós-gradução em Computação Aplicada
Av. dos Astronautas, 1758 – 12227-010 – São José dos Campos, SP.
E-mail: {hoffmann, demisio}@lac.inpe.br

Resumo

Este trabalho apresenta estudos de aprendizagem por reforço, aplicados à robótica móvel. Neste tipo de aprendizagem, um robô tem seu comportamento avaliado e aprimorado, num dado ambiente, recebendo apenas prêmios ou punições a cada iteração. Para tanto, técnicas de aprendizagem por reforço são descritas, bem como o popular algoritmo Q-learning. O robô é modelado numa arquitetura de agente com aprendizagem, tendo como sensores, operadores de visão computacional. O desempenho do agente é avaliado num experimento simulado de visão ativa.

Palavras-chave: *aprendizagem de máquina, Q-learning, navegação autônoma*

1. Introdução

Em um ambiente de robótica, um dos grandes desafios consiste em programar o sistema de controle do robô, maximizando assim sua iteração com o ambiente. Técnicas de aprendizado são de grande valia neste ambiente, pois pode-se aproveitar dados de outros sistemas ou informações de especialistas para se desenvolver sistemas de controle mais flexíveis. Neste sentido, técnicas de aprendizagem por reforço levam ainda vantagem, pois dispensam o uso de uma base de dados supervisionada, uma vez que a avaliação do comportamento do agente é feita somente através de prêmios e punições. Assim, é possível para um agente aprender a tomar decisões, observando continuamente sua iteração com o ambiente.

A robótica móvel, num contexto de navegação autônoma, é um tema muito estudado e pesquisado na área de Inteligência Artificial [8] [12] [13] [14], da qual vários paradigmas vêm sendo utilizados para abordar o problema, inclusive aprendizagem por reforço [1] [10] [11]. 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.

Várias arquiteturas para navegação autônoma têm sido propostas e podem ser categorizadas basicamente como: hierárquicas, com um alto nível (considera modelos e planos); de baixo nível (considera detecções e execuções de ação); e arquiteturas híbridas que combinam duas ou mais técnicas [7]. Neste artigo, a modelagem de robôs móveis é feita pela óptica de agentes, que definem uma arquitetura própria e uma maneira de iteração com o ambiente. Mas especificamente, buscou-se uma arquitetura de agente onde se pudesse incorporar técnicas de aprendizagem por reforço.

Russell e Norvig [16] definem agentes com aprendizagem como aqueles que são dotados com um tipo especial de programa de agente, com capacidade de aprender durante sua iteração com o ambiente. Sua arquitetura interna pode ser 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. A Figura (1) ilustra a arquitetura geral de um agente com aprendizagem e o relacionamento dos componentes do seu programa. Percebe-se ainda 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*.

Para compor o sistema de sensores do agente, prima-se pela utilização de operadores de visão computacional, que são comumente encontrados na literatura de navegação autônoma [2] [4] [9] [13] [15] [17] [18] [19] [20] . Sensores de visão computacional tem um grande potencial de informações do ambiente e deve ser explorado. Todavia, é importante ressaltar, que o uso de operadores clássicos de processamento de imagens, podem levar ao aumento do custo com-

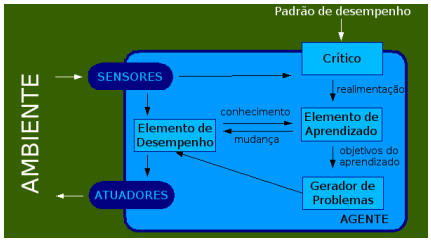

Figura 1. Arquitetura de um agente com aprendizagem e os componentes do seu programa. Fonte: adaptada de [16].

putacional do programa do agente, devendo-se assim buscar técnicas alternativas, menos custosas computacionalmente [3].

Buscando o desenvolvimento de um agente com aprendizagem, inicialmente, na secção 2, métodos de aprendizagem por reforço são apresentados. Posteriormente, na secção 3, o uso do algoritmo *Q-learning* é demonstrado através de experimentos simulados, numa aplicação de visão ativa. Em seguida, na seção 4, as conclusões são expostas.

2. Aprendizagem por reforço

Em um modelo de aprendizagem por reforço, um agente aprende a otimizar sua iteração com um ambiente por tentativa e erro. O ambiente pode ser parcialmente ou completamente observável, onde através de sensores, o agente identifica estados s. Em geral o agente pode modificar o ambiente, utilizando assim seus atuadores para efetuar ações a e mudar de estados [16].

Para cada ação tomada pelo agente, o mesmo pode receber uma recompensa, que será utilizada para avaliar o seu comportamente. Assim, uma *função de reforço* $r(s_t, a_t)$ é um mapeamento de estados/ações para recompensas, num instante de tempo t. A medida que se dá a iteração com o ambiente, o agente aprenderá a tomar decisões em determinadas situações que o levam a maximizar a soma de prêmios recebidos [16].

Para cada estado s_i então, existe um valor de utilidade associado à soma de todos prêmios recebidos durante uma época de iteração com o ambiente, iniciando em s_i, seguindo uma dada política, até alcançar um estado final. Desta forma, existe uma *função de valor* $V(s)$ que mapeia estados para valores de estados.

Uma política πdeterminará que ações deverão ser escolhidas em cada estado (*função de transição de estados*). Existe uma política ótima π^* e se conhecida, é possível encontrar uma função ótima de valores $V^*(s)$, e vice-versa. Durante o processo de aprendizagem, busca-se então aproximar π^* e $V^*(s)$ [5].

A dificuldade na aprendizagem por reforço surge quando um agente realiza uma ação e não é informado se essa ação foi boa ou não. Somente no final de um processo de aprendizagem lhe é fornecido um prêmio ou punição. Essas situações são conhecidas como recompensa atrasada (*delayed reward*), pois consequências futuras de uma ação não são incorporadas na sua recompensa, mas podem influenciar no desempenho global do agente.

Este problema configura uma tarefa de otimização e, em geral, são empregadas técnicas de programação dinâmica para se encontrar a função ótima de valores $V^*(s)$. Como ponto de partida, pode-se utilizar a equação de Bellman (Equação (1)), que define uma relação entre sucessivos estados. Essa relação parte do princípio que o valor de um estado $V^*(s)$, para uma política ótima π^*, é definido pela soma de todas recompensas, iniciando em s_i até alcançar um estado terminal [6]. O valor $\gamma \in [0, 1]$ é um fator de desconto usado para pesar o valor de reforço recebido no futuro.

$$V^*(s_t) = r + \gamma V^*(s_{t+1}) \tag{1}$$

Assumindo a Equação (1), a seguir serão descritos três métodos para se aproximar a função de valor de um estado.

2.1 Algoritmo de Iteração de Valor

O algoritmo de Iteração de Valor (*Value Iteration*) assume que a função de valor é aproximada por uma tabela (*lookup table*), ou seja, cada estado tem um elemento correspondente numa tabela. A busca pela função ótima de valor é feita através da atualização dos valores dos estados em sucessivas explorações do espaço de estados, até que os valores tenham convergido [6]. Cada valor é atualizado segundo a Equação (2).

$$V_{t+1}(s_t) = \max_{a_t} (r + \gamma V_t(s_{t+1})) \tag{2}$$

Observa-se que todas as ações a possíveis para um dado estado s devem ser testadas, o que só é viável na prática através de um modelo da dinâmica do sistema.

2.2 Aprendizagem por Diferença Temporal

Na aprendizagem por Diferença Temporal (TD - *Temporal Difference*), dada uma política π várias iterações completas com o ambiente são realizadas (método de Monte Carlo), realizando vários caminho pelo espaço de estados. Diferente do método de Iteração de Valor, este método não necessita testar várias ações viáveis a partir de um estado, apenas atualizar os valores dos estados, segundo a ação tomada

durante a iteração [5, 16]. Essa idéia é descrita pela Equação (3):

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\left(r + \gamma V_t(s_{t+1}) - V_t(s_t)\right) \quad (3)$$

onde $\alpha > 0$ é a taxa de aprendizado.

2.3 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 valores-Q, substituindo a função de valores e a função de transição de estados. Esse mapeamento é conhecido como função-Q, que define um valor-Q para cada ação possível num dado estado [5, 6, 16].

Assim, um Q-valor é definido como sendo a soma das recompensas recebidas ao realizar uma dada ação, seguindo uma política fornecida. Ao assumir então que o valor de um estado é definido pelo Q-valor máximo desse estado, pode-se dizer que o Q-valor é dado pela Equação (4):

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\Delta q \quad (4)$$

onde Δq é dado pela Equação (5).

$$\Delta q = r + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \quad (5)$$

3. Experimentos e resultados

Em sistemas de visão ativa, um agente é equipado com uma ou mais câmeras, onde é possível controlar o posicionamento dessa câmera ou do seu foco. Uma das vantagens desse tipo de visão computacional é permitir uma maior flexibilidade ao robô, pois o mesmo pode obter várias imagens da cena, sem a necessidade de mover-se pelo ambiente.

Em ambientes onde existem objetos móveis, um sistema de visão ativa permite, por exemplo, manter o foco de atenção sobre um dado objeto. A fim de se validar o uso de aprendizagem por reforço, neste trabalho é apresentado um experimento de foco de atenção num ambiente simulado. Para tanto, o ambiente em questão consiste num único objeto móvel, com características espectrais que o diferem da cena, por contraste. O objetivo do robô, neste sistema hipotético de visão, é aprender que ações deverão ser tomadas a fim de manter a projeção do objeto no centro da imagem.

O experimento é realizado em duas etapas: deteção do objeto alvo e atuação do agente. A primeira delas consiste no desenvolvimento do sensor, que visa identificar a localização aproximada do objeto alvo na imagem. A localização será utilizada pelo agente, na segunda etapa do processamento, que determinará as ações que o deverão ser tomadas para movimentar a câmera.

Assim, no módulo sensor, inicialmente a imagem colorida (em canais RGB) capturada da cena é reamostrada para uma imagem de 10 linhas e 10 colunas. Essa operação visa reduzir o espaço de estados e conseqüentemente a complexidade computacional do problema. A nova imagem reamostrada é submetida a uma operação de convolução, utilizando uma máscara de vizinhança 8, onde os níveis RGB de cada ponto da máscara são ativados numa rede neural, previamente treinada. A saída da rede neural trata-se da informação de presença ou não do objeto alvo, naquele ponto da imagem. Ao final da aplicação do filtro neural, espera-se identificar a localização aproximada do objeto na imagem reamostrada.

A rede neural artificial, utilizada no filtro, consiste numa rede de perceptrons de múltiplas camadas, com 27 unidades sensoriais, 10 unidades na camada oculta e 1 unidade de saída. Para o treinamento foi utilizando o algoritmo de retro-propagação do erro, com dados previamente coletados.

A Figura (2) mostra um caso particular de processamento realizado pelo módulo sensor. Nota-se na Figura (2-a) a imagem bruta capturada da cena, com o objeto alvo (um balão azul), posicionado na região superior direita. Após o processamento com filtro neural, a localização aproximada do objeto é destacada na Figura (2-b).

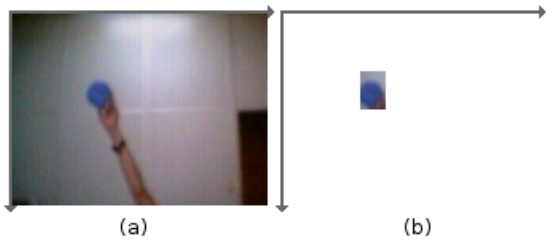

Figura 2. Identificação da localização aproximada do objeto alvo, pelo módulo sensor: (a) imagem original capturada; (b) imagem processada.

A localização na imagem do objeto alvo é recebida pelo agente como um dado sensorial, que será utilizado para identificar o estado atual do agente. A Figura (3) ilustra o mapeamento dos dados sensoriais do agente para os estados do ambiente. Nota-se que para cada coordenada inteira de localização do objeto, existe um estado associado. No centro da imagem, existem 4 estados terminais, para situações em que a imagem estiver centrada no objeto alvo. Além disto,

são definidos estados terminais adicionais de contorno da imagem, totalizando 144 estados. Na Figura (3) é possível perceber ainda as recompensas possíveis do agente, em função de cada estado: 1 para os estados terminais centrais, -1 para os estados terminais de contorno e -1/25 para os demais estados.

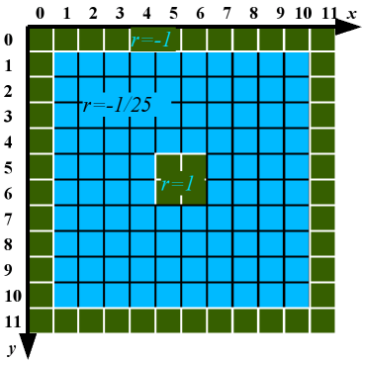

Figura 3. Possíveis estados do agente e as respectivas recompensas. Estados no contorno e centrais são terminais.

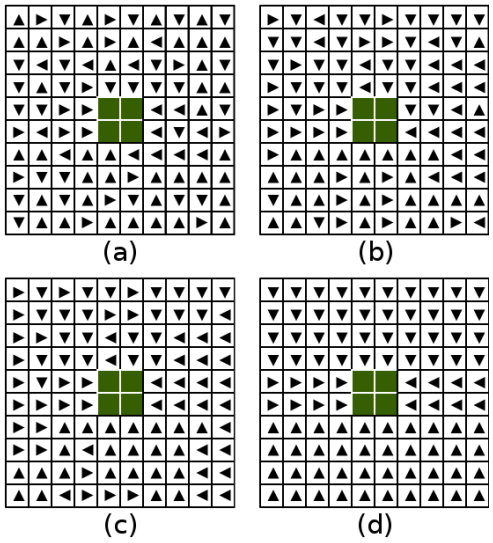

Figura 4. Política que mapeia as ações em cada estado, após (a) 40, (b) 120, (c) 160 e (d) 1560 iterações do algoritmo *Q-learning*

Em cada estado não terminal, o agente pode tomar 4 tipo de ações diferentes, acionando seus motores para mover a câmera para baixo, para cima, à esquerda e à direita. Assume-se que os movimentos realizados pelos atuadores sejam em espaços discretos e numa distância aparente próximo ao tamanho de cada ponto da imagem reamostrada.

A política utilizada pelo agente foi seleção aleatória das ações, com distribuição uniforme, durante um período limitado de aprendizagem. Em seguida, adotou-se uma política de escolha do maior $Q - valor$ (política *greedy*). A aprendizagem foi realizada com o algoritmo *Q-learning*, que utilizou os seguintes parâmetros: $\alpha = 0.9$ e o coeficiente $\gamma = 1$.

A Figura (4) ilustra a política que o agente deve seguir em cada estado, o que é uma reflexão direta dos valores de utilidade estimados para cada estado (o Q-valor mais alto de um estado define a ação a ser tomada). Inicialmente, os valore são inicializados em zero, mas logo com 40 iterações do algoritmo *Q-learning* (Figura (4-a)) percebe-se uma tendência na convergência dos valores, ficando essa tendência mais clara com 120 iterações (Figura (4-b)). Já com 160 iterações (Figura (4-c)) o agente já tem capacidade total de posicionar a câmera corretamente, a partir de qualquer estado inicial, ainda que as ações não sejam ótimas. Com 1560 iterações os Q-valores convergem, resultando em ações ótimas, como ilustrado na Figura (4-d).

Com este experimento, foi possível observar que o agente pôde aprender a tomar ações desejadas, utilizando-se sinais de reforço do ambiente. Fica claro que o mesmo comportamento do agente poderia ser obtido através de um simples algoritmo programado, todavia, neste experimento teve-se como objetivo avaliar o desempenho do algoritmo *Q-learning*, e por sua vez a capacidade de aprendizado do agente, extendendo-o para aplicações mais complexas.

4. Conclusões

Este trabalho apresentou algumas técnicas de aprendizagem por reforço, visando-se aplicações num contexto de robótica. Experimentos demonstraram o uso do algoritmo *Q-learning* aplicado ao problema de foco de atenção, em visão ativa. Observou-se que após um número pequeno de iterações com o ambiente, o agente foi capaz que aprender o comportamento ótimo de suas ações.

Todavia, ressalta-se que numa aplicação complexa de navegação autônoma, trabalhando-se num espaço contínuo de estados, as técnicas de aprendizagem deverão ser combinadas com aproximadores de funções, de modo a acelerar (ou até mesmo viabilizar) o processo de estimação dos valores de utilidade dos estados. Este tipo de aproximação é demonstrada, por exemplo, no trabalho de Tesauro [22] utilizando redes neurais, e Smart e Kaelbling [21], com o algoritmo de Hedger.

Os experimentos apresentados foram realizados sob condições específicas de iluminação, o que possibilitou o desenvolvimento de um sistema de visão satisfatório. Porém numa aplicação real, sistemas de visão devem ser robustos o suficiente para tolerar pe-

quenas mudanças de iluminação do ambiente, sendo assim esse módulo um fator crítico para identificação do estado do agente.

Por fim, cuidados de implementação visando processamento em tempo real, devem ser tomados, sobretudo em se tratando de ambientes com objetos móveis.

Referências

[1] Aranibar, D. B. & Alsina, P. J. "Reinforcement learning-based path planning for autonomous robots". In *I Encontro Nacional de Robótica Inteligente - SBC*. Salvador, pp. 1–10. 2004.

[2] Baluja, S. & Pomerleau, D. "Dynamic relevance: vision-based focus of attention using artificial neural networks". *Artificial Intelligence Magazine*, 1997, 97, 381–395.

[3] Bittencourt, J. R. & Osório, F. s. "Processamento de imagens inteligente usando redes neurais artificiais". In *Aprendizado, criação e integração na Iniciação Científica*, edited by M. A. Rocha & T. R. Cruz. Porto Alegre, 2002, pp. 11–35.

[4] Castro, A. P. A.; Silva, J. D. S. & Simoni, P. O. "Image based autonomous navigation fuzzy logic control". In *IJCNN International Joint Conference on Neural Networks*, vol. 1. Washington, pp. 2200–2205. 2001.

[5] ten Hagen, S. & Kröse, B. "A short introduction to reinforcement learning". In *Proc. of the 7th Belgian-Dutch Conf. on Machine Learning*. Tilburg, pp. 7–12. 1997. URL `citeseer.ist.psu.edu/tenhagen97short.html`.

[6] Harmon, M. E. & Harmon, S. S. "Reinforcement learning: a tutorial", 1996. URL `citeseer.ist.psu.edu/harmon96reinforcement.html`.

[7] Heinen, F. J. *Sistema de controle híbrido para robôs móveis autônomos*. Master's thesis, Universidade do Vale do Rio dos Sinos, São Leopoldo, 2001.

[8] Hoffmann, L. T.; Castro, A. P. A. & Silva, J. D. S. "Controle inteligente de um robô móvel utilizando imagens". In *I Encontro Nacional de Robótica Inteligente da SBC*. Salvador, pp. 1–10. 2004.

[9] Jochem, T. & Pomerleau, D. "Life in the fast lane". *Artificial Intelligence Magazine*, 1996, 17(2), 11–50.

[10] Kimura, H. & Kobayashi, S. "Reinforcement learning for locomotion of a two-linked robot arm". In *Proceedings of the 6th Europian Workshop on Learning Robots*. pp. 144–153. 1997.

[11] Kimura, H.; Miyazaki, K. & Kobayashi, S. "Reinforcement learning in pomdps with function approximation". In *Proceedings of the 14th International Conference on Machine Learning*. pp. 152–160. 1997.

[12] Kosaka, A. & Pan, J. "Purdue experiments in model-based vision for hallway navigation". In *Proceedings of Workshop on Vision for Robots in IROS'95*. pp. 87–96. 1995.

[13] Li, W.; Lu, G. & Wang, Y. "Recognizing white line markings for vision guided vehicle navigation". *Fuzzy Reasoning - PRL*, 1997, 18(8), 771–780.

[14] Li, W.; Xu, C.; Xiao, Q. & Xu, X. "Visual navigation of an autonomous robot using white line recognition". In *IEEE International Conference on Robotics and Automation - ICRA*. Taipet, Taiwan, pp. 3923–3928. 2003.

[15] Quiles, M. G. & Romero, R. A. F. "Controle de um robô móvel através do reconhecimento de cores e formas". In *I Encontro Nacional de Robótica Inteligente - SBC*. Salvador, pp. 1–10. 2004.

[16] Russell, S. & Norvig, P. *Inteligência Artificial*. Elsevier, Rio de Janeiro, 2004.

[17] Sargent, R.; Bailey, B.; Witty, C. & Wright, A. "Dynamic object capture using fast vision tracking". *Artificial Intelligence Magazine*, 1997, 18(1), 65–72.

[18] Shah, S. & Aggarwal, J. K. "Mobile robot navigation and scene modeling using stereo fish-eye lens system". *Machine Vision and Applications*, 1996, 10(4), 159–173. URL `citeseer.ist.psu.edu/shah97mobile.html`.

[19] Silva, J. D. S. & Simoni, P. O. "The correspondence problem: an uncertainty reasoning approach". In *IASTED International Conference on Artificial Intelligence and Applications*. Mágala, pp. 210–215. 2002.

[20] Silva, J. D. S. & Simoni, P. O. "Dempster-shafer theory as an inference method for corresponding geometric distorted images". In *IV Encontro Nacional de Inteligência Artificial (ENIA) da SBC*. Anais do XXIII Congresso da Sociedade Brasileira de Computação, Campinas, p. 6. 2003.

[21] Smart, W. D. & Kaelbling, L. P. “Practical reinforcement learning in continuous spaces”. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*. pp. 903–910. 2000.

[22] Tesauro, G. “Temporal difference learning and td-gammon”. *Communications of the ACM*, 1995, 38(3), 1–3.