EXTRAÇÃO DE ESTRUTURAS DO AMBIENTE PARA ACELERAÇÃO DO APRENDIZADO POR REFORÇO EM UMA APLICAÇÃO DE ROBÓTICA MÓVEL

LUIZ A. CELIBERTO JR., REINALDO A. C. BIANCHI

Faculdade de Engenharia Industrial - FEI, Departamento de Engenharia Elétrica,
Av. Humberto de A. Castelo Branco, 3972 – CEP: 09850-901- São Bernardo do Campo, Brasil.
E-mails: celibertojr@uol.com.br, rbianchi@fei.edu.br

Abstract— This article discusses a technique for the acceleration of reinforcement learning, that uses the image of the value function generated by the Q-Learning algorithm to define heuristics to be used by a learning agent. This technique uses computer vision algorithms to recognize the structure of the environment in which the agent is inserted and determine where are the passages through which the agent can enter or leave a room. With these information, an heuristics for the agent can be defined and used to speed up the learning process.

Keywords— Reinforcement Learning, Mobile Robotics, Computer Vision.

Resumo— O Aprendizado por Reforço é uma técnica muito conhecida para a solução de problemas de controle, que tem sido cada vez mais usada em aplicações de robótica. Porém, ela não é eficiente o bastante para ser usada em aplicações com exigências do mundo real. Este artigo apresenta uma técnica que utiliza algoritmos de visão computacional para extrair informações que permitam a utilização de heurísticas para a aceleração do aprendizado por reforço. De maneira geral, as descontinuidades na função valor são usadas para encontrar ambientes onde pode ser utilizada uma meta-ação única. Experimentos realizados mostram que esta técnica permite uma aceleração significativa do aprendizado.

Palavras-chave— Aprendizado por Reforço, Robótica Móvel Inteligente, Visão Computacional.

1 Introdução

Aprendizado por Reforço (AR) é uma técnica muito atraente quando se deseja solucionar uma variedade de problemas de controle e planejamento quando não existem modelos disponíveis *a priori*, já que seus algoritmos têm a convergência para uma situação de equilíbrio garantidas, além de permitirem o aprendizado de estratégias de controles adequadas.

No aprendizado por reforço, o agente aprende por meio da interação direta entre o agente e o ambiente através de recompensas. Estas recompensas são dadas através de reforços positivos e negativos, que são usadas para sinalizar ao agente se ele esta tomando as ações corretas ou incorretas. O objetivo do agente sempre será acumular o máximo reforço positivo.

Porém, aprender não é suficiente: o agente precisa aprender rapidamente, pois o ambiente em que ele esta localizado pode sofrer constantes mudanças. Infelizmente, o aprendizado por reforço é muito lento, necessitando de milhares ou milhões de ações para que um agente aprenda corretamente: a convergência dos algoritmos de AR só pode ser atingida após uma extensiva exploração do espaço de estados-ações.

Bianchi (2004) mostrou que a velocidade de convergência de um algoritmo de AR pode ser acelerada ao se utilizar funções heurísticas para guiar a exploração do espaço de estados-ações. Ele também propôs diversos algoritmos acelerados por heurísticas, entre eles, o Q-Learning Acelerado por Heurísticas, baseado no conhecido algoritmo Q-Learning (Watkins,1989).

O objetivo deste artigo é apresentar uma nova maneira de extrair a heurística a ser usada pelo algoritmo Q-Learning Acelerado por Heurísticas e mostrar que esta técnica permite um ganho no tempo de aprendizado de um agente.

O algoritmo aqui proposto se utiliza do fato que durante o aprendizado, um agente passa pelos mesmos lugares inúmeras vezes, sendo que podemos usar a memória dos ambientes em que o agente já passou para acelerar o seu aprendizado.

Para tanto, utilizamos técnicas de visão computacional e processamento de imagens para, a partir de indícios existentes no processo de aprendizado, retirar estruturas que podem ser úteis para a aceleração, pois geralmente representam objetos no mundo real.

Este artigo é organizado da seguinte maneira: na próxima seção são apresentadas,de forma sucinta, características do algoritmo de aprendizado por reforço e o capítulo 3 introduz a aceleração heurística. A seção 4 descreve o funcionamento do sistema, a seção 5 discute os resultados e a última seção apresenta a conclusão deste trabalho.

2 Aprendizado por Reforço

No aprendizado por reforço, um agente sem conhecimentos prévios aprende através de interações com o ambiente, recebendo recompensas por suas ações e assim descobrindo a política ótima.

O aprendizado por reforço é uma técnica de aprendizado não supervisionado devido a não exis-

tência de uma representação de pares de entrada e de saída. Para cada movimentação do agente não é fornecida nenhum tipo de informação externa que ajude seu deslocamento, tirando aquela que ele mesmo percebe da sua interação com o ambiente (Kaelbling,1996).

No ambiente, em cada intervalo de tempo o agente executa uma ação (a_t). Esta ação é determinada pela política já aprendida, informando o agente para onde ele deve se locomover (s_{t+1}) e tendo em vista a recompensa ($r_{s,a}$) que irá ganhar. A recompensa pode ser dada por valores negativos ou positivos, indicando se o agente está seguindo corretamente para o objetivo ou não. A figura 1 apresenta um esboço do funcionamento do aprendizado por reforço.

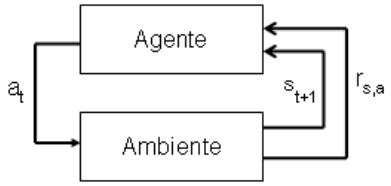

Fig 1. Aprendizado por reforço.

Dentro dos algoritmos de aprendizado por reforço, um dos mais conhecidos é o Q-Learning proposto por Watkins. No Q-Learning para cada movimento do agente é computado sua recompensa e o valor de seguir a melhor política com um desconto gama, esta política é aprendida através da interação com o ambiente e assim serão aprendidos quais os melhores movimentos para chegar a um objetivo, sendo a informação da política armazenada em uma matriz Q, que guarda os valores estimados por Q(estado, ação) (Uther e Veloso,2005).

O Q-Learning tem como proposta principal aprender através da interação uma política ótima π^* quando temos um desconhecido modelo de sistema. Esta política ótima é encontrada escolhendo a ação que maximiza os valores de Q, para um determinado estado. O valor de custo de um estado (função valor V(s)) também é encontrado facilmente, sendo que o custo de um estado V(s) é o valor máximo de Q(s,a) de um estado s, para todas as ações possíveis de serem executadas nesse estado.

Nesta técnica de aprendizado temos uma função de estimação Q que é calculada através da Exploração on-line do agente. A função de estimação Q é computada através da equação 1:

$$Q(s, a) = Q(s, a) + \alpha\,(r\,(s_t, a_t) + \gamma \max Q_t(s_t, a_t) - Q(s, a)) \quad (1)$$

O parâmetro γ é compreendido entre os valores de 0 a 1. Caso o valor de γ esteja muito perto de 0 o agente tende a considerar apenas valores imediatos de recompensa, caso os valores sejam muito perto de 1 o agente tem em vista as recompensas futuras com o maior peso.

Para a escolha das ações é usada uma estratégia conhecida como exploração aleatória ε-Greedy. O agente irá executar a ação que contenha o maior valor Q e probabilidade 1-ε e escolhe uma ação aleatória com probabilidade ε. A regra de transição de estados será dada pela seguinte equação:

$$\pi(s_t) = \begin{cases} a_{random} & \text{se } q \le \varepsilon, \\ \arg\max_{at} \hat{Q}_t(s_t, a_t) & \text{caso contrário,} \end{cases} \quad (2)$$

Sendo q um valor aleatório e com distribuição de probabilidade uniforme, ε compreendido entre os valores de 0 e 1 é o parâmetro que irá definir a taxa de exploração e de explotação e a_{random} é a ação aleatória selecionada.

O agente, através das interações com o ambiente, irá se deslocando de um estado ao outro e assim computando a função de estimação Q até conseguir chegar ao seu objetivo. Ao conseguir chegar ao seu objetivo é dito que ele completou um episódio.

A cada interação a tabela Q é atualizada. Depois de n interações é possível retirar valores condicionais desta tabela Q. Estes valores condicionais são relativos a um valor esperado da função Q seguindo uma política. Este valor condicional é chamado de função valor, que irá posteriormente influenciar na política de aprendizado e vice-versa.

Uma desvantagem do aprendizado por reforço é o tempo que o agente leva para aprender. Em muitos casos é necessário centenas de milhares de interações ou episódios para o agente poder aprender sobre onde está ou sobre o que está fazendo.

Devido a esta demora, é comum o uso de alguma forma de aceleração do aprendizado. A seção a seguir apresenta algumas técnicas que podem ser usadas para diminuir o tempo de aprendizado dos algoritmos de AR.

3 Aceleração do Aprendizado por Reforço

Um dos métodos de acelerar o algoritmo Q-Learning é aplicarmos uma heurística ou qualquer ajuda que acelere o aprendizado.

Utilizar uma heurística em um aprendizado por reforço, por exemplo, é fornecer uma ajuda ao agente para que ele possa se locomover melhor e conseguir chegar a um objetivo mais facilmente. Uma maneira é apresentada por Bianchi (2004), que define a heurística como uma técnica que no caso médio melhora a eficiência do algoritmo. Em seu trabalho, a heurística é usada para que o agente seja influenciado a escolher as ações durante o aprendizado.

Assim temos uma heurística definida por $H_t(s_t,a_t)$ que irá indicar a importância de executar em um estado s_t uma ação a_t. A política estará fortemente vinculada a sua função heurística. Sendo assim podemos dizer que teremos uma função heurística que é definida por uma "Política Heurística" (Bianchi,2004).

Devido aos algoritmos de aprendizado por reforço possuírem uma livre escolha das ações de treino, temos em uma heurística adequada uma aceleração do aprendizado e, no caso de uma heurística inadequada, temos um atraso do sistema que não o impede de convergir para um valor ótimo.

Bianchi (2004) também propôs diversos algoritmos acelerados por heurísticas, entre eles, o Q-Learning Acelerado por Heurísticas (HAQL), baseado no conhecido algoritmo Q-Learning. Este algoritmo funciona de maneira similar ao Q-Learning, porém permite que sejam utilizadas heurísticas que indicam o melhor caminho para um agente aprendiz seguir, em um determinado momento. As heurísticas são integradas ao funcionamento do algoritmo modificando-se a escolha das ações pelo agente. Mais detalhes posso ser encontrado em (Bianchi,2004).

Nos últimos anos, tivemos diversas outras abordagens para aceleração do aprendizado por reforço. Estas abordagens tentavam usar um melhor aproveitamento das experiências através de meios de generalização ou abstração.

Um exemplo conhecido que usa generalização temporal é o Q(λ) (Watkins,1992) que estende o Q-Learning, utilizando o conceito da elegitibilidade de um par estado/ação para a propagação das atualizações.Temos também o aprendizado usando como técnica de aceleração a utilização de base de casos (Drummond,2002).

Este método faz o uso das descontinuidades da função valor aprendido, como mostrado na figura 2, e conseguindo assim definir regiões correspondentes a sub-tarefas do sistema. Ao encontrar estas sub-tarefas o aprendizado é acelerado usando informações armazenadas em uma base de casos.

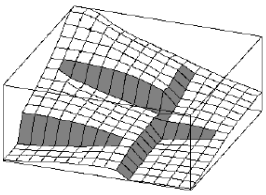

Fig 2. Descontinuidades da função valor (Drummond,2002).

4 O Algoritmo Proposto

O algoritmo proposto utiliza como entrada a tabela dos valores V(s) computada pelo algoritmo Q-Learning, a partir da tabela de valores Q. Esta tabela é uma imagem cujos valores dos pixeis correspondem aos valores V(s). Um exemplo da imagem criada é mostrado na figura 3.

Fig. 3. Imagens dos valores V(s).

Sobre esta imagem é aplicado um detector de bordas bem conhecido e usado na área de processamento de Imagens e Visão Computacional, o filtro Canny (Forsyth e Ponce,2003).

Alguma das razões que levaram ao uso deste filtro, é por ele ser considerado um filtro ótimo para detecção de bordas. O filtro Canny encontra bordas procurando por um máximo local do gradiente da imagem e este gradiente é calculado a partir da derivada de um filtro gaussiano.

Este tipo de filtro usa dois limiares para detectar bordas fortes e fracas e inclui as bordas fracas na saída somente quando elas estiverem conectadas a bordas fortes. O Filtro Canny é menos sensível a ruídos do que os demais, e o mais provável de detectar bordas fracas.

Depois de criada a imagem da função valor, esta é passada pelo filtro Canny e o resultado é mostrado na figura 4. A imagem de bordas gerada é muito parecida com uma imagem que contenha o mapa do ambiente.

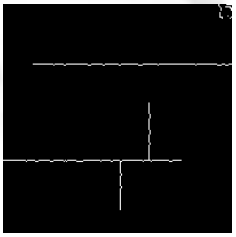

Fig. 4. Resultado do filtro Canny.

Após a detecção das bordas na imagem da função valor, é utilizado sobre a imagem resultante um algoritmo especialmente desenvolvido, com a finalidade de reconhecer ambientes fechados que correspondam a salas no mundo real e as passagens entre estas salas.

Este algoritmo procura, inicialmente, as bordas de imagens, isto é, variações de cores de branco para preto na imagem. Ao detectar qualquer borda na imagem, é determinada uma linha que segue a direção desta borda, computada a partir da diferença entre este ponto e seus dois vizinhos mais próximos.

Em seguida é feita uma varredura para procurar pontos que coincidem com esta linha. Caso mais que 15% dos pontos de bordas existentes na imagem se encontre sobre esta linha, é considerado que esta linha marca a separação entre dois ambientes.

Partindo do princípio que foi encontrada uma linha, a segunda parte do algoritmo deve detectar uma passagem que liga os dois ambientes que esta linha separa. Neste caso, foi adotado que qualquer espaço sobre a linha que marca a separação entre dois ambientes com tamanho maior que 35 pixels seriam considerado uma passagem. Valores abaixo deste considerados descontinuidades da linha por problemas diversos, não caracterizando uma passagem.

Detectadas as passagens entre os ambientes, é calculada a direção para qual o agente deve seguir a partir do gradiente sobre uma linha localizada perpendicularmente ao centro da passagem, na imagem original de valores V(s). Isto permite definir de qual sala o agente deve sair.

Uma terceira função do algoritmo é estender as linhas de passagens, criando novas passagens virtuais e desenhando com a extensão das passagens uma imagem quadriculada, como mostrada na figura 5. Esta imagem de saída é tratada com o algoritmo Blob Coloring (Ballard,1982).

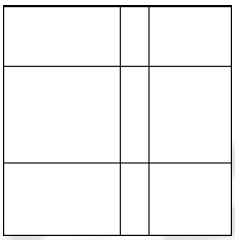

Fig. 5. Imagem de ambientes gerados pelo algoritmo.

Com o uso do Blob Coloring, iremos destacar cada sub-região dentro da imagem quadriculada. Esta informação será somada às informações anteriores já coletadas, como por exemplo a melhor orientação de uma passagem para a outra.

Os dados irão resultar em um diagrama de estados que fornecerá ao agente a ajuda necessária para acelerar o aprendizado, permitindo a criação de uma heurística com o caminho que o agente deve seguir.

5 Experimentos e Resultados Obtidos

O sistema foi implementado em linguagem C/C++ usando bibliotecas de visão computacional do OpenCV (OpenCV,2005) para aplicar o filtro Canny.

Nos experimentos realizados, foi possível detectar passagens com sucesso a partir do episódio 100 para as imagens nesta experiência. Valores de numero mínimo de episódios podem variar dependendo do tamanho da imagem e do quanto complexo for o ambiente.

Abaixo de 100 episódios não conseguimos informações necessárias de passagens, e assim não podemos ter uma certeza absoluta do número real de passagens que o ambiente possui, impossibilitando o uso de uma aceleração do aprendizado.

O número 100 de episódios mínimos foi obtido através de análises de dados empíricos como mostrado no gráfico da figura 7.

Para os dados levantados a partir da tabela da figura 7, foram usados 30 mundos diferentes, cada qual contendo 5 passagens.

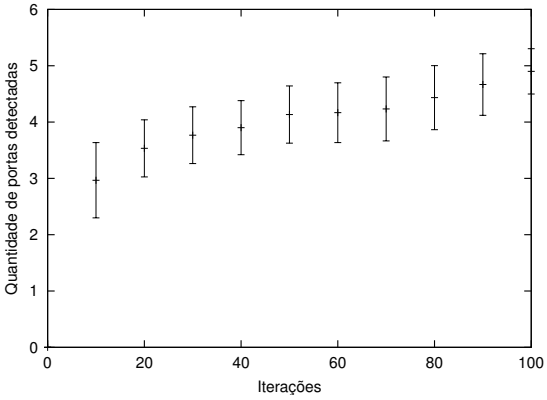

Fig. 6. Episódios versus número de passagens detectadas.

Assim sendo, para todos os resultados mostrados a seguir, e para todos os experimentos realizados, foram usadas imagens do episódio número 100.

Os resultados podem ser observados nas imagens da figura 7 compreendidas entre a imagem (a) e a imagem (e). Nestas imagens temos os processos de desenvolvimento da experiência e do reconhecimento de passagens. Para este mundo o objetivo do agente é conseguir chegar ao canto superior direito.

Na figura 7-a é apresentado o ambiente no qual o agente está inserido, sendo que ele inicia em uma posição aleatória e deve chegar a meta localizada no canto superior direito. É nesta representação de mundo que o agente irá se locomover e assim computar a função de estimação Q. Em nenhum momento o agente tem a noção desta imagem, sendo usado apenas para aplicar ao simulador empregado.

Na figura 7-b a imagem é criada com os valores da função V(s), extraída da tabela Q (s,a). É possível reconhecer que a imagem tem bordas em uma posição semelhança com a divisões entre ambientes na imagem em 7-a. Esta semelhança ficará sempre mais evidente com o passar do tempo e do nível de complexidade do ambiente.

Na figura 7-c é apresentada a imagem depois de tratada com o filtro Canny, recuperando as bordas da imagem e tornando ela bem nítida para ser tratada pelo algoritmo proposto. Esta imagem é obtida com os melhores valores possíveis de threshold do filtro Canny.

Na figura 7-d é apresentada o mundo quadriculado gerado pela definição das linhas que melhor representam as bordas existentes em 7-c e que define as salas (estados) existentes no mundo do agente e na figura 7-e temos a imagem com a localização das passagens detectadas (linhas brancas grossas) e das direções para cada passagem. Estas informações são

adicionadas à imagem do mundo que contém todas as informações sobre o melhor trajeto para o objetivo.

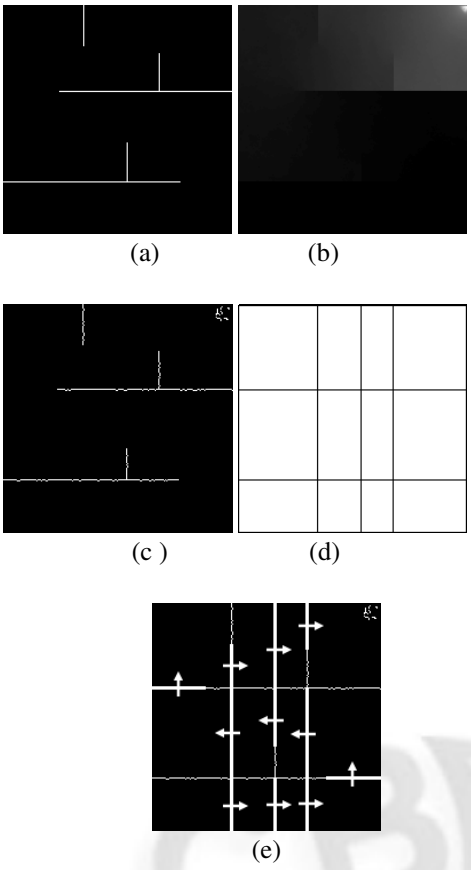

Fig. 7. Passos do experimento realizado.

Com os dados recolhidos pelo Blob Coloring retirado pela imagem em 7-d, e com as informações retiradas pelo algoritmo, é criado um diagrama de estados. O Blob Coloring adota como a primeira sub-região, no canto superior direito, como região 1. A próxima sub-região à esquerda como 2 e assim por diante até todas as sub-regiões serem numeradas.

A imagem de como são numeradas as sub-regiões é mostrada na figura 8, e o diagrama de estados é mostrado na figura 9. Assim, com o diagrama de estados, é possível passar ao agente a informação para onde ele deve ir dependendo do local que ele estiver, definindo uma heurística.

04	03	02	01
08	07	06	05
12	11	10	09

Fig. 8. Disposição das regiões.

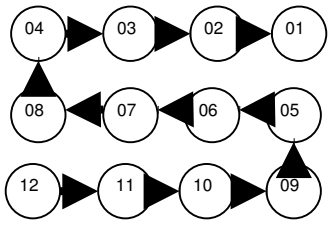

Fig. 9. Diagrama de relação entre as regiões encontradas no ambiente do agente (estados).

Sendo que também são conhecidas o tamanho de cada sub-região, bem como as suas áreas, todas estas informações são passadas ao agente. Para um mundo de 200 x 200 foram retiradas as seguintes coordenadas das sub-regiões mostradas na figura 10.

- Região: 1 Começa: x:127,y:137 Termina: x:198,y:198 Area: 4599
- Região: 2 Começa: x:127,y:109 Termina: x:198,y:134 Area: 1971
- Região: 3 Começa: x:127,y:71 Termina: x:198,y:106 Area: 2701
- Região: 4 Começa: x:127,y:2 Termina: x:198,y:68 Area: 4964
- Região: 5 Começa: x:48,y:137 Termina: x:124,y:198 Area: 4914
- Região: 6 Começa: x:48,y:109 Termina: x:124,y:134 Area: 2106
- Região: 7 Começa: x:48,y:71 Termina: x:124,y:106 Area: 2886
- Região: 8 Começa: x:48,y:2 Termina: x:124,y:68 Area: 5304
- Região: 9 Começa: x:2,y:137 Termina: x:45,y:198 Area: 2835
- Região: 10 Começa: x:2,y:109 Termina: x:45,y:134 Area: 1215
- Região: 11 Começa: x:2,y:71 Termina: x:45,y:106 Area: 1665
- Região: 12 Começa: x:2,y:2 Termina: x:45,y:68 Area: 3060

Fig. 10. Tamanho das sub-regiões

A partir destas informações, foi criada uma heurística para a movimentação do agente. Esta heurística é computada para todos os estados, e funciona como se fosse definida macro-ações a serem tomadas em cada sala, que leva o agente de um ponto qualquer em uma sala para a sua saída.

Para permitir uma comparação, foram realizados 30 experimentos em 10 ambientes diferentes (em um total de 300 experimentos) nos quais foi comparada a execução do algoritmo Q-Learning com a execução do algoritmo Q-Learning Acelerado por Heurísticas (HAQL).

O resultado da comparação pode ser vista na figura 11. Nela, pode-se ver que até o centésimo episódio tanto o Q-Learning quanto o HAQL aprendem de maneira similar, levando aproximadamente o mesmo número de passos para atingir a meta, a cada iteração. Porém, a partir da centésima iteração, o HAQL passa a utilizar a heurística definida a partir da função valor V(s). Pode se ver que a partir deste momento, o HAQL passa a executar o caminho ótimo a cada iteração, executando um número mínimo de passos (a linha se torna invisível por estar muito próxima ao valor zero). Enquanto isso o Q-Learning continua com aprendendo, levando ainda milhares de iterações para ser capaz de executar a tarefa no mesmo número de passos que o HAQL.

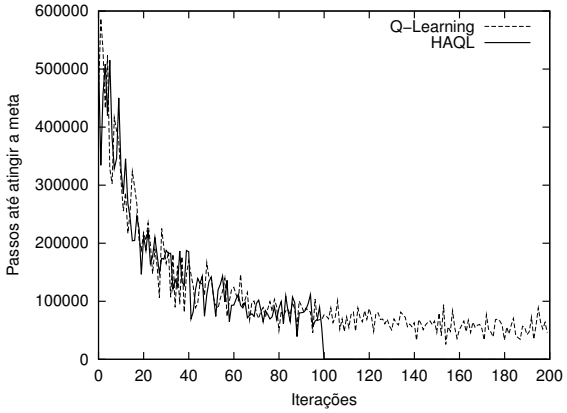

Fig. 11. Comparação entre o funcionamento dos algoritmos Q-Learning e Q-Learning Acelerado por Heurísticas.

6 Conclusão

Este trabalho mostra que é possível utilizar informações existentes no processo de aprendizado (a função valor V(s)) para definir uma heurística para um agente, acelerando o aprendizado por reforço em um estágio inicial do aprendizado.

Um dos fatores a ser observado é que, para os experimentos realizados, o algoritmo Q-Learning (sem a aceleração) só irá apresentar um desempenho similar ao Q-Learning acelerado por heurística quando o aprendizado chegar próximo às 100.000 iterações.

Finalmente, por ser uma fácil implementação, o algoritmo para a extração da heurística aqui proposta pode ser utilizado em diversos algoritmos de Aprendizado por Reforço Acelerado por Heurísticas existentes, como o SARSA Acelerado por Heurísticas (Bianchi,2004), entre outros.

Referências Bibliográficas

Ballard, D. H.; Brown, C. M. Computer Vision. Englewood Cliffs, Prentice-Hall, 1982.

Bianchi, R. A. C. Uso de Heurística para a aceleração do aprendizado por reforço. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, 2004.

Forsyth and Ponce ,Computer Vision: A Modern Approach, Prentice Hall, 2003.

Intel, Intel Open Source Computer Vision Library, Intel Research. Disponivel em: http://www.intel.com/research/mrl/research/opencv/, 2005.

Kaelbling, L.P.; Littman M. L.; Moore, W.A. Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research 4, May 1996, p. 237-285.

Uther, W; Veloso, M. Adversial Reinforcement Learning, Carnegie Mellon University, 2003

Drummond, C. Accelerating reinforcement learning by composing solutions of automati-cally identified subtask, Journal of Artificial Intelligence Research, p.59-104, 2002

Watkins, C. J. C. H. Learning from Delayed Rewards. Tese (Doutorado) —University of Cambridge, 1989.

Watkins, C. J. C. H.; DAYAN, P. Q-learning. Machine Learning, v. 8, p.279–292, 1992.