X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

MÉTODO HÍBRIDO DELIBERATIVO PARA A NAVEGAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS

Alcides X. Benicasa e Roseli A.F. Romero
Universidade Federal de Sergipe - DSI, Itabaiana, SE, Brasil
Universidade de São Paulo - USP/ICMC, São Carlos, SP, Brasil
alcides@ufs.br; {alcides, rafrance}@icmc.usp.br

Resumo – Este artigo tem como objetivo principal apresentar um método de navegação autônoma para robôs móveis utilizando uma arquitetura híbrida, composta por técnicas probabilísticas de mapeamento e aprendizado por reforço. O robô deverá aprender inicialmente os limites do ambiente e como se locomover de forma inteligente entre dois pontos distintos. Para a simulação do ambiente foram utilizados os softwares Player/Stage, que tornaram possível verificar o comportamento do robô móvel através do mapa utilizado. O método de mapeamento utilizado para a representação do ambiente foi baseado em grade de ocupação que, a seguir, foi utilizado para delimitar o ambiente no processo de aprendizado por reforço. As técnicas de aprendizado *Q-Learning* e *R-Learning* foram implementadas e comparadas. Os métodos demonstraram a capacidade de aprendizado pelo robô de forma a cumprir com sucesso os objetivos deste trabalho.

Palavras-chave – Robôs móveis autônomos, mapeamento métrico probabilístico e aprendizado por reforço.

Abstract – This article presents a method of autonomous navigation for mobile robots using a hybrid architecture, composed of mapping and probabilistic techniques of reinforcement learning (RL). The robot must first learn the limits of the environment and how to move intelligently between two distinct points. For the simulation environment we used the software Player/Stage, which made it possible to verify the behavior of the mobile robot used across the map. The mapping method used for the representation of the environment was based on grade of occupation, then was used to define the environment in the process of reinforcement learning. The both learning techniques Q-Learning and R-Learning have been implemented and compared. The methods demonstrated the ability of learning by the robot in order to successfully accomplish the goals of this work.

Keywords – Autonomous mobile robots, mapping metric probabilistic and reinforcement learning.

1 Introdução

A navegação autônoma de robô é atualmente um dos maiores desafios para pesquisadores desta área. Do ponto de vista biológico, para se locomover de forma inteligente em um ambiente inicialmente desconhecido faz-se necessário o mapeamento do ambiente, o que possibilitará encontrar melhores trajetórias para o cumprimento de uma determinada tarefa. Dentre as formas de mapeamento existentes, um dos métodos mais amplamente utilizados é o mapeamento métrico probabilístico, denominado Grade de Ocupação, proposto inicialmente por Elfes [1], e utilizado com sucesso em diversas pesquisas [2–4].

O controle da trajetória de robôs móveis autônomos é também considerada uma importante área de pesquisa. Para o desenvolvimento do processo de aprendizado de trajetórias são necessárias técnicas que possibilitem a comunicação do robô e o ambiente, permitindo escolher as melhores ações a serem executadas durante o aprendizado de caminhos. A teoria da análise do comportamento demonstra que os organismos vivos são capazes de aprender por meio de interações com o ambiente, recebendo estímulos reforçadores e punições em resposta às suas ações [5]. Para o controle de trajetórias, esses estímulos podem modificar diretamente o comportamento do robô durante o processo de aprendizagem, reforçando ou inibindo determinados caminhos. Para a validação do método proposto neste trabalho, foi investigado e comparado o potencial dos algoritmos de aprendizado Q-Learning e R-Learning.

A estrutura deste trabalho está organizada da seguinte forma: na seção 2 está descrito o processo de mapeamento do ambiente utilizando a representação espacial em grade de ocupação. Na seção 3 é apresentado o desenvolvimento do aprendizado das trajetórias utilizando os algoritmos Q-Learning e R-Learning. Em seguida, alguns resultados sobre as trajetórias aprendidas são apresentados na seção 4 e, por fim, as conclusões obtidas estão na seção 5.

2 Mapeamento com Representação Espacial em Grade de Ocupação

O processo de mapeamento pode ser visto como um modelo espacial do ambiente de trabalho do robô obtido a partir de informações sensoriais. A representação por grade de ocupação discretiza os espaços contínuos do ambiente, de forma que o ambiente passa a ser representado sob a forma de uma matriz. Cada elemento da matriz (também chamado de célula) representa um local do ambiente que pode estar, de acordo com uma formulação probabilística, ocupado ou vazio, ou pode ainda não ter sido explorado [6].

O mapeamento em grade de ocupação tem como objetivo a construção de mapas consistentes a partir de dados sensoriais, sobre a hipótese de que a posição do robô é conhecida. A idéia básica desse método de mapeamento é representar o ambiente em uma grade ou matriz multi-dimensional (geralmente 2D ou 3D), com células de mesmo tamanho, onde cada célula corresponde a uma variável aleatória cujo valor representa a probabilidade de ocupação [1,6] .

Os estados ou valores das células são estimados pela interpretação dos dados provenientes dos sensores de alcance modelados de forma probabilística por uma função densidade de probabilidade [1]. Através de regras probabilísticas Bayesianas é possível atualizar o valor das células, sempre que novas leituras dos sensores forem tomadas a partir de diferentes pontos do ambiente. O modelo espacial probabilístico resultante servirá como mapa do ambiente do robô e será usado diretamente para tarefas de navegação, como planejamento de caminho e desvio de obstáculo, estimativa de posição, entre outras. Neste trabalho será utilizado especificamente para o planejamento de caminhos.

De acordo com os objetivos deste trabalho, e para efeitos de simplificação, assumiu-se a situação na qual se utiliza um sensor ideal, caracterizado por uma função de densidade de probabilidade $p(r|z) = d(r-z)$, em que r é a leitura sensorial obtida pelo robô; z é a distância atual do objeto detectado; e d é o delta de Kronecker. Neste contexto, foi utilizado a forma fechada abaixo (Equação 1).

$$P[s(C_i = Ocupada|r] = \begin{cases} 0 & x < r, x \in C_i \\ 1 & x, r \in C_i \\ 1/2 & x > r, x \in C_i \end{cases} \quad (1)$$

em que, $s(C_i)$ é o estado de ocupação da célula C_i. Assim, se um sensor ideal captura uma medida de distância r, as células correspondentes têm probabilidade de ocupação 1; as células precedentes são vazias e, portanto, possuem probabilidade de ocupação 0; e as células sucessoras que ainda não foram observadas assumem a probabilidade de ocupação $\frac{1}{2}$.

2.1 Mapeamento baseado em sonares

Os sonares estão distribuídos na parte frontal do robô (*array* de sonares). Considerando as regiões cobertas por um determinado sonar, pode-se observar na Figura1 as regiões I, II e III, representando, respectivamente, a área livre de obstáculos, o obstáculo e a área das células sucessoras.

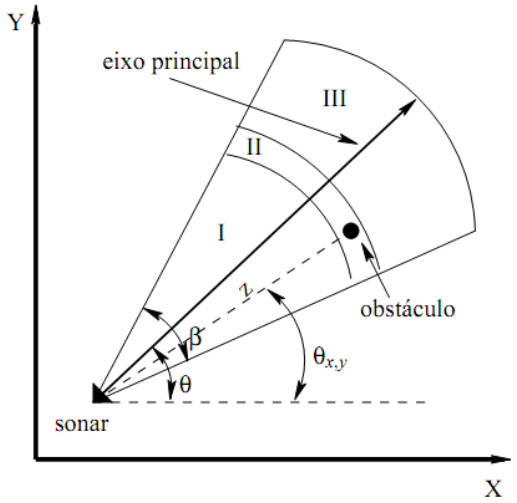


Figura 1: Regiões cobertas por um sonar. Fonte: [6]

Para a implementação do mapeamento métrico baseado em grade de ocupação foi realizada a adaptação do algoritmo inicial proposto por [6,7], descrito a seguir.

Algoritmo 1: Grade de Ocupação $(m_{x,y}, x_t, z_t)$

```
for todas as celulas m_{x,y} do
    if m_{x,y} está_no_campo_visual de z_t then
        Atualizar valor de ocupação de m_{x,y} de acordo com Equação 1;
    else
        Manter valor de ocupação atual;
    end
end
```

O Algoritmo 1 tem como variáveis de entrada uma matriz $m_{x,y}$ contendo todos os valores de ocupação atribuídos a grade de ocupação construída até o instante, o vetor de localização do robô $x_t = (x, y, \theta)$ no instante t e os valores de leitura dos sensores z_t no instante t. Para cada célula $m_{x,y}$ da grade construída é verificado se está dentro do campo de visão formado por todos os sonares. Caso esteja, o valor de ocupação da célula será atualizado de acordo com a Equação 1. Caso a célula não esteja no campo de visão dos sonares, esta permanecerá com o mesmo valor de ocupação.

A função *está_no_campo_visual* descrita no Algoritmo 2 tem como entradas a célula $m_{x,y}$ a ser analisada, o vetor de localização do robô x_t e as medições sensoriais z_t. Inicialmente são calculadas a distância e a orientação entre o robô e a célula, tornando possível identificar o sensor com orientação mais próxima à célula.

Encontrado o sonar k mais próximo à célula $m_{x,y}$, verifica-se a célula está dentro do seu campo visual, baseando-se na distância z_t^k medida por este sonar e no seu ângulo de orientação $\theta_{k,sens}$ em relação ao robô. O parâmetro z_{max} é a distância máxima possível de ser medida pelo sonar, β é a abertura do feixe do sonar e α é um valor que indica a espessura média dos obstáculos no ambiente. Caso o teste seja verdadeiro, a célula recebe o valor de ocupação 0.5, indicando que ainda não foi possível mapeá-la.

Algoritmo 2: Está no Campo Visual $(m_{x,y}, x_t, z_t)$

Definir X, Y como centro de massa de $m_{x,y}$;
$r = \sqrt{(X-x)^2 - (Y-y)^2}$;
$\phi = atan2(Y - y, X - x)$;
$k = argmin_j |\phi - \theta_{j,sens}|$;
if $r > min(z_{max}, z_t^k + \alpha/2)$ *ou* $(|\phi - \theta_{k,sens}| > \beta/2)$ **then**
  retornar 0.5;
end
if $(z_t^k < z_{max})$ *e* $(|r - z_t^k| < \alpha/2)$ **then**
  retornar 1;
end
if $(r \leq z_t^k)$ **then**
  retornar 0;
end

Caso o teste feito tenha resultado falso, verifica-se se a distância medida pelo sensor z_t^k é menor que distância máxima z_{max} e se a célula está compreendida na região do obstáculo detectado. Caso verdadeiro, a célula recebe o valor de ocupação 1, indicando sua ocupação. Para finalizar, se a expressão for falsa, é realizado um teste para verificar se a distância entre o robô e a célula é menor que a medição z_t^k do sonar. Se verdadeiro, a célula recebe o valor 0 (*zero*), o que a identifica como uma célula disponível. De forma geral, o Algoritmo 2 tem como função atribuir valores constantes para as células que estão dentro do cone do sonar, na extremidade frontal do cone e na região fora no cone.

Para o desenvolvimento deste trabalho foram considerados os seguintes parâmetros: $\beta = 20$, $\alpha = 0.7$ e $z_{max} = 5$. A Figura2 apresenta o ambiente original e sua representação espacial em grade de ocupação após a execução do processo.

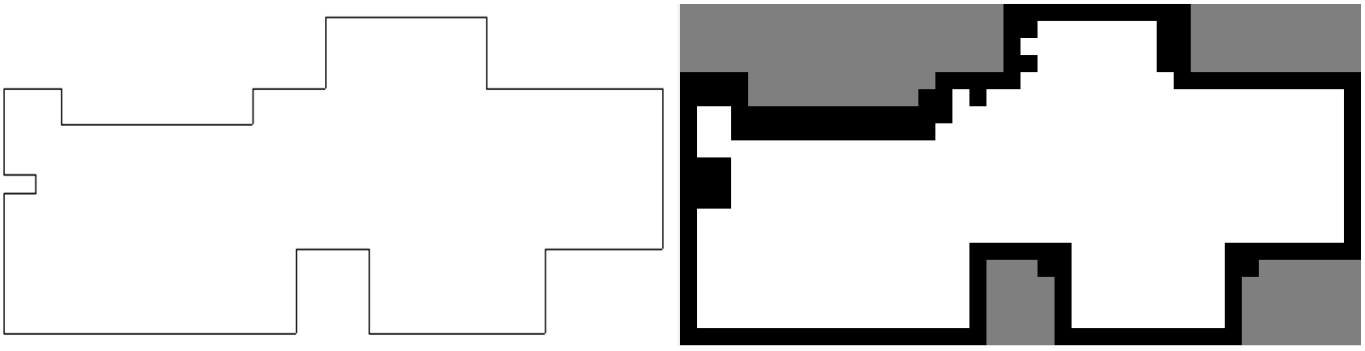


Figura 2: Ambiente Original (esquerda) / Grade de Ocupação (direita)

Devido ao fato de que o mapeamento será utilizado como modelo espacial para processo de aprendizado por reforço, foi criado o arquivo denominado `occ_grid_map.txt`. De maneira geral, o arquivo é composto por informações pertinentes a cada célula $m_{x,y}$, de acordo com a Equação 1.

3 Aprendizado por Reforço

Como um dos objetivos principais deste trabalho, necessitamos que o robô aprenda a se locomover, de forma inteligente, através da representação espacial criada (Figura2). Considerando que o ambiente é inicialmente desconhecido, no que refere-se

a ausência de qualquer política de movimentação e ainda, limitando-se somente ao espaço de navegação representado espacialmente, os algoritmos de aprendizado por reforço podem ser utilizados para encontrar caminhos ótimos entre pontos definidos previamente, como um modelo promissor para o processo de aprendizado por interações com o ambiente.

O Aprendizado por Reforço (AR) é um modelo para mapear ações e situações, de maneira a maximizar o valor do estímulo positivo fornecido pelo ambiente, baseado no mesmo princípio do condicionamento operante: o princípio do reforço [5,8]. Nesse modelo não é informado ao agente (robô) quais ações devem ser tomadas em determinadas situações. O robô deverá descobrir quais ações tomadas em determinados contextos do ambiente resultam em uma maior recompensa. O AR é definido por não caracterizar métodos de aprendizado, mas por caracterizar um problema de aprendizado. Composto como segue:

- um conjunto de estados do ambiente $S = s_1, s_2, \ldots, s_n$;
- um conjunto de possíveis ações $A = a_1, a_2, \ldots, a_n$;
- uma função de probabilidade de transição $P(s'|s, a)$, que determina a probabilidade de se ir para o estado s' dado que se está no estado s e é tomada a ação a;
- reforços associados à ação a tomada no estado s, $r(s, a) \in R$.

Em um problema de AR, o ambiente é inicialmente desconhecido, de forma que as probabilidades de transição devem ser, implícita ou explicitamente, aprendidas com o passar do tempo. Os algoritmos de AR que aprendem diretamente a política ótima, incorporando implicitamente essas probabilidades de transição, são geralmente os mais populares, por serem mais simples e fáceis de implementar [9]. Este trabalho se detém no uso de dois algoritmos de aprendizado por reforço: *Q-Learning* e *R-Learning*.

Conforme especificações pré-definidas para a validação deste trabalho, os algoritmos citados deverão aprender a política ótima referente às seguintes coordenadas do ambiente mapeado: Posição da Lixeira = $(-19, 2)$, Posição do Objeto 1 = $(2, 4)$ e Posição do Objeto 2 = $(9, -7)$, conhecendo previamente a Posição Inicial do robô $(-19, 8)$) (Figura3).

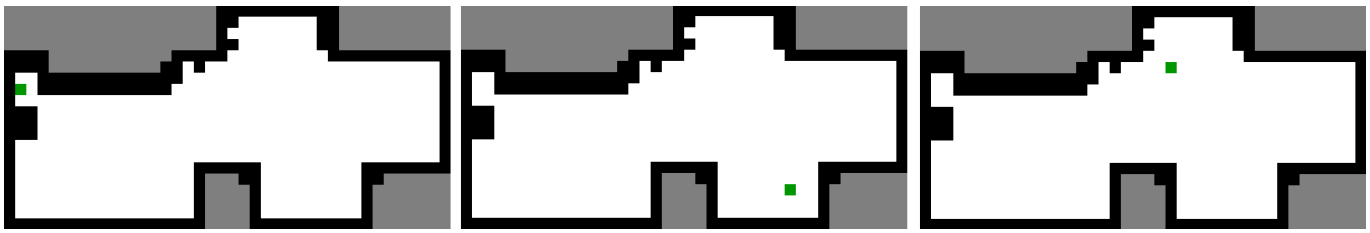


Figura 3: lixeira / objeto 1 / objeto 2

A tarefa a ser executada após o treinamento consiste em o robô se locomover ao objeto 1, levando-o em seguida à lixeira, locomover-se ao objeto 2 e finalizar levando-o também para a lixeira. Para a aplicação desenvolvida para os aprendizados *Q-Learning* e *R-Learning* iniciar corretamente faz-se necessário que o arquivo `occ_grid_ map.txt` esteja no diretório da aplicação.

3.1 Q-Learning

O algoritmo de aprendizado *Q-Learning* é um dos métodos de aprendizado por diferença temporal mais simples de ser implementado [8, 10]. Sua regra de atualização é dada pela Equação 2.

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \tag{2}$$

onde $\alpha \in]0, 1]$ é uma taxa de aprendizado e $\gamma \in [0, 1[$ é a taxa de desconto utilizado para garantir que os valores de Q sejam finitos, ambos parâmetros de projeto.

Dessa forma, uma tabela com os valores de $Q(s, a)$ é construída conforme o robô percorre o espaço de estados do ambiente e explora seu espaço de ações. Isso significa que o $resultado$ do aprendizado é registrado em uma tabela Q. Prova-se que os valores dessa tabela convergem para os valores de $Q^*(s, a)$ caso todos os pares (estado, ação) sejam visitados infinitas vezes [11]. Satisfazer essa condição é impossível na prática, mas esse resultado teórico é uma indicação de que após os estados serem visitados muitas vezes, a política determinada a partir da tabela $Q^*(s, a)$ corresponderá à política ótima.

No Algoritmo 3, após executar a ação a, o agente sai do estado s e vai para um estado s', recebendo por esta ação uma recompensa imediata r. No estado s' é feita uma busca, entre as ações disponíveis, para encontrar a ação a' que tenha o maior valor de retorno esperado, representado por $\max_{a'} Q(s', a')$. Caso a ação a' seja tomada como sendo a próxima a ser executada, possivelmente existirá a probabilidade de se atingir máximos locais. Sendo assim, foi utilizado neste trabalho, para cada passo do episódio, em 70% dos casos, a ação que retorne o valor máximo e nos outros 30% faz-se escolhas aleatórias para evitar os máximos locais. As Figura4,5 e 6 apresentam os gráficos de convergência referente ao aprendizado das trajetórias entre o robô e objeto 1, objeto 2 e lixeira, respectivamente. Os parâmetros utilizados para o aprendizado de *Q* foram: $\alpha = 0.3$ e $\gamma = 0.8$.

Algoritmo 3: Algoritmo Q-Learning

Inicie $Q(s, a)$ arbitrariamente;
for *cada episódio* **do**
  Obtenha o estado atual s do ambiente;
  for *cada passo do episódio* **do**
    Selecione uma ação $a \in A(s)$;
    Execute a ação a;
    Observe os valores de s' e r;
    Atualize o valor de $Q(s, a)$ (Equacão 2);
    $s \leftarrow s'$;
  end
end

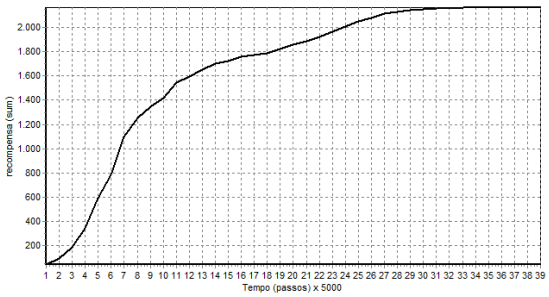


Figura 4: Gráfico de convergência do aprendizado Q do objeto 1

3.2 R-Learning

A técnica de aprendizado por reforço denominada *R-Learning* maximiza a recompensa média a cada passo, ou seja, utiliza *average-reward model*. O *Q-Learning* não maximiza a recompensa média, mas descontos acumulados de recompensa, sendo assim, esperamos que o *R-Learning* encontre maiores máximos globais e não apenas máximos locais, como é o caso do *Q-Learning* [12, 13].

O algoritmo *R-Learning* possui regra similar ao *Q-Learning*, sendo baseado na dedução de valores $R(s, a)$, e devendo escolher ações a em um estado s. A cada passo, o robô escolhe a ação que tem o maior valor $\max_a R(s, a)$, exceto que algumas vezes a escolha é aleatória. Os valores de $R(s, a)$ são ajustados a cada ação baseado na Equação 3,

$$R(s, a) = R(s, a) + \alpha(r - p + \max_{a'} R(s', a') - R(s, a)) \tag{3}$$

diferindo da regra do *Q-Learning* somente por subtrair a recompensa média p do reforço imediato r e por não ter o desconto γ para o próximo estado. O cálculo da recompensa média é apresentado na Equação 4.

$$p = p - \beta(r - p + \max_{a'} R(s', a') - \max_{a} R(s, a)) \tag{4}$$

onde p somente é atualizado quando uma ação não aleatória foi tomada. É importante observar que a recompensa média p não depende de algum estado particular, sendo uma constante para todo o conjunto de estados. O Algoritmo 4 descreve o método *R-Learning*.

Algoritmo 4: Algoritmo R-Learning

Inicie p e $R(s, a)$ arbitrariamente;
for *sempre* **do**
  Obtenha o estado atual s do ambiente;
  Selecione uma ação $a \in A(s)$;
  Execute a ação a;
  Observe os valores de s' e r;
  Atualize o valor de $Q(s, a)$ (Equacão 3);
  if $R(s, a) = \max_a R(s, a)$ **then**
    Atualize o valor de p (Equação 4);
  end
end

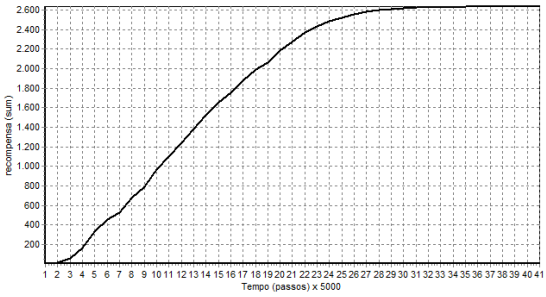


Figura 5: Gráfico de convergência do aprendizado Q do objeto 2

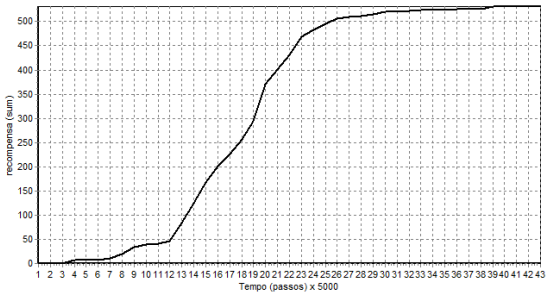


Figura 6: Gráfico de convergência do aprendizado Q da lixeira

Os parâmetros utilizados para o aprendizado de *R* foram: $\alpha = 0.9$ e $\beta = 0.6$. As Figura7,8 e 9 apresentam os gráficos de convergência referente ao aprendizado das trajetórias.

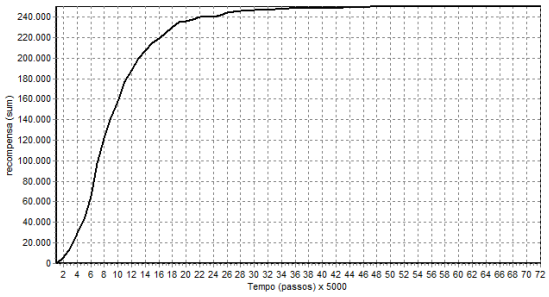


Figura 7: Gráfico de convergência do aprendizado R do objeto 1

Pode-se observar que o algoritmo *Q-Learning* converge mais rapidamente e, para todos os itens, com um menor número de passos em relação ao algoritmo *R-Learning*. Entretanto, o *R-Learning* conseguiu maximizar a recompensa, ao contrário do *Q-Learning* que caiu em máximos locais. A Tabela 1 apresenta os principais pontos desta comparação.

Tabela 1: Comparação de Tempo de Convergência e Maximização de Recompensa (*Q-Learning* e *R-Learning*)

$Algoritmo$	$obj.1.tempo$	$obj.1.r_{max}$	$obj.2.tempo$	$obj.2.r_{max}$	$lixeira.tempo$	$lixeira.r_{max}$
$Q-Learning$	39	2.200	41	2.650	43	550
$R-Learning$	72	250.000	150	180.000	460	270.000

4 A Trajetória

Como mencionado nas seções anteriores, ao término do aprendizado, a trajetória do robô pelo ambiente deverá consistir, de forma ordenada, dos seguintes passos:

- atingir a posição do Objeto 1;
- encaminhar-se à Lixeira;
- atingir a posição do Objeto 2;

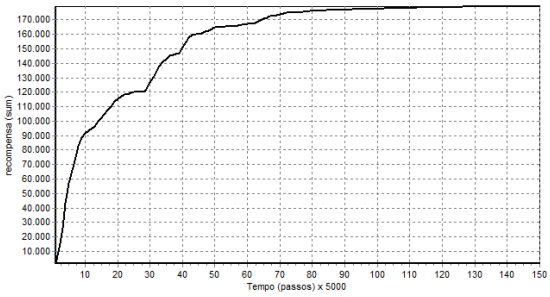


Figura 8: Gráfico de convergência do aprendizado R do objeto 2

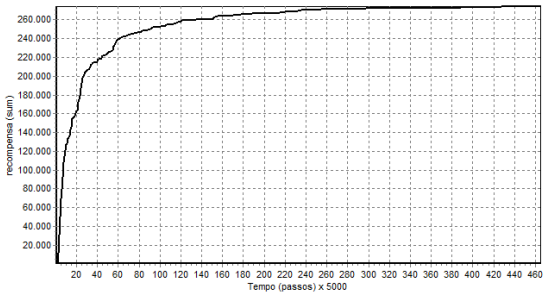
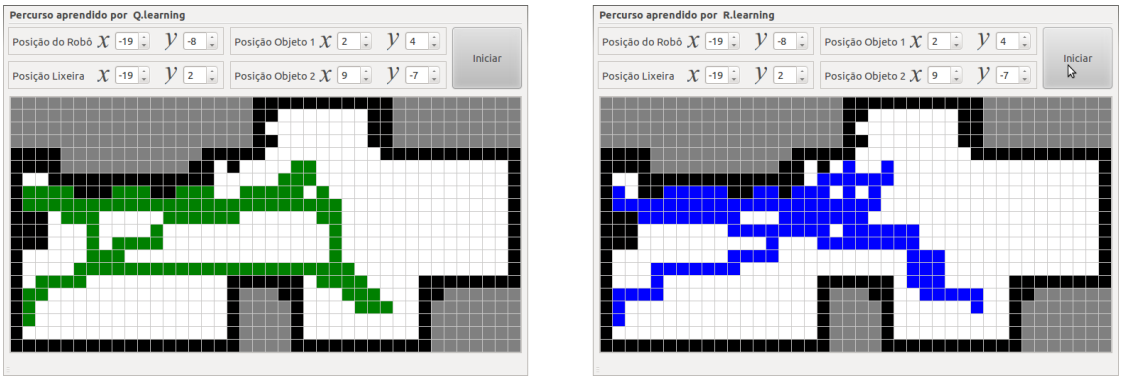


Figura 9: Gráfico de convergência do aprendizado R da lixeira

- retornar à Lixeira.

Foram desenvolvidos dois aplicativos para esta etapa final, responsáveis pela realização da trajetória do robô pelo ambiente original utilizando os aprendizados Q e R, respectivamente.



(a) (b)

Figura 10: Trajetória utilizando a tabela Q(a) e tabela R(b)

Uma observação importante a se destacar é que o arquivo `occ_grid_map.txt`, criado na primeira etapa deste trabalho, não se faz necessário neste momento. O robô irá necessitar somente dos arquivos referente aos aprendizados Q e R. A Figura10 apresenta as trajetórias realizadas pelo robô utilizando os aprendizados obtidos pelos métodos *Q-Learning* (Figura10(a)) e *Q-Learning* (Figura10(b)).

5 Conclusões

Pode-se concluir, como apresentado na Figura10(b), que as trajetórias realizadas utilizando o aprendizado R possuem a forma de parábolas, ao contrário das trajetórias utilizando o aprendizado Q, que apresentam ângulos mais retos. Sendo assim as distâncias percorridas utilizando R são menores que as distâncias percorridas utilizando Q. Isso dá-se ao fato de o aprendizado *Q-Learning* ter como uma de suas características principais a maximização da recompensa média e, conforme esperado, o *R-Learning* encontrou maiores máximos globais e não apenas máximos locais, como foi o caso do *Q-Learning*.

O processo desenvolvido nestas etapas mostrou-se adequado para a validação do método proposto, apto assim a aprender inicialmente os limites de um ambiente desconhecido e locomover-se, de forma inteligente, entre objetivos pré-definidos. Outras técnicas de aprendizado podem ser utilizadas, avaliadas e comparadas, de modo a obter-se mecanismos aptos a propor trajetórias inteligentes e econômicas, não somente aplicada ao cenário deste projeto.

REFERÊNCIAS

[1] A. Elfes. "Sonar-based real-world mapping and navigation". *IEEE Journal of Robotics and Automation*, vol. 3, pp. 249–265, 1987.

[2] J. MULLANE, M. D. ADAMS and W. S. WIJESOMA. "Robotic Mapping Using Measurement Likelihood Filtering". *The International Journal of Robotics Research*, vol. 28, pp. 172–190, 2009.

[3] S. NOYKOV and C. ROUMENIN. "Occupancy grids building by sonar and mobile robot". *Robotics and Autonomous Systems*, vol. 55, pp. 162–175, 2007.

[4] A. BIRK and S. CARPIN. "Merging Occupancy Grid Maps From Multiple Robots". *In Proceedings of the IEEE*, vol. 94, pp. 1384–1397, 2006.

[5] S. Thrun, B. W. and F. D. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, EUA, 1998.

[6] A. Souza. "Mapeamento com Sonar Usando Grade de Ocupação Baseado em Modelagem Probabilística." Master's thesis, Universidade Federal do Rio Grande do Norte - UFRN, 2008.

[7] S. Thrun, B. W. and F. D. *Probabilistic Robotics*. MIT Press, Cambridge, Massachusetts, EUA, 2005.

[8] C. A. Policastro. "Arquitetura robótica inspirada na análise do comportamento". Ph.D. thesis, Universidade de São Paulo, Outubro 2008.

[9] A. Selvatici. "AAREACT: Uma Arquitetura Comportamental Adaptativa para Robôs Móveis que Integra Visão, Sonares e Odometria." Master's thesis, Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais, 2005.

[10] C. Watkins. "Models of delayed reinforcement learning". Ph.D. thesis, PhD thesis, Psychology Department, Cambridge University., Cambridge, 1989.

[11] T. M. Mitchel. *Machine Learning*. McGraw Hill, Boston, 1997.

[12] A. Schwartz. "In Machine Learning: Proceedings of the Tenth International Conference, San Mateo, CA". *IEEE Transactions on Neural Networks*, 1993.

[13] G. Faria and R. Romero. "Explorando o Potencial de Algoritmos de Aprendizado com Reforço em Robôs Móveis". *Proceedings of the IV Brazilian Conference on Neural Networks*, pp. 237–242, 1999.