Modelagem de Um Agente Móvel de Aprendizagem para Vagueio em Ambientes Inexplorados

Leandro Toss Hoffmann[1], **José Demisio Simões da Silva**[1]

[1]Programa de Pós-graduação em Computação Aplicada
Instituto Nacional de Pesquisas Espaciais (INPE)
Av. dos Astronautas, 1758 –12.227-010 – São José dos Campos – SP – Brazil

{hoffmann,demisio}@lac.inpe.br *

Abstract. *This work shows a learning mobile agent model which performs a wander task, while it tries to explore an unknow terrain in an homogeneous way. The learning task, as well the decision make process, is acquired by Q-learning algorithm, that is by a reinforcement learning approach. Beyound sensor data, the state mapping function uses agent's built-in variables. Those variables are represented by a fuzzy-cartesian map which tracks the agent's wandering and counts the visitation indicative of explorated regions. The developed model is validated using simulations of navigation environmnets with or without obstacles. The results are compared to a random navigation strategy and quatitatived by presented indicators.*

Resumo. *Este trabalho apresenta a modelagem de um agente móvel de aprendizagem para executar a tarefa de vagueio, objetivando a exploração de um terreno desconhecido de forma homogênea. A aprendizagem da tarefa de navegação, bem como o processo de tomada de decisão, é realizadas através do algoritmo Q-learning de aprendizagem por reforço. Além dos dados sensoriais, a função de mapeamento de estados utiliza variáveis internas do agente. Essas variáveis são representadas por um mapa nebuloso-cartesiano, que acompanha o deslocamento do agente pelo ambiente e contabiliza o índice de visitação das regiões exploradas. O modelo desenvolvido é validado por meio de simulações em ambientes de navegação com ou sem obstáculos. Os resultados são comparados com uma estratégia de navegação aleatória e avaliados de forma quantitativa com indicadores que serão apresentados.* [1]

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. Na prática, devido a complexidade computacional do problema, o uso de técnicas de computação inteligente vêm se popularizando.

[1]Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte a realização deste trabalho.

Neste trabalho, é apresentado um modelo de navegação (*wandering*) para guiar um robô móvel em um ambiente desconhecido. Por ser baseado na arquitetura de um agente de aprendizagem [Russell and Norvig 2004], os termos agentes e robô são utilizados indistintamente. Técnicas de aprendizagem, em especial 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. Uma das principais vantagens do RL é viabilizar o aprendizado *online* do agente, embora a convergência ótima da escolha das ações possa ser lenta, dependendo do algoritmo utilizado.

Vários trabalhos na literatura mostram aplicações de aprendizado por reforço na área de robótica e navegação. Nos trabalhos de Gaskett e outros [Gaskett et al. 2000], Smart e Kaelbling [Smart and Kaelbling 2001]e Zhu e Levinson [Zhu 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 e Alsina [Aranibar and Alsina 2004] aplicam o algoritmo *Q-learning* no planejamento de rotas de robôs autônomos. Bhanu e outros [Bir Bhanu Pat Leang and Patterson 2001] apresentam uma implementação física do ambiente de labirintos, descritos no trabalho de Sutton e Barto [Sutton and Barto 1998]. Yen e Hickey [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. Um dos algoritmos mais populares em aplicações de RL é o *Q-learning* de Watkin, porém técnicas híbridas também são exploradas. Faria e Romero [Faria and Romero 2004] 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 em Macek e outros [Macek et al. 2002] para tratar o problema de representação estrutural de estados contínuos, em aplicação de desvio de obstáculos em robôs móveis.

O objetivo desse trabalho é prover ao robô uma exploração homogênea do terreno, ou seja, a trajetória deve primar por regiões do ambiente menos visitadas. Para tanto, o sistema de navegação é composto pelo algoritmo *Q-learning*, além de um mapa nebuloso-cartesiano, que auxilia o agente a registrar seu deslocamento no ambiente. Desta forma, esse artigo descreve inicialmente, na Seção 2., a teoria de aprendizagem por reforço. Em seguida, na Seção 3., é apresentada a base da teoria de conjuntos nebulosos, necessários para a construção do mapa nebuloso-cartesiano. Posteriormente, a Seção 4. traz alguns resultados de experimentos realizados; e finalmente a Seção 5. conclui o trabalho e discute trabalhos futuros.

2. Aprendizagem por reforço

Em um modelo de aprendizagem por reforço, um agente aprende a otimizar sua interação com o ambiente por tentativa e erro. O ambiente pode ser parcialmente ou completamente observável e, 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 estado. Para cada ação tomada, o agente pode receber uma recompensa, que será utili-

zada para avaliar o seu comportamento. 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 interação com o ambiente, o agente aprenderá a tomar decisões em determinadas situações que o levam a maximizar a soma de prêmios recebidos [Sutton and Barto 1998]. Para cada estado S_i então, existe um valor de utilidade associado, e uma *função de valor* $V(S)$ que mapeia estados para valores de estados. Uma política π determinará que ações deverão ser escolhidas em cada estado e, quando a política ótima π^* é conhecida, é possível encontrar uma função ótima de valores $V^*(S)$, e vice-versa. Durante o processo de aprendizagem, busca-se aproximar π^* e $V^*(S)$ [Sutton and Barto 1998].

O algoritmo *Q-learning* é uma técnica popularmente utilizada em aprendizagem por reforço, combinada com o método de aprendizagem por Diferença Temporal. 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 a partir de um dado estado [Sutton and Barto 1998]. 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 dado pelo maior Q-valor desse estado, pode-se dizer que um Q-valor é definido pela Equação (1):

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

3. Lógica nebulosa e Mapas Nebuloso-Cartesiano

Na teoria clássica dos conjuntos, um conjunto inclui ou não um dado elemento, permitindo assim apenas dois valores de pertencimento: 0 ou 1 (falso ou verdadeiro). Na teoria dos conjuntos nebulosos, por outro lado, um elemento x pode pertencer a um conjunto nebuloso A completamente ($\mu_A = 1$), ou com um certo grau de pertencimento ($0 < \mu_A(x) < 1$). Logo, a Lógica Nebulosa é uma lógica de multi-valor que permite o uso de termos lingüísticos para definir valores intermediários, além de valores convencionais do tipo ”sim/não”, ”verdadeiro/falso”ou ”preto/branco”.

Num sistema de Lógica Nebulosa, variáveis lingüísticas são mapeadas por um ou mais conjuntos nebulosos, identificados por termos lingüísticos. Define-se suporte do conjunto nebuloso A ($Su(A)$) todos os elementos do universo de discurso que tenham valores de pertinência maior que zero; e Núcleo do conjunto nebuloso A ($Nu(A)$) todos os elementos do universo de discurso que tenham valores de pertinência igual a um.

3.1. Mapas nebulosos-cartesiano

Tunstel e Jamshidi [Tunstel and Jamshidi 1994] propuseram o uso de Lógica Nebulosa na construção de mapas, que auxiliariam a navegação de robôs. O mapa seria descrito em eixos de coordenadas num universo de variáveis nebulosas, ou seja, cada coordenada seria representada por termos lingüísticos. Por exemplo, a localização do agente ou de obstáculos poderia ser representada por coordenadas do tipo: um pouco ao leste e muito ao norte; ou região central ao sul.

A vantagem de se utilizar um mapa nebuloso-cartesiano está no fato dos dados sensoriais e atuadores serem imprecisos. Quando esses registros são tratados e manipulados em termos lingüísticos, o dado ruidoso torna-se mais tolerável. Além disto, em

uma tarefa de navegação, é justo que registros de exploração sejam feitos em coordenadas nebulosas-cartesiana, pois no mundo real é difícil de definir fronteiras precisas entre regiões adjacentes. Por outro lado, atualizar um mapa nebuloso-cartesiano pode se tornar mais custoso computacionalmente do que atualizar mapas tradicionais que usam apenas um valor simples de coordenada cartesia.

4. Experimentos e Resultados

Os experimentos que serão descritos a seguir combinam as técnicas de aprendizado por reforço e de lógica nebulosa, apresentadas anteriormente, num contexto de agente móvel e autônomo. A idéia geral da modelagem consiste num agente de aprendizagem, provido com o algoritmo $Q - learning$, mas que também acompanha as modificações do ambiente, mantendo variáveis internas, representadas por um mapa nebuloso-cartesiano.

4.1. Modelagem do mapa nebuloso-cartesiano

O mapa nebuloso-cartesiano é utilizado pelo agente para demarcar as regiões visitadas no terreno explorado. Cada região é representada por uma coordenada nebulosa-cartesiana. O modelo de mapa nebuloso adotado nestes experimentos consiste em 2 variáveis lingüísticas (eixo de coordenadas x e y), cada uma com 5 termos lingüísticos. Assim, para o eixo x tem-se os termos: N, NC, CX, S e SC. (denotando Norte, Norte Centro, Centro X,...). E analogamente para o eixo y : O, OC, CY, LC e L.

Assumindo que o agente move-se sempre numa distância constante d, optou-se em delimitar a área do ambiente a ser explorada em $25d \times 25d$. Com isso, estipulou-se de forma empírica, os conjuntos nebulosos por funções lineares definidas por: $Su(N) = Su(O) = [0, 7)$; $Nu(N) = Nu(O) = [0, 2]$; $Su(NC) = Su(OC) = (2, 12)$; $Nu(N) = Nu(O) = \{7\}$; $Su(CY) = Su(CX) = (7, 17)$; $Nu(CY) = Nu(CX) = \{12\}$; $Su(SC) = Su(LC) = (12, 22)$; $Nu(SC) = Nu(LC) = \{17\}$; $Su(S) = Su(L) = (17, 24]$; e $Nu(S) = Nu(L) = [22, 24]$.

4.2. Modelagem do agente

O agente móvel deve explorar o ambiente, inicialmente desconhecido, tomando a cada instante de tempo t, apenas uma das seguintes **ações**: deslocar-se à frente a uma distância d (*AcFrente*), rotacionar 90° à esquerda (*AcEsquerda*) ou 90° à direita (*AcDireita*).

Cada estado S é definido por uma quádrupla $S = (s^1, s^2, s^3, s^4)$, onde s^1, s^2, s^3 e s^4 são símbolos que representam a situação da região adjacente à esquerda, à frente, à direita e atrás, respectivamente. Isto significa que o estado é composto por informações da vizinhança 4 da região atual. Os símbolos s^i podem assumir os identificadores apresentados na Tabela 1.

Para completar a modelagem do agente de aprendizagem, define-se sua função de **reforço** r, através da Equação 2, onde PA é um Prêmio por Afastamento e PR um Prêmio por Região.

$$r = \begin{cases} -1,0 & \text{se o agente colidiu} \\ -0,2 + PA + PR & \text{caso contrário} \end{cases} \quad (2)$$

O objetivo imediato da Equação 2 é punir o agente caso ele venha colidir com algum obstáculo. Se os termos PA e PR forem nulos, o agente terá uma pequena punição,

Tabela 1. Símbolos que podem compor a codificação de um estado do agente

ID	Símbolo	Ocasião de escolha
0	x	se existe um obstáculo imediato na direção da região adjacente
1	:	se existe um obstáculo não imediato na direção da região adjacente, mas sua distância é menor do que a distância até a região adjacente
2	*	se a região adjacente é a menos explorada da vizinhança
3	˜	se a região adjacente é a mais explorada da vizinhança
4	|	se a região adjacente pertence ao limite do mapa, ou seja, existe uma parede no limite externo da região vizinha
5	.	caso nenhuma das opções anteriores se adequar

ou seja, toda ação será punida com uma perda de energia. O termo PA (Equação 3) é uma pequena premiação caso o agente se afaste da região adjacente mais visitada.

$$PA = \begin{cases} +0,25 & \text{se } s_{t-1}^4 = 3 \text{ e a ação foi } \textit{AcFrente} \\ 0 & \text{caso contrário} \end{cases} \tag{3}$$

Já o termo PR (Equação 4) é a parte da recompensa responsável por induzir o agente a navegar por regiões g pouco exploradas. Ela só é ativada quando o agente muda de região predominante no mapa. A premiação consiste no valor de pertencimento da nova região explorada, em função da localização atual do agente. Soma-se a isto um prêmio extra e subtraí-se alguns descontos.

$$PR = \begin{cases} extra + \mu_Y * \mu_X - iTxVt & \text{se } g_t\left(y,x\right) \neq g_{t-1}\left(y,x\right) \\ 0,0 & \text{caso contrário} \end{cases} \tag{4}$$

A premiação extra (Equação 5) induz o agente a navegar para uma região pouco explorada e o inibe a navegar para uma região muito explorada. O valor de desconto $iTxVt$ será apresentado na sub-secção a seguir, na Equação 7, que é grande para regiões muito exploradas no mapa e pequeno caso a região seja pouco explorada.

$$extra = \begin{cases} 2,0 & \text{se } s_{t-1}^2 = 2 \\ 0,0 & \text{se } s_{t-1}^2 = 3 \\ 0,5 & \text{caso contrário} \end{cases} \tag{5}$$

Para treinamento do algoritmo *Q-learning*, foram utilizados os seguintes valores: taxa de aprendizado α = 0,1; fator de desconto γ = 0,8; número de ações tomadas pelo agente = 1 milhão; e probabilidade de escolha aleatória de ações $\epsilon - greedy$ inicialmente fixado em 0,9, decaindo linearmente a partir da ação de número 200 mil, até ser zerada.

4.3. Indicadores para avaliação dos resultados

Para auxiliar o processo de avaliação dos resultados obtidos, a seguir serão definidos alguns indicadores específicos:

- **Número de Visitas (*iVt*)**: é a quantidade de passeios numa dada região g do terreno explorado, num instante de tempo. É representado pela soma de todas marcações efetuadas pelo agente em seu mapa nebuloso-cartesiano.

- **Média de Visitação do Terreno (*iMdVt*)**: é o número médio de visitas de cada região do terreno. Em razão da modelagem do mapa nebuloso-cartesiano, pode ser determinada simplesmente por $iMdVt_t = t/25$.
- **Distância de Visitação (*iDstVt*)**: representa a homogeneidade de visitação do terreno. É calculado pelo desvio padrão [2] do número de visitas de cada região, normalizado pela Média de Visitação do Terreno, como visto na Equação 6. Quanto maior a distância, maior a heterogeneidade no número de visitas das regiões.

$$iDstVt_t = \sqrt{\frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left(\frac{iVt_t\left(g\left(y,x\right)\right)}{iMdVt_t} - 1\right)^2} \tag{6}$$

  onde: Y: é o número de variáveis nebulosas na coordenada y do mapa;
  X: é o número de variáveis nebulosas na coordenada x do mapa;
  g: é a uma região (x, y) no mapa nebuloso-cartesiano.
- **Taxa de Visitação (*iTxVt*)**: compara o quão distante o número de visitas de uma dada região, em t, está da região mais visitada (Equação 7).

$$iTxVt_t\left(g\right) = \frac{iVt_t\left(g\right)}{\max_g iVt_t\left(g\right)} \tag{7}$$

- **Índice de Visitação (*iInVt*)**: indica a média de visitação de um terreno, tomando como referência o valor de visitação da região mais visitada (Equação 8). Assim, quanto mais próximo de 1 esse indicador estiver, maior o número de regiões com visitação próxima ao máximo.

$$iInVt_t = \frac{1}{Y.X}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left(\frac{iVt_t\left(g\left(y,x\right)\right)}{\max_g iVt_t\left(g\right)}\right) \tag{8}$$

4.4. Um ambiente simples sem obstáculos

A fim de validar o modelo proposto, a seguir são apresentados alguns experimentos realizados com o agente de aprendizado, num ambiente simulado. Inicialmente, a primeira simulação é realizada num ambiente onde não existem obstáculos, além daqueles que delimitam o mapa. Para comparar a estratégia de navegação baseada em aprendizado por reforço, experimentos com uma política de escolha de ações aleatória ($\epsilon = 1$) são efetuados. Foram realizadas 10 simulações, com 10 mil ações, com o algoritmo *Q-learning* já treinado, reposicionando o agente aleatoriamente a cada nova simulação.

As curvas médias de Distância de Visitação do algoritmo *Q-learning* e da política aleatória, estão ilustradas no gráfico da Figura 1. Observa-se que o algoritmo *Q-learning* converge rapidamente para um *iDstVt* inferior, permanecendo estável ao longo das iterações. Ainda na Figura 1, observa-se a curva de convergência do *iDstVt* para a política aleatória, ao longo de 1 milhão de ações.

4.5. Ambientes com obstáculos

Com a inserção de obstáculos no ambiente, a navegação se torna mais complexa, exigindo do agente a capacidade de evitar colisões, enquanto vagueia pelo terreno. Neste trabalho, foram utilizados três tipos de ambientes com obstáculos, ilustrados na Figura 2, onde os

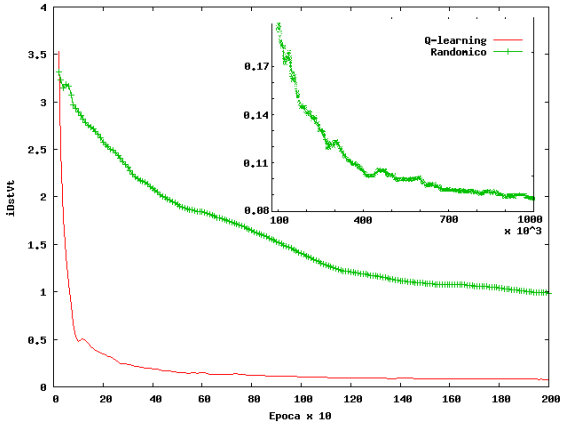

Figura 1. Curvas das médias de iDstVt obtidas pelo algoritmo *Q-learning* e navegação Aleatória, em função de t, em ambiente sem obstáculos.

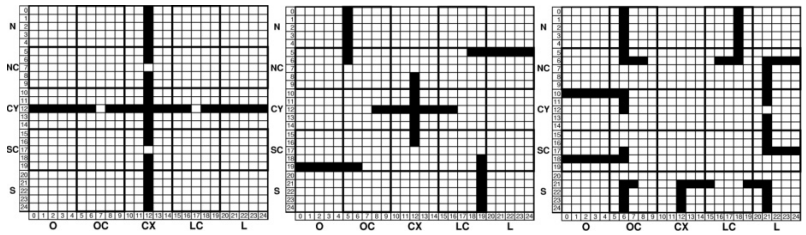

Figura 2. Mapa de obstáculos utilizados nas simulações. Da esquerda para a direita: mapa Cruz, Cata-vento e Escritório.

obstáculos estão identificados em preto. Vale ressaltar que nas simulações, assume-se ambientes fechados, com obstáculos padrão nos limites dos mapas.

Utilizando o mapa de obstáculos *Cruz*, o algoritmo *Q-learning* foi novamente treinado, com os mesmos parâmetros do experimento anterior. A Figura 3 mostra os gráficos das curvas das médias dos indicadores *iDstVt* e *iInVt*, durante as 2 mil e 10 mil ações inicias, respectivamente. Nota-se através do indicador *iDstVt* que ambos algoritmos têm uma maior dificuldade em manter uma cobertura homogênea do terreno, mas o algoritmo baseado em aprendizado por reforço obtém uma melhor convergência. Essa homogeneidade fica mais clara através do indicar *iInVt*.

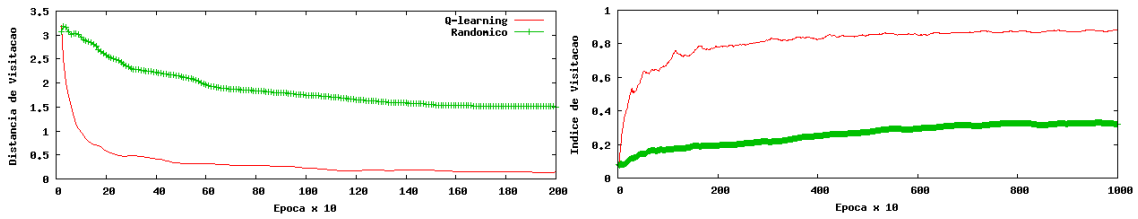

Figura 3. Curvas das médias de iDstVt e iInVt obtidas pelo algoritmo *Q-learning* e navegação Aleatória, em função de t, no ambiente *Cruz*.

Simulações com o mapa de obstáculos *Cata-vento* foram efetuadas a fim de se observar a generalização no aprendizado do algoritmo *Q-learning*. Para tanto, o algoritmo

[2]A rigor, no cálculo do desvio padrão, a variância tem como divisor o número de amostras menos 1.

de aprendizado por reforço foi treinado num ambiente representado pelo mapa *Escritório*, mas avaliado no mapa *Cata-vento*.

O primeiro gráfico da Figura 4 demonstra a capacidade de navegação do agente de aprendizagem, através da convergência do indicador *iDstVt* em relação a estratégia de navegação aleatória. O segundo gráfico da Figura 4 indica a média de recompensas coletadas a cada 100 ações tomadas na simulação. Neste gráfico é possível observar a otimização da navegação aprendida pelo algoritmo *Q-learning*, segundo a função de recompensas estabelecida.

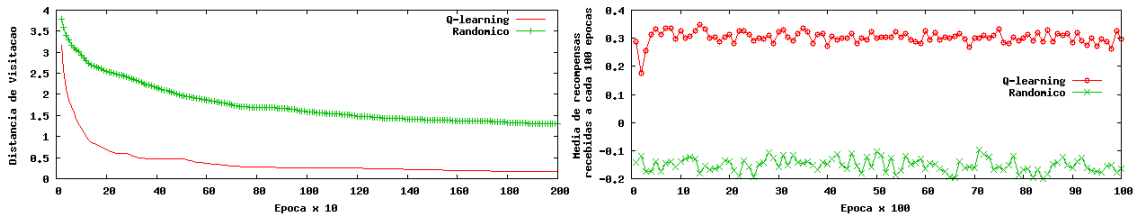

Figura 4. Curvas das médias de iDstVt e Recompesas obtidas pelo algoritmo *Q-learning* e navegação Aleatória, em função de t, no ambiente *Cata-vento*.

4.6. Nova modelagem da arquitetura do agente

Não é incomum que a estratégia adotada na navegação entre em conflito com as ações tomadas para evitar colisões. Este fato foi observado em simulações de navegação do agente, em um ambiente com obstáculos do mapa *Escritório*. O resultado é que o agente fica preso a uma determinada região do terreno e não consegue, por conta do seu mapeamento de ações, navegação adequadamente. Isto ocorre, porque mesmo em ambientes para o qual o agente foi treinado, existem situações em que o mapeamento de dados sensoriais para estados-ações, leva a uma incerteza de percepção (*perceptual aliasing*), como já foi descrito por outros autores [Crook and Hayes 2003].

Para minimizar esse problema, propôs-se uma modificação na arquitetura do agente. A literatura tem mostrado que impraticável a obtenção de um sistema de navegação suficientemente robusto que seja unicamente reativo ao ambiente. [Heinen 2001], [Medeiros 1998] e [Song and Sheen 2000]. A nova modelagem proposta para a arquitetura do agente vai de encontro com essa idéia, combinando em 3 módulos, sub-sistemas de control de colisão, navegação e detecção de armadilhas:

- Módulo de **anti-colisão**, que pode ser um sub-sistema de aprendizado por reforço, específico para evitar colisões, ou regras simples de anti-colisão;
- Módulo de **navegação**, idêntico a modelagem que foi utilizada até o momento;
- Módulo de **detecção de armadilhas**, que contém variáveis de estado para monitorar o mapa nebuloso-cartesiano e identificar possíveis armadilhas.

O módulo de navegação continuará encarregado de selecionar as ações a serem tomadas, mas pode ser censurado pelo módulo de anti-colisão, caso venha a escolher uma ação que leve a uma colisão. O módulo de detecção de armadilhas, por sua vez, aumenta o coeficiente $\epsilon - greedy$, sempre que uma situação de armadilha for detectada. Essa detecção é feita através do monitoramento do indicador *iDstVt* em função do tempo. Para tanto, o agente armazena duas variáveis internas: $iDstVtD_t = iDstVt_t - iDstVt_{t-1}$

e $iDstVtI_t = iDstVtI_{t-1} + iDstVt_t - SetPoint$, onde $SetPoint$ é uma constante indicadora de $iDstVt$ ideal. Sempre que essas variáveis ultrapassarem um determinado limiar, o coeficiente $\epsilon - greedy$ é incrementado, caso contrário decrementado.

Com essas medidas, o agente foi capaz de escapar de armadilhas, e continuar a navegar pelo terreno do mapa *Escritório* de forma homogênea. Esse comportamento é observado nos gráficos da Figura 5, onde os indicadores de Distância de Visitação e Índice de Visitação convergiram de maneira semelhante aos experimentos anteriores.

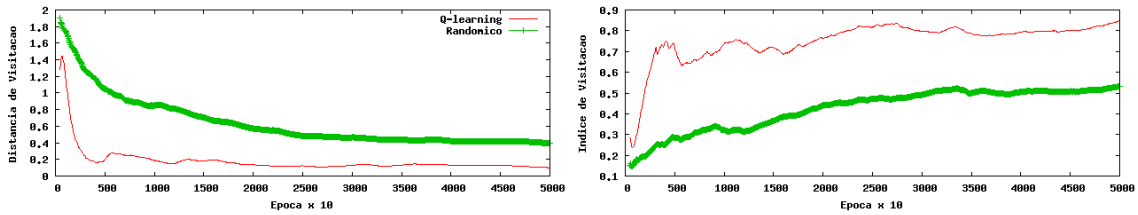

Figura 5. Curvas das médias de iDstVt e iInVt obtidas pelo algoritmo *Q-learning* e navegação Aleatória, em função de t, no ambiente *Escritório*.

5. Discussão e Conclusões

Este trabalho apresentou a modelagem de um agente móvel de aprendizagem para navegar num ambiente desconhecido. Como método de aprendizagem foi utilizado o algoritmo *Q-learning*, objetivando a exploração homogênea de um ambiente. As tarefas de aprendizagem e navegação foram auxiliadas por um mapa nebuloso-cartesiano, utilizado para contabilizar as regiões exploradas pelo agente.

Foram experimentados quatro ambientes, na sua maioria com obstáculos, onde a qualidade da navegação foi quantificada em termos de indicadores, descritos neste trabalho. Os indicadores obtidos nas simulações do sistema de navegação com aprendizagem por reforço foram comparados com resultados obtidos por uma estratégia de navegação aleatória. Notou-se que em todos ambientes simulados, o algoritmo de navegação baseado em aprendizado por reforço obteve os melhores índices de desempenho, demonstrando sua capacidade de orientar a navegação do agente por ambientes inexplorados, de forma homogênea. Além disto, através da simulação da navegação do agente num ambiente sem prévio treinamento, observou-se a capacidade de generalização do modelo desenvolvido e do algoritmo *Q-learning*, bem como a viabilidade do seu uso na exploração de ambientes desconhecidos e pouco acessíveis para treinamento.

Através dos resultados apresentados, encorajaram o uso da modelagem de navegação baseada em aprendizado por reforço, por meio da obtenção de índices de desempenho superiores a uma estratégia de navegação puramente aleatória, além da rápida convergência de índices de exploração homogêneos. Essa vantagem, pode ser aproveitada em missões de robôs, onde exista restrição de tempo para exploração, mas deseja-se uma cobertura geral do terreno.

Por outro lado, dependendo da configuração do ambiente, o agente pode ter dificuldade para escapar de armadilhas, devido a disposição dos obstáculos. Sendo assim, como trabalhos futuros, objetiva-se realizar um número maior de experimentos a fim de se observar o comportamento do agente nessas situações, e aprimorar a estratégia de detecção e escape dessas regiões.

Por último, espera-se avaliar o uso dos mapas nebulosos-cartesianos em diferentes dimensões de ambientes, buscando-se uma adaptação dinâmica do tamanho do mapa. A medida que algumas área do terreno foram suficientemente exploradas, pode-se alterar a granularidade do mapa e aumentar a área de exploração. Ou de forma similar, para uma pequena região de interesse, modificar o mapa para explorá-la com maior atenção.

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*.

Crook, P. A. and Hayes, G. (2003). Learning in a state of confusion: perceptual aliasing in grid world navigation. In *Proceedings of TIMR 2003 - Towards Intelligent Mobile Robots*, pages 1–10, Bristol.

Faria, G. and Romero, R. A. F. (2004). Estratégia para futebol de robôs baseada em campos potenciais. In *EnRI - XXIV Congresso da Sociedade Brasileira de Computação*, page 10, Salvador.

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*.

Heinen, F. J. (2001). Sistema de controle híbrido para robôs móveis autônomos. Master's thesis, Universidade do Vale do Rio dos Sinos, 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.

Medeiros, A. A. D. (1998). A survey of control architectures for autonomous mobile robots. *Journal of the Brazilian Computer Society*, 4(3):35–43.

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

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

Song, K. T. and Sheen, L. H. (2000). Heuristic fuzzy-neuro network and its application to reactive navigation of a mobile robot. *Fuzzy Sets and Systems*, 110:331–340.

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

Tunstel, E. and Jamshidi, M. (1994). Fuzzy logic and behavior control strategy for autonomous mobile robot mapping. In *IEEE International Conference on Fuzzy Systems*, Orlando.

Yen, G. and Hickey, T. (2002). Reinforcement learning algorithms for robotic navigation in dynamic environments. In *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.