OBTENÇÃO DE MAPAS COGNITIVOS PARA O ROBÔ MÓVEL MAGELLAN PRO

SILDOMAR T. MONTEIRO, CARLOS H. C. RIBEIRO

Instituto Tecnológico de Aeronáutica, Divisão de Ciência da Computação
Praça Mal. Eduardo Gomes, 50, Vila das Acácias
12228-900 São José dos Campos – SP
E-mails: sildomar@comp.ita.br, carlos@comp.ita.br

Resumo— Este artigo apresenta a implementação de um método para o aprendizado eficiente de mapas cognitivos de ambientes fechados em um robô móvel Magellan Pro. O modelo de mundo obtido é constituído por partições de tamanho variável representando regiões sensorialmente homogêneas do ambiente. O método utilizado é uma combinação dos paradigmas métrico e topológico e produz um mapa de partições retangulares de baixa complexidade, baseado na suposição de que os obstáculos são paralelos ou perpendiculares uns aos outros. Uma rede neuronal é utilizada para interpretar leituras dos sonares. Resultados experimentais mostram que o método permite o aprendizado de um mapa confiável do ambiente para o robô móvel Magellan Pro, usando apenas leituras de sonar.

Abstract— *This paper presents the implementation of a method for the efficient learning of cognitive maps of indoor environments on a Magellan Pro mobile robot. The obtained world model is constituted by partitions of variable size representing perceptually homogeneous regions of the environment. The method used is a combination of metric and topological paradigms and produces a map of rectangular partitions of low complexity, based on the assumption that obstacles are parallel or perpendicular to each other. A neural network is used to interpret sonar readings. Experimental results show that the method allows learning of a reliable map of the environment for the Magellan Pro robot, using solely sonar readings.*

Keywords— *Autonomous mobile robots, map learning, neural networks, occupancy grid, robotics, topological graph.*

1 Introdução

A criação de mapas a partir de dados sensoriais locais coletados à medida que o robô se move por um ambiente desconhecido é um dos maiores objetivos da pesquisa de robôs móveis. Utilizando algoritmos para aprendizagem de mapas, pode-se adquirir autonomamente um modelo do mundo ao redor do robô.

O principal objetivo deste trabalho é a implementação de um algoritmo que permita adquirir um mapa do ambiente utilizando o robô Magellan Pro em ambiente real, que servirá como base para testes de algoritmos de aprendizagem para controle (aprendizagem por reforço de tarefas de navegação, por exemplo).

Existem muitos tipos diferentes de mapas usados para localização de robôs em ambientes fechados, baseados na forma da informação sensorial e nos requisitos de representação da localização. Na literatura, encontramos duas abordagens principais: o paradigma métrico e o paradigma topológico.

O paradigma métrico (ou geométrico) permite a construção de um modelo que é uma representação geométrica do mundo. Um exemplo desta abordagem é um sistema de *grids* de ocupação, na qual *grids* de duas dimensões espaçados de forma simétrica apresentam a probabilidade de ocupação de cada célula do *grid* em correspondência àquela área no mundo. Neste modelo, os *grids* de ocupação distinguem lugares diferentes baseado na posição geométrica do robô dentro de uma estrutura de coordenadas globais. Já o paradigma topológico é mais qualitativo. Ele consiste de um grafo no qual os nós representam regiões sensorialmente diferentes do mundo e os arcos indicam relações espaciais entre eles. Neste modelo, a posição do robô relativa ao modelo é determinada com base em pontos de referência (*landmarks*) ou características sensoriais momentâneas distintas. A resolução do mapa topológico é proporcional ao grau de complexidade do ambiente.

Ambos os paradigmas possuem qualidades e fraquezas bem distintas e complementares (Thrun, 1998). O modelo métrico é fácil de construir, representar e manter, permite o reconhecimento de lugares (baseado em geometria) de forma não ambígua, e facilita o cálculo do caminho mais curto, enquanto o modelo topológico é difícil de construir e manter em ambientes grandes se a informação dos sensores for ambígua, apresenta reconhecimento de lugares sempre difícil, e pode produzir caminhos sub-ótimos. Por outro lado, o modelo topológico permite planejamento eficiente, é mais eficiente computacionalmente (resolução depende da complexidade do ambiente), não necessita determinação precisa da posição do robô, enquanto o modelo métrico apresenta planejamento ineficiente, é computacionalmente menos eficiente (resolução não depende da complexidade do ambiente) e necessita determinação precisa da posição do robô.

O método adotado para a implementação é uma combinação dos paradigmas métrico e topológico para construção de mapas. Utilizou-se integralmente a abordagem proposta por (Arleo, Millan e Floreano, 1999), que apresenta um algoritmo para o aprendizado eficiente de mapas cognitivos de resolução variável para navegação autônoma de robôs em ambientes fechados. O modelo gerado consiste de partições de resolução variável que representam regiões sensorialmente homogêneas. O mapa resultante é uma representação reduzida da estrutura geométrica do mundo,

permitindo uma otimização no uso dos recursos de memória e tempo. Nenhum sistema de visão computacional é utilizado, apenas leituras de sonares.

2 Mapas Cognitivos de Resolução Variável

O método proposto usa um conjunto P de partições de resolução variável para modelar um ambiente fechado estruturado. Áreas sensorialmente homogêneas são modeladas como uma partição $p \in P$, um modelo local que codifica o conhecimento do robô sobre aquela região. Esta abordagem permite a construção de um modelo simplificado do mundo, sendo que as partições de resolução variável possuem sempre formato retangular. Isto reduz a complexidade de atualização e manutenção das partições, mas parte da suposição de que todos os obstáculos são paralelos ou perpendiculares entre si e que suas bordas estejam alinhadas com os eixos *x* e *y*, o que nem sempre é a realidade.

O algoritmo para aprendizagem de mapas cognitivos de resolução variável é apresentado na Fig. 1.

Início
Repetir
1. Exploração: Seleção do próximo alvo.
2. Planejamento: Cálculo de uma trajetória para o alvo através das partições atuais.
3. Ação: Realiza o movimento do robô através de controladores de baixo nível.
4. Se o robô atinge o alvo, então ele explora esta região exaustivamente.
5. Atualização do mapa: Se uma incoerência foi detectada, então o robô desiste da exploração e modela o obstáculo desconhecido.
   5.1. Aproxima-se do obstáculo
   5.2. Enquanto o robô não visita uma canto conhecido do obstáculo
      5.2.1. Alinha-se com um dos limites do obstáculo
      5.2.2. Analisa o limite e o aproxima a uma linha reta utilizando uma interpretação neuronal dos sensores.
      5.2.3. Move-se até o final do limite
      5.2.4. Memoriza o canto do obstáculo
   5.3. Aumenta a resolução da partição.
   5.4. Atualiza o banco de dados de experiências negativas
   5.5. Remove partições redundantes do conjunto de partições
   5.6. Abstrai um novo grafo topológico.
Até que o mapa esteja construído.
Fim

Figura 1. Algoritmo de aprendizagem de mapas cognitivos de resolução variável.

O processo de aprendizagem do mapa pode ser descrito como um algoritmo cíclico composto de atividades de exploração e atualização do modelo. O robô explora o ambiente baseado no conhecimento adquirido, e apenas interrompe a exploração para incorporar um obstáculo desconhecido ao modelo, atualizando a resolução das partições.

O algoritmo tem uma estrutura modular, sendo identificados seis módulos principais que são explicados em detalhes nas subseções a seguir.

2.1 Interpretação neuronal dos sensores

Os dados provenientes dos sensores do robô são interpretados através de uma rede neural *feed-forward*, resultando em um *grid* de ocupação que melhora a percepção local do robô.

Para a interpretação neural dos sensores utilizou-se o método proposto originalmente por Moravec (1988) e Elfes (1987), com o objetivo de realizar uma leitura mais confiável dos sensores - que geralmente são afetados por ruídos de distribuição desconhecida - e permitir uma interpretação desta leitura para todos os sensores simultaneamente.

A partir das leituras dos sensores, a rede neural N permite a obtenção de um *grid* de ocupação local G constituído de $n \times n$ células. Este *grid* de ocupação é uma visão do mundo ao redor do robô, que se move e gira junto com ele.

Para cada célula $(i, j) \in G$, a entrada da rede N consiste de um vetor de dados constituído pelas leituras $s = (s_1, s_2, s_3)$ de três sensores orientados na direção de (i, j) e pelas coordenadas polares do centro da célula (i, j) em relação ao robô, sendo θ_{ij} o ângulo relativo ao primeiro dos três sensores mais próximos da célula e d_{ij} a distância relativa ao centro do robô. A saída da rede é um valor $\Pr ob(occ_{ij} \mid s)$ que indica a probabilidade da célula estar ocupada (Fig. 2).

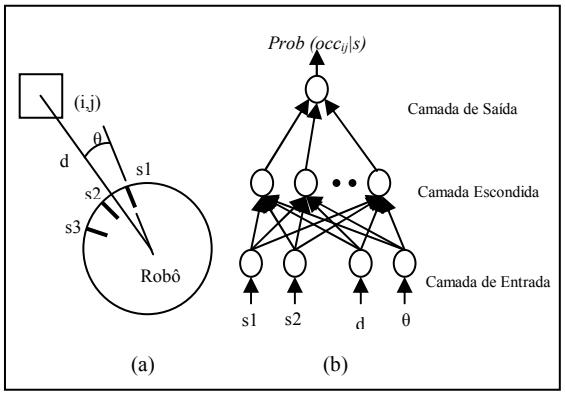

Figura 2. (a) Leituras dos sonares e a distância e o ângulo em relação ao centro do robô, para uma célula (i,j). (b) Rede neural que interpreta as leituras dos sensores como uma probabilidade de ocupação.

O robô utiliza a rede neural N enquanto acompanha os limites do obstáculo. A partir disso, as leituras consecutivas das interpretações neurais dos sensores podem ser integradas no tempo com o objetivo de melhorar a confiabilidade do *grid* de ocupação G. Assim, sejam M leituras consecutivas dos sensores $s^1, s^2, \cdots, s^M$, a probabilidade de ocupação da célula (i, j) pode ser vista como $P_{ij} = \Pr ob(occ_{ij} \mid s^1, s^2, \cdots, s^M)$. A integração no tempo é obtida aplicando-se a regra de Bayes para estimar esta probabilidade (Thrun, 1998):

$$P_{ij} = 1 - \left(1 + \prod_{m=1}^{M} \frac{\Pr ob(occ_{ij} \mid s^m)}{1 - \Pr ob(occ_{ij} \mid s^m)} \right)^{-1}$$

A interpretação neuronal pode compensar eventuais erros devido à reflexão de sinal do sonar, e a integração no tempo de interpretações consecutivas produz uma reconstrução aceitável, apesar da limitada informação sensorial produzida pelos sonares.

2.2 Identificação dos limites do obstáculo

A partir do *grid* de ocupação local obtido pela interpretação neuronal dos sensores, os limites do obstáculo são aproximados para uma linha reta que será utilizada para construir as partições P.

Um limite consiste de células (i, j) com $\Pr ob(occ_{ij}) > 1 - \varepsilon$ e que estão mais próximas do robô. O parâmetro ε permite definir um limiar acima do qual as leituras de probabilidade de ocupação da célula obtidas pela rede neuronal podem ser entendidas como indicando que ela está ocupada.

Para calcular a linha reta que define o limite do obstáculo utilizou-se uma versão do método X^2 (Press et al., 1992).

Seja uma janela M dentro do *grid* G indicando a presença do limite de um obstáculo. Cada célula dentro desta janela pode ser vista como um ponto (i, j) $(i = 1 \cdots M_i, j = 1 \cdots M_j)$ que possui coordenada em relação ao centro da célula e probabilidade de ocupação específicas. Considere, para cada $i_m \le M_i$, o valor mínimo de i_m que mantém verdadeira a sentença $\Pr ob(occ_{i_m j_m}) > 1 - \varepsilon$. Isto produz um conjunto de pontos (i_m, j_m). Utiliza-se o método X^2 para calcular a melhor linha reta $y(x) = y(x; a, b) = a + bx$ que aproxima este conjunto de pontos (Fig. 4).

A seguir, o robô adota a seguinte estratégia. Verifica-se, para cada nova linha reta l_1 encontrada ao se modelar um novo obstáculo, a existência de uma linha reta l_2 previamente modelada que pode ser alinhada com ela. Caso exista esta linha e a distância ortogonal d entre l_1 e l_2 seja menor que um limiar d_ε, então o robô faz o alinhamento da linha l_1 com a linha l_2. Isto permite manter tão baixa quanto possível a complexidade do conjunto de partições P, além de criar mapas ajustados às regularidades estruturais do ambiente (duas paredes alinhadas segurando uma porta, por exemplo).

2.3 Atualização das partições

Para cada novo obstáculo encontrado pelo robô,é realizada uma aproximação dos cantos do obstáculo desconhecido como linhas retas. Uma vez modelado todo o perímetro do obstáculo, a resolução das partições é aumentada na região que o contém.

Ao se aproximar de um obstáculo desconhecido, o robô se alinha com um dos seus limites. Então, acompanha esse limite e começa a utilizar o *grid* local G e o método X^2. Assim que o robô encontra a linha reta que aproxima este primeiro limite, ele pára de utilizar o *grid* local G, isto é, a interpretação neuronal dos sensores. A seguir, ele se move ao longo da linha até o final do limite, utilizando apenas as leituras brutas dos sensores. Então, o robô gira para se alinhar ao novo limite e começa a modelá-lo seguindo a borda e usando o *grid* local G e o método X^2. Novamente, assim que a linha reta que aproxima o limite é encontrada, o robô pára de utilizar a interpretação neuronal dos sensores. Neste ponto, ele calcula a intersecção entre a linha reta atual e a anterior para identificar o canto visitado. A seguir, ele repete o processo de seguir este limite até o final, girar, e modelar o novo canto. Ele continua repetindo este processo até que todo o perímetro do obstáculo tenha sido percorrido. A condição de parada é reconhecida levando em conta as duas linhas retas que modelam o primeiro e segundo limites do obstáculo. Ao encontrar uma linha, ele verifica se é a linha que modela a primeira borda do obstáculo. Se for o caso, continua a modelar a próxima borda. Se ele encontrar a linha que aproxima a segunda borda do obstáculo, ele considera o obstáculo inteiro como modelado.

Uma vez que todos os cantos de um obstáculo tenham sido memorizados, é possível aumentar a resolução das partições P para modelar o novo obstáculo. Cada novo canto é conectado à borda perpendicular mais próxima de uma das partições existentes, criando assim partições retangulares. Pode acontecer que o aumento na resolução de P produza redundâncias. Duas partições adjacentes são consideradas redundantes se ambas representarem obstáculo ou espaço livre e elas puderam ser fundidas para produzir uma partição retangular. Após atualizar P, remove-se as partições redundantes de todo conjunto de partições P.

Após a atualização das partições P, o robô armazena o conhecimento a respeito das transições físicas entre as novas partições. Em particular, mantém-se um banco de dados D de experiências negativas memorizando as transições proibidas entre as novas partições contendo o obstáculo e as partições adjacentes. Uma experiência negativa entre duas partições α e β é armazenada na forma de uma tripla na forma $(\alpha, \beta, falso)$. O banco de dados D é a memória de longo prazo do robô a respeito do relacionamento espacial existente entre as partições.

2.4 Exploração do ambiente

No algoritmo (Fig. 1), o robô está sempre explorando o ambiente para melhorar o seu mapa. A partir do conjunto de partições P atualmente modeladas, a região menos conhecida do ambiente é escolhida como alvo pelo módulo de exploração. O mapa atual

do ambiente é aperfeiçoado pela exploração contínua do ambiente pelo robô.

O robô inicia o processo de exploração do ambiente escolhendo uma trajetória aleatória e seguindo em linha reta, uma vez que inicialmente o ambiente é desconhecido e P é vazio. Quando P não for vazio, a exploração é feita selecionando-se uma partição alvo, movendo o robô através do ambiente para alcançá-la, e fazendo a exploração desta partição exaustivamente. A exploração é interrompida quando um objeto desconhecido é detectado, para que seja realizada a sua modelagem. Somente a seguir a exploração é retomada, através da escolha de nova partição alvo.

A exploração é baseada em uma estimativa de valor de utilidade de exploração U_e, uma função heurística de valor real que mede o quanto uma partição merece ser explorada, calculada para cada partição. Um contador $c(p)$ mantém a informação do número de visitações para cada partição p. Para considerar quando a partição foi visitada, o contador é multiplicado por um fator de deterioração $\lambda \leq 1$. Assim, o valor de utilidade é maior para as partições que foram visitadas menos vezes e há mais tempo. Assim, o explorador atualiza a função de utilidade como segue:

$$c(p) = \lambda \cdot c(p) \quad \forall p \in P$$

$$U_e(p) = \begin{cases} \dfrac{1}{c(p)} & c(p) > 0 \\ K & c(p) = 0 \end{cases}$$

onde $K > 1$ é uma constante.

Finalmente, o explorador escolhe como alvo a partição que maximiza U_e:

$$alvo = \arg\max_{p \in P} U_e(p)$$

2.5 Planejamento e ação

Para atingir a partição alvo escolhida pelo explorador, há um planejador que calcula o caminho através das partições. Este planejador deriva um grafo topológico a partir do conjunto de partições P atuais. Neste grafo, os nós correspondem às partições e os arcos são obtidos a partir da memória de longo prazo D. O nó correspondente à partição α está conectado ao nó correspondente à partição adjacente β se $(\alpha, \beta, falso) \notin D$.

Partindo do nó correspondente à partição atual, o planejador procura no grafo o menor caminho que leva ao nó associado ao alvo. A partir daí, controladores de baixo nível conduzem o robô até lá. Todas as trajetórias seguidas pelo robô devem ser retas paralelas aos eixos x e y do ambiente.

2.6 Controle de baixo nível

Um módulo reativo de baixo nível controla o deslocamento do robô tratando pequenas inconsistências do mapa (detalhes finos não modelados) e possíveis obstáculos móveis (pessoas, por exemplo). Este módulo também é utilizado para fazer o robô seguir os limites de novos obstáculos a serem modelados.

3 Resultados Experimentais

O Magellan Pro (Fig. 3) tem 40,6 cm de diâmetro e 25,4 cm de altura e possui 16 sonares, distribuídos de maneira simétrica, 8 na parte frontal e 8 na parte traseira, o que lhe permite uma percepção de 360º do ambiente. O robô é também dotado de dois motores de passo que permitem movê-lo para frente, para trás e girar. A determinação de posição e orientação é feita por odometria.

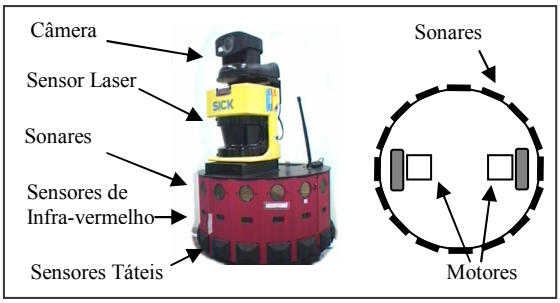

Figura 3. Sensores do Robô Magellan Pro e visão esquemática. Só os sonares foram utilizados para os experimentos descritos nesse artigo.

Para a interpretação neuronal dos sensores, utilizou-se um *grid* local de 14 x 14 células, cada célula cobrindo uma área de 10 x 10 cm. O robô ocupa as 4 x 4 células centrais, como mostra a Fig. 4. Note que, no exemplo da figura, apesar do obstáculo ser uma parede localizada ao lado direito do robô, a lateral do *grid* não está totalmente preenchida, indicando que houve reflexão do sinal de alguns dos sonares. Mesmo com esse problema, o método conseguiu aproximar uma linha reta modelando o obstáculo com boa precisão.

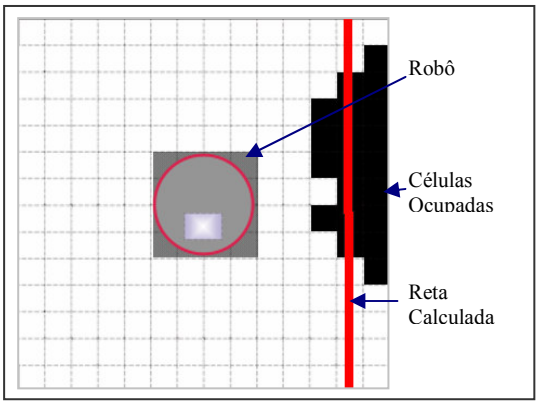

Figura 4. Exemplo de *Grid* de ocupação, após interpretação neural, integração no tempo e aproximação por reta.

Os experimentos foram conduzidos em uma sala fechada preparada com obstáculos (Fig. 5). A região da sala mede aproximadamente 3 x 3,5 m.

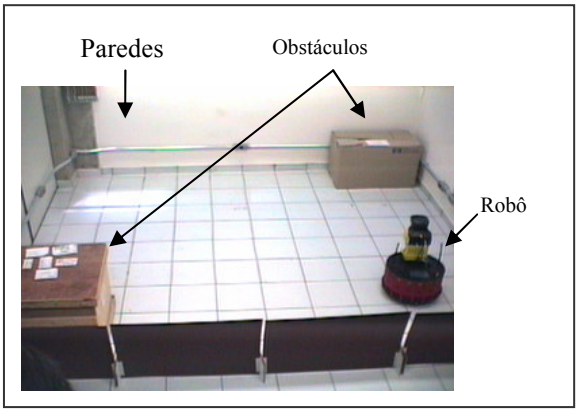

Figura 5. Ambiente para experimentos com robô Magellan Pro.

Adotou-se um procedimento de calibração para diminuir os erros na leitura de odometria. Verificou-se que ao seguir linhas retas o erro era pequeno, porém, ao realizar giros de 90 graus, o erro existente entre o ângulo calculado por odometria e o efetivamente obtido pelo movimento do robô era considerável. Este erro, cumulativo, afetava a trajetória do robô, que deveria ser sempre ortogonal aos eixos do ambiente. Realizou-se uma série de testes em que o robô deveria realizar giros de 90 graus. Observou-se o erro entre o ângulo lido por odometria e o ângulo real. Então, calculando a média de todos os n testes realizados temos uma estimativa da incerteza esperada do ângulo: $e = (1/n)\sum_{i=1}^{n} e_i$. Este fator foi utilizado para compensar o ângulo observado no giro do robô no programa de controle de baixo nível e no calculo baseado em odometria.

Para avaliar o desempenho do método na construção de mapas, experimentos foram realizados dispondo obstáculos em diferentes configurações. Como a resolução das partições varia em função da complexidade do ambiente, variou-se esta última com o objetivo de obter particionamentos distintos. Os resultados são apresentados a seguir

Primeiramente, modelou-se a sala sem obstáculos (mapa 1, Fig. 6). Depois, com apenas um obstáculo lateral (mapa 2, Fig. 7). A seguir, com dois obstáculos nas extremidades da sala (mapa 3, Fig. 8), e, finalmente, com um obstáculo no centro da sala (mapa 4, Fig. 9). Nos gráficos, os obstáculos reais são as linhas grossas e áreas escuras, as partições obtidas são as linhas finas, e a área em cinza é a área mal classificada. O grafo topológico correspondente foi desenhado sobre o mapa.

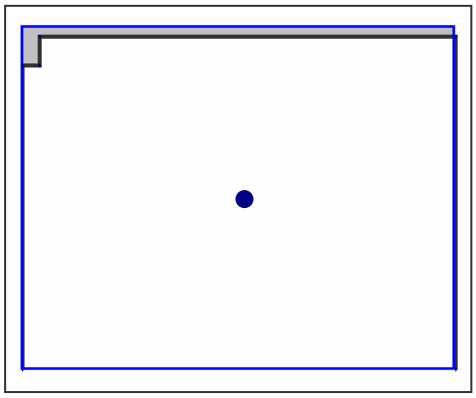

Figura 6. Mapa 1 – Sala vazia.

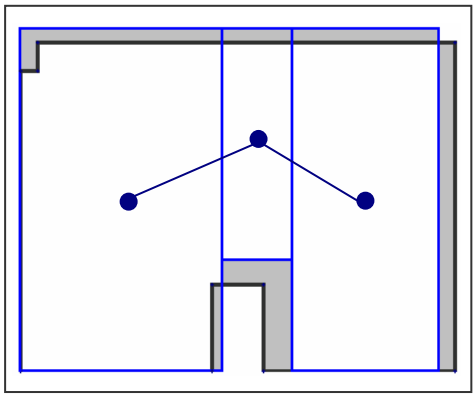

Figura 7. Mapa 2 – Sala com um obstáculo lateral.

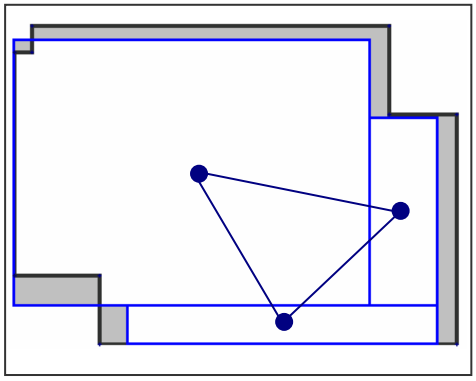

Figura 8. Mapa 3 – Sala com dois obstáculos nas extremidades.

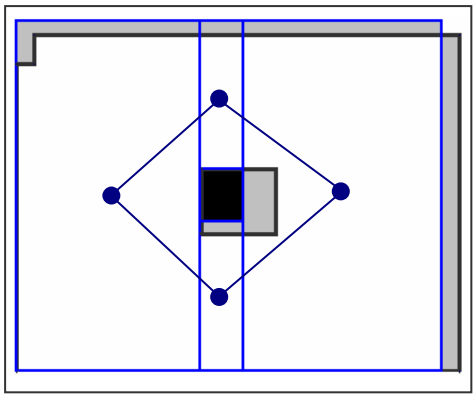

Figura 9. Mapa 4 – Sala com um obstáculo no centro.

Uma forma de medir a precisão do mapa aprendido é calcular a razão entre a área mal classificada e a área

total do mundo (Arleo, Millán e Floreano, 1999). Partições mal classificadas são áreas livres classificadas como ocupadas, ou vice-versa. Seja a soma das superfícies mal classificadas A_e e a superfície total do ambiente A_{tot}. Assim, o erro será $e = A_e / A_{tot}$. Aplicando esta medida, obtemos a Tabela 1.

Tabela 1. Medida quantitativa de precisão dos mapas aprendidos.

Mapa	$Área_{tot}$ (m^2)	$Área_e$ (m^2)	Erro
1	10,21	0,331	0,0324
2	9,89	1,192	0,1205
3	9,35	1,065	0,1139
4	9,88	1,047	0,1060

4 Análise dos Resultados

Analisando os gráficos dos mapas aprendidos, pode-se verificar que o primeiro mapa modelou a sala com uma taxa de erro baixa. Os outros mapas, com obstáculos, apresentaram erro maior, mas mantendo uma média coerente, em torno de 0,1.

Diversos motivos podem estar causando este desempenho. Primeiramente, o tamanho da área modelada é bastante pequeno. Sabe-se que a abordagem não é apropriada para áreas pequenas (Arleo, Millán e Floreano, 1999). Apesar disso, os mapas representam razoavelmente os obstáculos encontrados na sala. Segundo, o sistema de orientação e posicionamento é baseado apenas em leituras de sonares e odometria, o que acarreta grande imprecisão, mesmo se compensada por estratégias adotadas para melhorar as medidas de distância. Finalmente, imprecisões na modelagem das partições representando os obstáculos podem advir da resolução do *grid* local e de imprecisões das linhas retas que aproximam os obstáculos.

5 Conclusão

O algoritmo testado apresenta uma solução para a construção de mapas, possuindo um compromisso entre precisão e eficiência. Utiliza estruturas simples, o que permite sua implementação e uso em um ambiente para construção de mapas em tempo real com os recursos computacionais normalmente disponíveis, conforme atestado pela baixa complexidade do conjunto de partições de resolução variável e pela simplicidade dos grafos topológicos - que permitem um fácil planejamento de caminho. Além disso, a interpretação neural dos sensores e sua integração no tempo são utilizadas por pouco tempo, apenas durante o início da modelagem de uma nova borda de obstáculo, aumentando a eficiência computacional do algoritmo.

Sugere-se como trabalho futuro a repetição da experiência utilizando uma resolução de *grid* local e uma distância de aproximação (parâmetro empírico baseado nas leituras dos sonares) para modelagem de limite de obstáculos diferentes, uma vez que a precisão das partições depende desses dois fatores. Outra sugestão é testar o algoritmo modelando ambientes maiores, o que permitiria observar a provável diminuição do erro das áreas mal modeladas relativo à área total. Como desenvolvimento futuro, serão realizados experimentos com métodos de aprendizagem por reforço na execução de uma tarefa de navegação robótica utilizando como base os mapas cognitivos gerados através do algoritmo testado.

Agradecimentos

Os autores agradecem o apoio da FAPESP (processos 00/06147-3 e 99/05772-2).

Referências Bibliográficas

Arleo, A., Millán, J. R., Floreano, D. (1999). Efficient Learning of Variable-Resolution Cognitive Maps for Autonomous Indoor Navigation. *IEEE Transactions on Robotics and Automation*, v. 15, n. 6, p. 990-1000, Dezembro.

Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3): 249-265, Junho.

Moravec, H. P. (1988). Sensor fusion in certainty grids for mobile robots. *Artificial Intelligence*, v. 9, p. 61-74.

Press, W. H., Teukolosky, S. A., Vetterling, W. T., e Flannery, B. P. (1992). *Numerical recipes in C*. Cambridge: Cambridge Univ. Press.

Thrun, S. (1998). Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, v. 99, p. 21-71.