1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Sistemas Robóticos Autónomos

2021/2022

LabWork #3

Data de Entrega: 01 de Maio 2022

Construção de Mapas usando Modelação Bayesiana em Grelhas de Ocupação – Incorporação na solução de navegação VFF ou VFH –

Introdução

Uma das abordagens utilizadas para a construção de mapas, utilizando sensores de distância, é através dos chamados mapas de ocupação. Estes mapas baseiam-se na discretização regular do espaço de navegação em regiões espaciais, dando origem a matrizes de células, onde cada célula representa uma parcela da área de navegação da plataforma. Os elementos dessa matriz contêm uma estimativa do estado dessa área no que diz respeito à presença ou não de obstáculos. Assim, para cada medida obtida pelo sensor são atualizadas as célula que corresponde á posição estimada do obstáculo, atualizando o nível de confiança das células como estando OCUPADAS vs. VAZIAS.
O conceito de célula de ocupação foi inicialmente proposto por *Alberto Elfes* e desde então tem sido adaptado e melhorado por diversos autores. Neste trabalho, iremos implementar a solução proposta por *Sebastian Thrun* [2], a qual adota uma representação espacial estocástica.

A SOLUÇÃO de S. THRUN

A objetivo de qualquer algoritmo de mapeamento baseado em grelhas de ocupação é calcular a probabilidade *a posteriori* de mapas considerando as leituras realizadas

$$p(m|z_{1:t},x_{1:t})$$

onde, *m* representa o mapa, $z_{1:t}$ o conjunto de observações realizadas até ao instante *t*, e $x_{1:t}$ o percurso efetuado pelo robot, isto é, a sequência de *poses* da plataforma, que na presente abordagem se considera como conhecida.

REPRESENTAÇÃO ESPACIAL ESTOCÁSTICA

O conceito de mapa de grelhas de ocupação particiona o espaço a mapear numa grelha finita de células, tal que $m = \sum_{xy} m_{xy}$.

Cada célula m_{xy} tem associado um valor binário de ocupação que define o estado de ocupação ou vazio da célula.

- **CARACTERIZAÇÃO de cada CÉLULA**
  - xy : Célula de mapa
  - m(xy) : variável de estado associado à célula xy
    - Váriável aleatória discreta, com dois estados:
      - **Ocupado (OCC=1)**
      - **Vazio (EMP=0)**

$$p(m_{xy}) + p(\sim m_{xy}) = 1$$
$$p\left(m_{xy} = OCC\right) + p(m_{xy} = EMP) = 1$$

A abordagem standard das grelhas de ocupação subdivide o problema complexo de estimação do mapa num conjunto de problemas independentes, através da estimação da probabilidade de ocupação individual de cada célula,

$$p\left(m_{xy} | z_{1:t}, x_{1:t}\right)$$

para todas as células m_{xy} que constituem o mapa. Cada um destes problemas de estimação passa a ser um problema de natureza binária com estado estático, podendo ser resolvido através da utilização de um filtro Bayesiano binário. Esta decomposição do problema em sub-problemas independentes é conveniente e simplifica significativamente o problema, mas incorpora alguns problemas ao processo, nomeadamente a impossibilidade de representar dependências entre células vizinhas. Com esta abordagem, a probabilidade *à posterior* sobre mapas é aproximada pelo produto das suas probabilidades *marginais*

$$p\left(m | z_{1:t}, x_{1:t}\right) = \prod_{xy} p\left(m_{xy} | z_{1:t}, x_{1:t}\right)$$

Adoptando a factorização do problema apresentada acima, a estimação da probabilidade de ocupação para cada célula da grelha transforma-se num problema de estimação binário com estado de natureza estático, isto é, o estado de uma célula não se altera função do tempo (não se considera a existência de mapas dinâmicos). A solução para este problema será apresentada com mais detalhe mais à frente, passando o algoritmo de mapeamento a contemplar uma representação de ocupação baseada em *log-odds:*

$$l_t(m_{xy}) = \log \frac{p\left(m_{xy} | z_{1:t}, x_{1:t}\right)}{1 - p\left(m_{xy} | z_{1:t}, x_{1:t}\right)}$$

A vantagem desta representação face à representação probabilística é permitir evitar instabilidades numéricas para probabilidades próximas de zero e de um, podendo as probabilidades ser facilmente recuperadas do rácio *log-odds*, através de

$$p\left(m_{xy} | z_{1:t}, x_{1:t}\right) = 1 - \frac{1}{1 + \exp(l_t(m_{xy}))}$$

FILTRO BAYESIANO BINÁRIO com ESTADO ESTÁTICO

Dada a natureza estática do mapa, a confiança (*belief*) é uma função das observações sendo representada por

$$bel_t\left(m_{xy}\right) = p\left(m_{xy} | z_{1:t}, x_{1:t}\right)$$

onde o estado de uma célula é escolhido de entre dois possíveis valores, representados por

m_{xy} e $\sim m_{xy}$. A confiança para $\sim m_{xy}$ é dada por $bel_t(\sim m_{xy}) = 1 - bel_t(m_{xy})$. A falta de índice temporal no estado m_{xy} reflete o facto do estado não se alterar ao longo do tempo.

Em problemas desta natureza, a confiança é frequentemente implementada através do *log* das *hipóteses* (*log odds ratio*), onde a hipótese de estado de uma célula m_{xy} é definido como o rácio da probabilidade de ocupado dividido pela probabilidade de vazio

$$\frac{p(OCC)}{p(EMP)} = \frac{p(m_{xy})}{p(\sim m_{xy})} = \frac{p(m_{xy})}{1 - p(_{xy})}.$$

O *log odds* é obtido através do logaritmo deste rácio, obtendo-se

$$l(m_{xy}) = \log \frac{p(m_{xy})}{1 - p(m_{xy})}$$

Apresentado a probabilidade valores no intervalo $[0..1]$, a hipótese apresenta valores no intervalo $[0..+\infty]$ e o *log odds* apresenta valores no intervalo $[-\infty..+\infty]$.

Representando a probabilidade de ocupação de uma célula através da equação de Bayes normalizada (uma vez que a trajectória é conhecida, deixamos cair a representação da pose), obtém-se

$$p(m_{xy}|z_{1:t}) = \frac{p(z_t|m_{xy}, z_{1:t-1})p(m_{xy}|z_{1:t-1})}{p(z_t|z_{1:t-1})} = \frac{p(z_t|m_{xy})p(m_{xy}|z_{1:t-1})}{p(z_t|z_{1:t-1})}.$$

Aplicando a regra de Bayes ao modelo da observação $p(z_t|m_{xy})$, obtém-se

$$p(z_t|m_{xy}) = \frac{p(m_{xy}|z_t)p(z_t)}{p(m_{xy})},$$

que substituído na equação da probabilidade de ocupação, dá origem a

$$p(m_{xy}|z_{1:t}) = \frac{p(m_{xy}|z_t)p(z_t)p(m_{xy}|z_{1:t-1})}{p(m_{xy})p(z_t|z_{1:t-1})}.$$

Por analogia, obtém-se para a probabilidade de vazio

$$p(\sim m_{xy}|z_{1:t}) = \frac{p(\sim m_{xy}|z_t)p(z_t)p(\sim m_{xy}|z_{1:t-1})}{p(\sim m_{xy})p(z_t|z_{1:t-1})}$$

Dividindo ambas as probabilidades obtém-se a hipótese de estado, eliminando a necessidade de estimar algumas probabilidades de difícil estimação, resultando

$$\frac{p(m_{xy}|z_{1:t})}{p(\sim m_{xy}|z_{1:t})} = \frac{p(m_{xy}|z_t)}{p(\sim m_{xy}|z_t)} \frac{p(m_{xy}|z_{1:t-1})}{p(\sim m_{xy}|z_{1:t-1})} \frac{p(\sim m_{xy})}{p(m_{xy})}$$

$$= \frac{p(m_{xy}|z_t)}{1 - p(m_{xy}|z_t)} \frac{p(m_{xy}|z_{1:t-1})}{1 - p(m_{xy}|z_{1:t-1})} \frac{1 - p(m_{xy})}{p(m_{xy})}$$

Representando o *log odds* por $l_t(m_{xy})$, obtém-se após o *log odds* da confiança para o instante *t* calculando o logaritmo de ambos os termos da equação anterior, resultando

$$l_t(m_{xy}) = \log\frac{p(m_{xy}|z_t)}{1-p(m_{xy}|z_t)} + \log\frac{p(m_{xy}|z_{1:t-1})}{1-p(m_{xy}|z_{1:t-1})} + \log\frac{1-p(m_{xy})}{p(m_{xy})}$$

$$= l_{t-1}(m_{xy}) + \log\frac{p(m_{xy}|z_t)}{1-p(m_{xy}|z_t)} - l_0(m_{xy})$$

Nesta formulação, $p(m_{xy})$ representa a probabilidade *a priori* do estado de ocupação da célula m_{xy}. Cada observação e correspondente actualização de estado envolve a adição do conhecimento *a priori* da ocupação da célula (na formulação *log odds*).

$$l_0(m_{xy}) = \log\frac{p(m_{xy}=OCC)}{p(m_{xy}=EMP)} = \log\frac{p(m_{xy})}{1-p(m_{mx})}.$$

O algoritmo de mapeamento usando grelhas de ocupação é apresentado de seguida, envolvendo a modelização inversa do sensor. A função *inverse_sensor_model* implementa o modelo de observação inverso $p(m_{xy}|z_t,x_t)$ na formulação *log odds*

$$inverse\_sensor\_model(m_{xy},x_t,z_t) = \log\frac{p(m_{xy}|z_t,x_t)}{1-p(m_{xy}|z_t,x_t)}$$

```
1:  Algorithm occupancy_grid_mapping({l_{t-1,i}}, x_t, z_t):
2:      for all cells m_i do
3:          if m_i in perceptual field of z_t then
4:              l_{t,i} = l_{t-1,i} + inverse_sensor_model(m_i, x_t, z_t) − l_0
5:          else
6:              l_{t,i} = l_{t-1,i}
7:          endif
8:      endfor
9:      return {l_{t,i}}
```

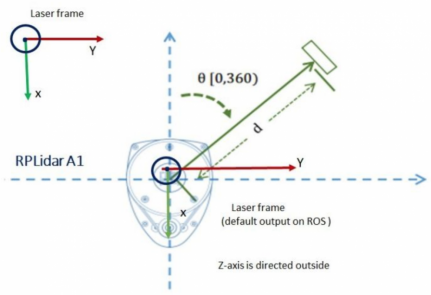
MODELO INVERSO DO SENSOR LIDAR

A obtenção do modelo inverso do sensor é uma tarefa de alguma complexidade que na sua versão mais simples adota uma abordagem Ad-Hoc semelhante à apresentada de seguida:

Algorithm inverse_range_sensor_model(i, x_t, z_t)**:**
  Let x_i, y_i be the center-of-mass of $\mathbf{m}_i$
  $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$
  $\phi = \text{atan2}(y_i - y, x_i - x) - \theta$
  $k = \text{argmin}_j \, |\phi - \theta_{j,\text{sens}}|$
  if $r > \min(z_{\max}, z_t^k + \alpha/2)$ or $|\phi - \theta_{k,\text{sens}}| > \beta/2$ then
    return l_0
  if $z_t^k < z_{\max}$ and $|r - z_{\max}| < \alpha/2$
    return l_{occ}
  if $r \leq z_t^k$
    return l_{free}
  endif

Nesta abordagem, $l_{occ} > l_0$ e $l_{free} < l_0$.

O sensor RPLIDAR é um sensor lidar rotativo, que funciona por triangulação e cuja velocidade de rotação pode ser controlada, influenciando diretamente a resolução angular das leituras. No simulador a desenvolver para o sensor considere que o feixe do LIDAR roda com uma velocidade angular ω fixa, devendo a atualização do mapa de ocupação estar condicionada a essa velocidade de rotação. O sensor RPLIDAR tem associado um sistema de coordenadas como se apresenta na figura,



sendo a informação de leitura fornecida em coordenadas polares, i.e, $[\theta, d]$, tal como representado na figura. Na classe TurtleBot foi fornecida a função (*readLidar*) que devolve as leituras válidas do RPLIDAR no sistema de coordenadas da plataforma. A função *readLidar* devolve as coordenadas (x, y) dos pontos detetados por feixe e o ângulo de orientação do feixe no sistema de coordenadas do RPLIDAR. Para mapear essa informação no sistema de coordenadas do mundo, considere a transformação

$$ {}^{W}_{R}T = \begin{bmatrix} cos\theta & -sin\theta & x - 0.0305 \\ sin\theta & cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} $$

sendo (x, y, θ) a pose atual da plataforma.

Para a concretização desta solução, considere os seguintes valores para o RPLIDAR:

- $z_{max} = 2m$: Alcance máximo da leitura do sensor LIDAR. Poderá ser ajustado para ambientes de maior dimensão. O alcance típico identificado para o sensor é ~3.5m;
- $z_{min} \approx 10cm$: Alcance mínimo do sensor LIDAR. Deverá ser confirmado experimentalmente;

- Resolução do sensor LIDAR:
  - $< 1.0cm, para\ distâncias\ 0.12 < d < 0.5m$;
  - $< 3.5\%\ da\ distância\ medida, todas\ as\ distâncias\ acima\ de\ 0.5m$;
- $\alpha = 5cm$: Espessura de influência do objeto;
- $\beta = 1^o$: Resolução Angular do LIDAR;
- $l_0 = 0.0$: *log ODDS* para o espaço sem influência de leitura do LIDAR ($p\left(m_{xy} \middle| z_t, x_t\right) = 0.5$);
- $l_{occ} = 0.65$: *log ODDS* para o espaço de inferência em ocupação ($p\left(m_{xy} \middle| z_t, x_t\right) = 0.657$);
- $l_{free} = -0.65$: *log ODDS* para o espaço de inferência vazio ($p\left(m_{xy} \middle| z_t, x_t\right) = 0.343$);

Trabalho a desenvolver

Para a realização deste trabalho, é necessário integrar na plataforma TurtleBot alguns dos objetivos dos trabalhos #1 e #2, nomeadamente o controlo de movimento da plataforma e localização odométrica da plataforma.

Integrando na plataforma as funcionalidades desenvolvidas nos anteriores trabalhos, considere agora que a plataforma integra o sensor RPLIDAR cujas especificações acompanham este trabalho.

A informação sensorial de distância fornecida pelo LIDAR deve ser usada para mapeamento do espaço circundante utilizando uma abordagem em grelha de ocupação, a qual deverá ser incorporada nos algoritmos de navegação desenvolvidos no trabalho #2.

Implemente uma grelha de ocupação onde cada célula representa uma área de 5x5cm, considerando um ambiente de navegação com dimensão não superior a 400cmx400cm. Considere que conhece a localização da plataforma em qualquer instante da navegação (através de odometria) e controle o movimento da plataforma usando os algoritmos de movimento desenvolvidos no trabalho 2 (VFF & VFH).

Contrariamente ao considerado no trabalho laboratorial 2, considere agora que a plataforma não possui o conhecimento prévio do mapa do espaço de navegação, devendo a sua construção ser realizada em simultâneo com a navegação da plataforma.

Como validação para a solução desenvolvida considere as seguintes situações:

1. Navegação entre os pontos A-B, sem colisão, utilizando os algoritmos VFF & VFH;
2. Navegação entre os pontos A-B-A, sem colisão, utilizando os algoritmos VFF & VFH;
3. Navegação entre um conjunto de pontos A-B-C-...., sem colisão, utilizando os algoritmos VFF & VFH

Em qualquer das situações anteriormente enumeradas, analise a fiabilidade/qualidade do mapa de ocupação construído durante o movimento da plataforma. Considere diferentes configurações de obstáculos, avaliando o desempenho do algoritmo em ambientes com diferentes graus de dificuldade de navegação.

Caso considere necessário, como solução para identificação das células do mapa de ocupação sob a influência do feixe de um LIDAR, poderá utilizar o algoritmo de *Bresenham* ou o algoritmo *Traversal* proposto por *Amanatidis*, ambos algoritmos amplamente utilizados no desenho de retas em estruturas matriciais, como é o caso da grelha discreta regular usada para mapear a ocupação.

Entrega do Trabalho

O trabalho deverá ser realizado até ao dia 01/05/2022 e deverá ser entregue na plataforma INFORESTUDANTE acompanhado de:

1. Ficheiros da solução desenvolvida, devidamente organizados e comentados.
2. um relatório (em pdf), que deverá conter uma descrição dos diversos passos bem como a discussão dos resultados. Esse relatório não poderá ultrapassar as 8 páginas de texto podendo atingir as 10 páginas se contiver figuras.

Notas: Todos os trabalhos serão testados num espaço de navegação com 400x400 (cm) e que poderá conter um qualquer número de obstáculos dentro dessa região.

Referências

[1] Elfes, A., (1989). "Using occupancy grids for mobile robot perception and navigation," in *Computer*, vol. 22, no. 6, pp. 46-57.
[2] Thrun, S.; Burgard, W.; Fox, D. (2005). Probabilistic Robotics. Cambridge, Mass: MIT Press., Chapter 9.1+9.2
*[3] Bresenham, J. E. (1965). "Algorithm for computer control of a digital plotter". IBM Systems Journal. **4** (1): 25–30.*
[4] Amanatides, J; Woo, A. (1987), "A fast voxel traversal algorithm for ray tracing". Eurographics, 87(3), 1987.