Markov Localization in the CiberRato Simulation Environment

Rafael Morgado - 104277

Universidade de Aveiro, Departamento de Eletrónica, Telecomunicações e Informática
rafa.morgado@ua.pt

1 Introdução

O objetivo deste trabalho é desenvolver um agente robótico capaz de determinar a sua localização num ambiente de labirinto usando o método de Markov Localization. Este método utiliza uma distribuição de probabilidade sobre as células do labirinto para estimar a posição do robô, com base em medições dos sensores de obstáculos.

2 Descrição do Ambiente

O ambiente de simulação utilizado foi o *CiberRato*, configurado com um labirinto de tamanho máximo de 7x14 células. As células são definidas como quadrados de lados iguais com o dobro do diâmetro do robô.

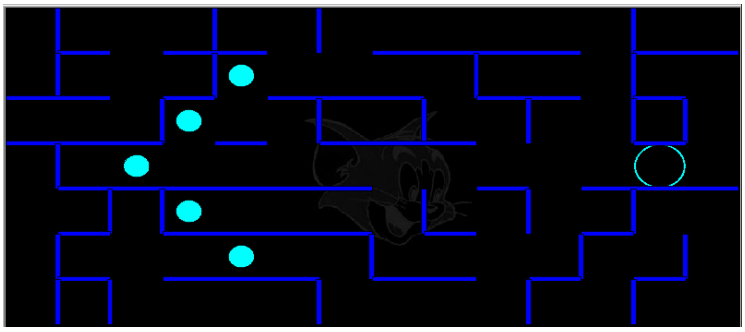


Fig. 1. Exemplo de labirinto no ambiente de simulação CiberRato.

2.1 Configuração do Robô

O robô possui:

- 2 motores (esquerdo e direito).
- 4 sensores de obstáculos, programados para estarem em 0^{o},90^{o},180^{o} e -90^{o}
- 3 LEDs indicativos (visita, retorno e finalização).
- Um sensor de solo e um GPS (não utilizados neste trabalho).

2.2 Modelo de Movimento

O modelo de movimento do robô baseia-se em equações que atualizam a posição e a orientação do robô. Ele utiliza um filtro IIR para suavizar os comandos enviados aos motores e separa o movimento em translação e rotação.

Filtro IIR Os comandos dos motores ($inleft_t$ e $inright_t$) são suavizados utilizando um filtro IIR para modelar as características inerciais dos motores. A potência efetiva aplicada aos motores é dada por:

$$out_t = \frac{in_t + out_{t-1}}{2} \cdot noise$$

Translação O deslocamento linear (lin) do robô é calculado como a média das potências dos motores:

$$lin = \frac{out_{right} + out_{left}}{2}$$

As novas coordenadas (x_t, y_t) são calculadas considerando a orientação (θ_{t-1}):

$$x_t = x_{t-1} + lin \cdot \cos(\theta_{t-1})$$

$$y_t = y_{t-1} + lin \cdot \sin(\theta_{t-1})$$

Rotação A rotação (rot) é determinada pela diferença das potências dos motores, dividida pelo diâmetro do robô ($robotDiam$):

$$rot = \frac{out_{right} - out_{left}}{robotDiam}$$

A nova orientação (θ_t) é então calculada como:

$$\theta_t = \theta_{t-1} + rot$$

Resumo O modelo de movimento combina as componentes de translação e rotação para calcular a nova pose do robô (x_t, y_t, θ_t) no próximo instante.

3 Implementação

A implementação do agente foi realizada em Python, utilizando o ambiente de simulação *CiberRato*. O código desenvolvido foi projetado para localizar o robô no labirinto, atualizando a matriz de probabilidade com base nos sensores de obstáculos. O agente foi ainda implementado numa classe chamada `MyRob`, que herda funcionalidades de `CRobLinkAngs`, permitindo a comunicação com o simulador.

3.1 Comando para Execução

O agente pode ser executado utilizando o seguinte comando:

```
python3 mainRob.py -m ../Labs/PathFinder/pathFinderDefault_lab.xml -p 3
```

Neste comando:

- `-m`: Especifica o arquivo XML do labirinto.
- `-p`: Indica a posição inicial do robô.

3.2 Atualização de Probabilidade

O agente utiliza os dados recolhidos pelos sensores de obstáculos para calcular a probabilidade de estar em cada célula do labirinto. Isso é feito comparando os valores esperados (*expected measures*) com os valores medidos, ajustando a matriz de probabilidade com base num modelo probabilístico Gaussiano.

3.3 Funções Importantes

As funções principais incluem:

- `movement_model`: Calcula a próxima posição do robô com base nos comandos enviados aos motores.
- `sense`: Atualiza a matriz de probabilidade com base nas leituras dos sensores, comparando os valores medidos com os valores esperados para cada célula. Ela utiliza um modelo probabilístico Gaussiano para calcular a correspondência entre as medições e ajusta a probabilidade de cada célula com base na "compatibilidade" das leituras.
- `move`: Atualiza a matriz de probabilidade com base num movimento simples do robô. Ela considera que o robô pode se deslocar para a direita (ou, se estiver na borda, permanece na mesma célula). Essa movimentação é representada na matriz de probabilidade, que reflete a incerteza sobre a posição do robô após o movimento.
- `save_probability_matrix`: Salva a matriz de probabilidade atualizada em um arquivo `localization.out`.
- `calculate_expected_measures`: Calcula os valores esperados dos sensores com base no mapa do labirinto.

3.4 Exemplo de Execução

Durante a execução, o agente realiza as seguintes ações:

- Lê os valores dos sensores de obstáculos.
- Atualiza a matriz de probabilidade e salva no arquivo `localization.out`.
- Movimenta-se para a direita e repete o processo

A Figura 2 apresenta um exemplo de saída visual da matriz de probabilidade no terminal.

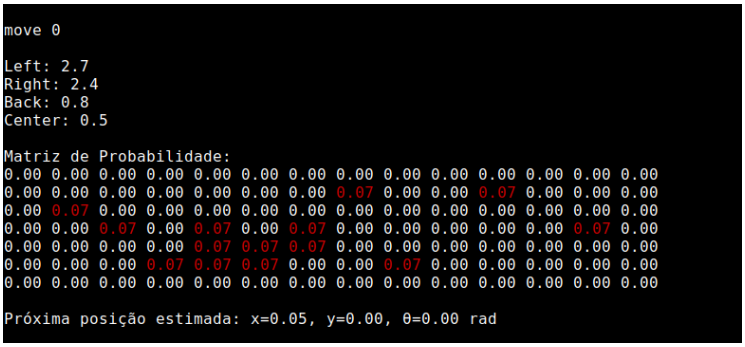


Fig. 2. Exemplo de matriz de probabilidade exibida durante a execução do agente.

4 Resultados

Os testes foram realizados em múltiplos cenários no ambiente *CiberRato*, utilizando não só o labirinto apresentado na Figura 3, como outros. Durante a execução, o agente demonstrou sua capacidade de estimar a sua posição com base nas medições dos sensores de obstáculos.

4.1 Evolução da Matriz de Probabilidade

A seguir, apresentamos três exemplos que ilustram a execução do agente e o comportamento da matriz de probabilidade:

- A Figura 3 mostra o mapa do labirinto utilizado, incluindo a posição inicial, neste caso -p 3.
- A Figura 4 apresenta a matriz de probabilidade inicial, onde todas as células têm uma distribuição uniforme devido à incerteza inicial sobre a posição do robô.
- A Figura 5 mostra a matriz de probabilidade após algumas atualizações, destacando como as leituras dos sensores ajudam o agente a eliminar hipóteses incorretas.
- A Figura 6 apresenta a matriz de probabilidade com toda a probabilidade concentrada na célula correta.

4.2 Modelo de Movimento

O modelo de movimento foi testado para verificar a estimativa de posições após os comandos enviados aos motores. A tabela abaixo apresenta exemplos de posições estimadas em diferentes momentos durante a execução do agente:

Esses resultados mostram que o robô se desloca consistentemente ao longo do eixo x, mantendo a orientação inicial ($\theta = 0.00\,rad$). O deslocamento linear aumenta progressivamente, indicando que o modelo de movimento é consistente com os comandos enviados aos motores.

Table 1. Estimativa da posição do robô com base no modelo de movimento, para -p 3

Iteração	x_t (m)	y_t (m)	θ_t (rad)
1	0.05	0.00	0.00
2	0.12	0.00	0.00
3	0.21	0.00	0.00

4.3 Execução Visual

Os resultados evidenciam a eficácia do modelo probabilístico utilizado, que permite ao agente estimar a posição mesmo em cenários com ruído nos sensores. A atualização contínua da matriz de probabilidade contribuiu para uma localização robusta e precisa.

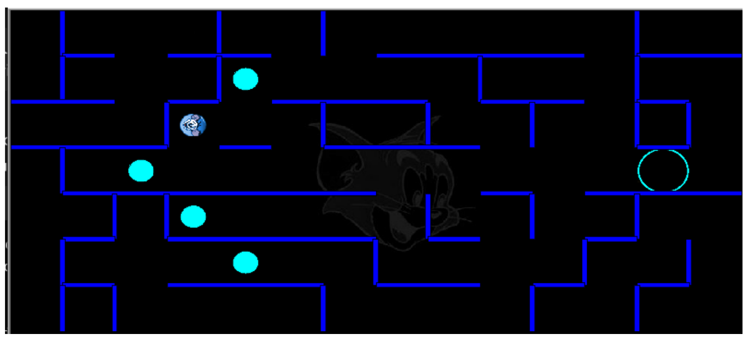
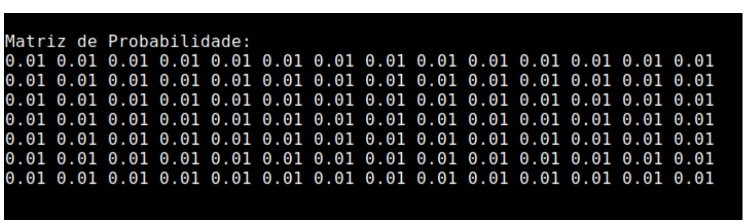


Fig. 3. Mapa do labirinto utilizado nos testes, com as paredes e a posição inicial do robô.



```
Matriz de Probabilidade:
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
```

Fig. 4. Matriz de probabilidade inicial. A incerteza está distribuída uniformemente por todas as células.

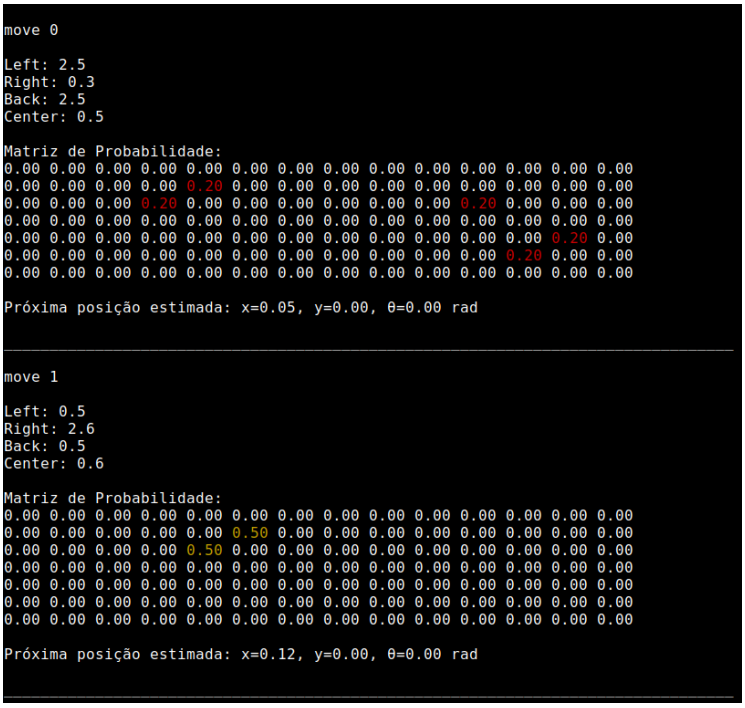


Fig. 5. Matriz de probabilidade atualizada após algumas iterações. As leituras dos sensores começam a eliminar hipóteses incorretas.

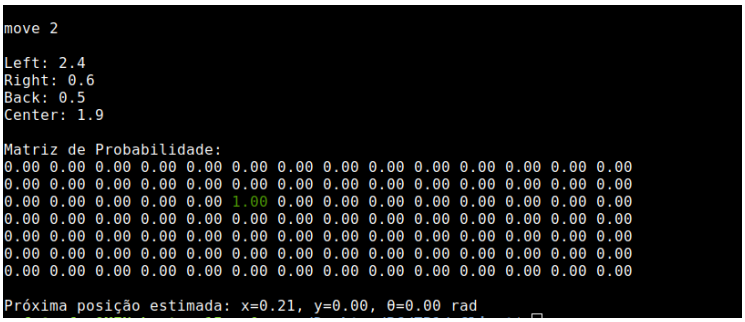


Fig. 6. Matriz de probabilidade final. Toda a probabilidade está concentrada na célula correta, indicando sucesso na localização.

5 Conclusão

O agente desenvolvido demonstrou ser eficaz na localização de sua posição no ambiente de simulação *CiberRato*, utilizando apenas os valores medidos pelos sensores de obstáculos. O modelo de movimento e a matriz de probabilidade apresentaram os resultados pretendidos para os cenários a que foram sujeitos.

Apesar do sucesso, melhorias podem incluir o uso de sensores adicionais e a aplicação do sistema em cenários mais complexos.

Este trabalho serve como base para o desenvolvimento de agentes mais avançados, capazes de operar em ambientes mais desafiadores e com maior robustez.