Robotic challenge solver using the CiberRato simulation environment

André Alves

Universidade de Aveiro
andr.alves@ua.pt
88811

Abstract. Implementação do quarto desafio do CiberRato para a unidade curricular de Robótica Móvel e Inteligente que consiste em usar um simulador de um robô e programá-lo por um mapa com tarefas de localização, mapeamento e planeamento. O robô é equipado com sensores e motores.

Keywords: CiberRato · Robótica Móvel e Inteligente · Challenges · Robótica.

1 Introdução

No âmbito da unidade curricular de Robótica Móvel e Inteligente, foi elaborado um agente que comanda um robô simulado com o objetivo de se localizar, de mapear uma zona desconhecida e de planear o melhor caminho.

2 Desafio

Neste desafio do CiberRato o ambiente de simulação será um robô com:

- 2 motores (esquerdo e direito)
- 3 Leds
- Bússola
- 4 Sensores de obstáculos (cada um colocado a 90° do seguinte cobrindo todos os lados)
- Sensor de colisão
- Sensor do chão (deteta os marcos do mapa)

O sensor de GPS é omitido pelo que não será possível o seu uso para efeitos de localização.

2.1 Localização

Com o objetivo de localizar o robô foram utilizadas as equações do movimento em função da potência usada nos motores. As equações usadas foram:

$$out_t = \frac{in_i + out_{t-1}}{2}$$

O out_t é a potência efetiva aplicada no instante t em que é afetada pela média da potência indicada aos motores (in_i) e a potência efetiva no instante anterior (out_{t-1}). Esta potência é calculada para cada motor.

$$lin = \frac{out_t^r + out_t^l}{2}$$

$$x_t = x_{t-1} \pm lin \quad ou \quad y_t = y_{t-1} \pm lin$$

O lin é a média da potência efetiva calculada para os dois motores no instante t. É adicionada à posição anterior essa potência calculada para obter a posição atual. Como é sempre calculada para um ângulo divisível por 90 (0°, 90°, 180° e 270°) é usada a expressão do x para os ângulos de 0° e 180° e a expressão do y para os ângulos de 90° e 270°(ou -90°). Nos casos do ângulo ser 180° e 270°(ou -90°) em vez do valor da potência ser adicionado, ele é subtraído.

No caso de erros de calculo da posição, o agente tenta corrigir usando o sensor de obstáculos da frente. Quando chega a um ponto de coordenadas ímpares e conseguir detetar uma parede à frente, este usa o valor do sensor mais o raio do robô para conseguir detetar a posição correta no eixo em que ele está orientado.

Movimento Para o calculo da potência a ser indicada para os motores foi usado controlador proporcional integral derivativo (ou controlador pid). O uso do controlador pid permite que a potência usada seja menor quanto mais próxima do objetivo ele estiver.

3 Mapeamento

Tendo acesso à localização calculada do robô, o agente começa a mapear os caminhos livres, paredes e marcos. Sabendo que o mapa tem o tamanho 28 de comprimento por 14 de largura e não sabendo a posição inicial nesse mapa, pode haver no máximo mais 27 unidades de medida na horizontal para um dos lados e 13 na vertical. Por isso assumiu-se que a sua posição inicial seria x:27, y:13 e como cada célula tem 2 unidades de medida de comprimento fez-se o robô movimentar-se 2 unidades de comprimento de cada vez. Sempre que chega a uma posição nova o robô analisa todos os lados adiciona paredes se nos locais onde há paredes e caminho livre onde há caminho livre. Se encontrar um marco sinaliza esse marco no mapa.

As Sinalizações são:

- 'X' para caminho livre;
- '—' ou '-' para paredes verticais ou horizontais (respetivamente);
- 'I' em que I representa um número referente ao marco, sendo o marco 0 a posição inicial do robô;
- ' ' para um local desconhecido.

São utilizadas duas listas e um dicionário, uma lista para guardar as paredes já encontradas, outra para guardar os sítios já visitados e o dicionário para guardar os marcos com as respetivas coordenadas. Além disso, um dicionário é utilizado, onde são colocadas, como chaves, as coordenadas de sítios que o agente já percorreu mas que podiam levar a um caminho novo. Assim, o objeto do dicionário são as coordenadas adjacentes às da sua chave, situando um sítio que ainda não foi visitado. Isto acontece quando o agente chega a uma bifurcação ou entroncamento, e tem mais de um sítio que ainda não visitou, escolhendo assim um para ir e colocando os restantes no dicionário.

Para a movimentação entre dois pontos conhecidos no mapa é usado o algoritmo A*[1] que retorna o caminho para esse ponto.

4 Planeamento

Para o planeamento de um percurso fechado com o custo mínimo que permita a visita de todos os marcos a começar no marco 0 foram feitas permutações[2] de todos os marcos e calculado o percurso mais curto entre eles usando o algoritmo A*. A permutação com menor custo é a usada. No final é criada uma lista com todos os pontos do percurso, pela ordem que o robô teria de seguir.

5 Conclusão

Na maioria dos casos o agente consegue concluir o mapa com sucesso, gerando os ficheiros de mapa e planeamento corretos, no entanto nem sempre isso acontece e tem certas limitações.

Limitação temporal O robô movimenta-se lentamente devido às pausas constantes a cada 2 coordenadas e em certos momentos que demora a ter o ângulo correto para se movimentar. Devido a estas limitações, o robô raramente conclui o mapa nos 5000 ticks, demorando normalmente até 6000 ticks.

Robô preso Em poucas execuções do agente, o robô fica preso num beco sem saída em que fica eternamente a movimentar-se para a frente e trás (sem colidir). Esse erro não foi possível resolver a tempo e costuma ser pouco frequente.

Robô colide Devido a um mau cálculo das coordenadas o robô poderá colidir com uma parede. Este erro é bastante raro, mas quando acontece é geralmente num canto de uma parede, o que torna mais difícil a sua prevenção.

Apesar destas limitações considera-se que o agente cumpre, na maioria, os seus requisitos. No entanto há espaço para melhoria, como o cálculo das coordenadas em que se podia tentar obter melhor o erro e tentar limitar o número de vezes que o robô para numa reta sem obstáculos.

References

1. Alves, André, Pires, Márcia, "Bomberman IIA" *Algoritmo A star*, Outubro 2019, https://github.com/andralves717/trabalho-de-grupo-bomberman-bomberman-iia-88747-88811/blob/master/astar.py.
2. Itertools team, "Permutations". https://docs.python.org/3/library/itertools.html#itertools.permutations.