Trabalho Final - Robótica Móvel

Marcelo dos Santos
Matheus Buratti Zagonel
Raphael Kaviak Machnicki

Agosto de 2021

Resumo

Nesse trabalho desenvolvemos um algoritmo para locomoção de um robô no gazebo. Este robô soluciona o problema do labirinto, por meio de uma política que sempre mantém o robô seguindo uma parede, até achar a saída.

1 Introdução

Achar a saída de um labirinto é um problema bastante comum em computação, mais especificamente na parte de robótica. Existem diferentes tipos de algoritmos para resolver esse problema, como recursivos que realizam uma busca em largura [4], outros que utilizam aprendizado por reforço [5] e também algoritmos que utilizam uma imagem do labirinto e fazem uso de técnicas de processamento de imagem, visão computacional e teoria de grafos para chegar numa solução [3]. O algoritmo que implementamos segue a seguinte política: segue a parede, mantendo-a sempre à sua direita [4].

Esse algoritmo é conhecido como o método da mão direita. A ideia é sempre manter a mão direita em uma parede e com certeza depois de um tempo será possível encontrar a saída. Podemos facilmente fazer um mapeamento desse método para um algoritmo que mantém o robô sempre seguindo uma parede, até que seja encontrada a saída.

2 Implementação

Para implementar o algoritmo descrito acima, escrevemos um código em python que simula o método da mão direita para a resolução de labirintos. O código faz parte dos ambientes ROS e Gazebo, que permitem o uso do modelo publisher-subscriber, ou seja, os sensores do robô medem os dados do ambiente e os publicam em uma interface, que será acessada pelo nosso algoritmo, que por sua vez tem a função de a partir desses dados, devolver a próxima ação a ser executada pelo robô. Essa ação também é publicada na interface, que é comum ao robô e ao controlador.

O objetivo do codigo é que o robô sempre esteja a uma distancia fixa da parede a direita. Ele faz isso seguindo os seguintes passos: encontrar uma parede, emparelhar o lado direito com a parede, seguir a parede até que encontre uma curva para a direita ou uma parede em sua frente. Se encontrar uma curva direita, ele fará a curva. Se encontrar uma parede a frente, ele irá girar até uma direção que não tenha mais parede na sua frente.

Para isso, divide-se os sensores, que indicam a distância até um obstáculo, em 3 faixas, que representam orientações diferentes: esquerda, frente e direita. Se o menor valor indicado por uma ou mais dessas faixas for menor que uma distância pré-estabelecida, o robô deve tomar uma ação. As possíveis situações que o robô pode se encontrar estão elencadas abaixo, de maneira que a ordem disposta representa a prioridade de ser executada, de forma que a primeira tem a maior prioridade e a terceira, a menor.

- Obtáculo a frente: o agente não tem como seguir em frente, então ele gira para a esquerda até ficar paralelo a uma parede. As figuras 1 e 2 indicam o comportamento do robô ao encontrar uma parede em sua frente.

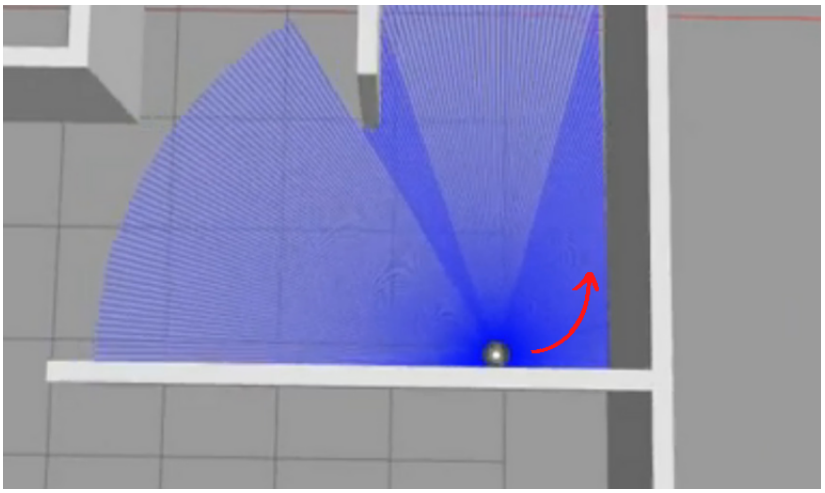

Figura 1: turn_left() Antes

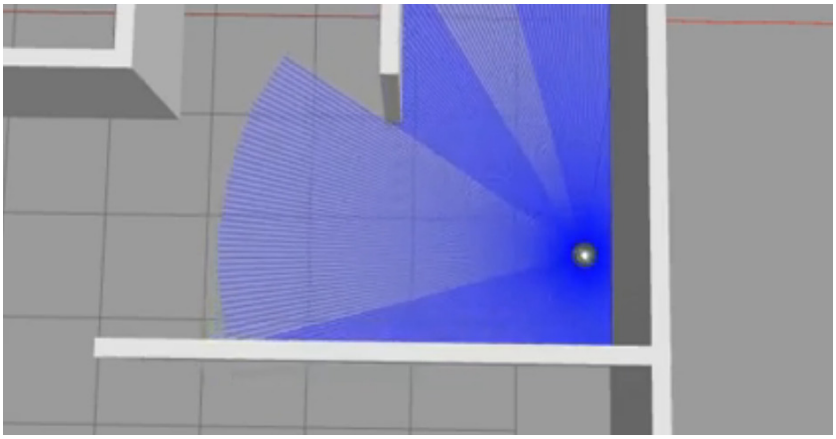

Figura 2: turn_left() Depois

- Nada a direita: o robô precisa achar uma parede. Ele fará um movimento em espiral para a direita até achar uma parede, resolvendo assim o problema de não ter nenhuma parede por perto para seguir. As figuras 3 e 4 indicam o comportamento do robô ao perceber que não há uma parede próximo a ele.

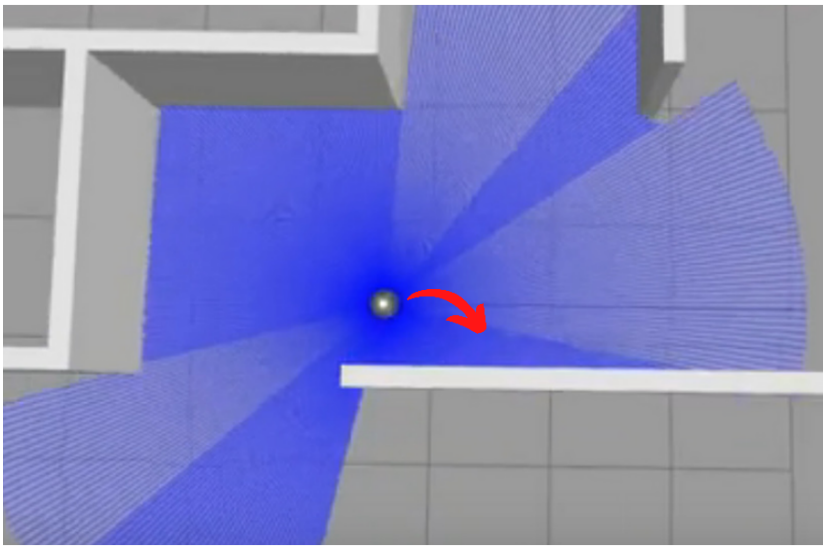

Figura 3: find_wall() Antes

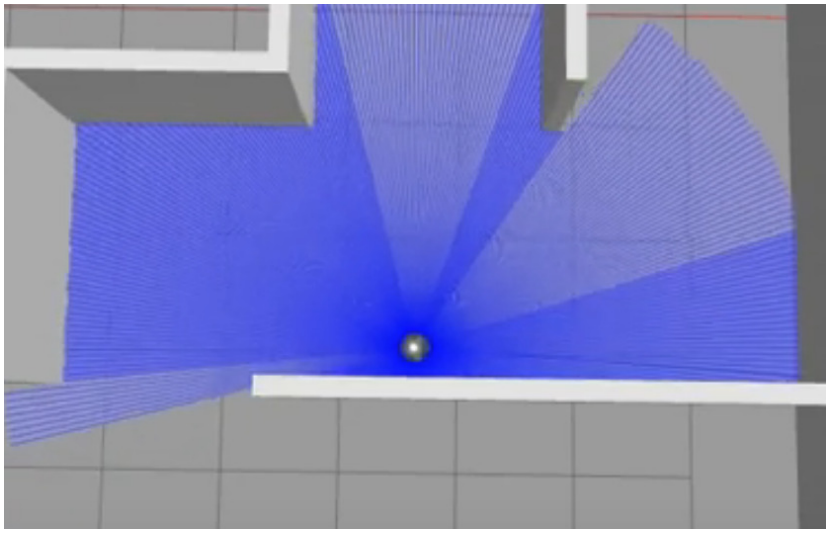

Figura 4: find_wall() Depois

- Parede a direita e nada a frente: o agente segue reto até achar um obstáculo ou a parede acabar. As figuras 5 e 6 indicam o robô seguindo uma parede por toda sua extensão.

Figura 5: follow_wall() Antes

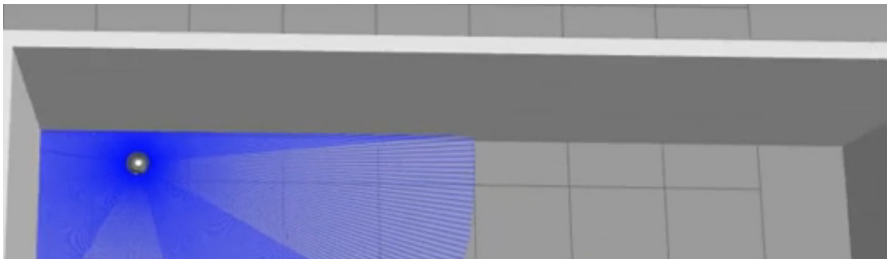

Figura 6: follow_wall() Depois

O programa implementado pode ser obtido em [1] e o vídeo com a demostração pode ser vito em [2].

3 Conclusão

O algoritmo aparenta ser bem sólido, porém, através de diferentes testes, percebemos que diferentes parametros mudam o comportamento do algoritmo de maneira considerável, por exemplo, o tamanho das faixas dos sensores, a distância para reconhecer um obstáculo e o tamanho do robô. O robô que mostrou o melhor desempenho nos testes foi o *burguer*.

Foram realizados testes com diferentes medidas de distância para que o robô consiga detectar um obstáculo. Vimos que, dependendo do tamanho do corredor, isso altera consideravelmente o comportamento esperado do robô. Se o corredor for muito estreito, o robô provavelmente terá dificuldade de locomoção não só devido ao tamanho do robô ser grande comparado ao tamanho do corredor, mas pelo motivo do sensor indicar que existe uma parede próximoa do robô em quase todas as direções. Neste caso o algoritmo pode não funcionar de maneira muito eficiente.

O método da mão direita não é o melhor no quesito de resolver o labirinto. Isso ocorre porque o robô pode andar por caminhos desnecessários até chegar à saída. Além disso, se no labirinto tiver paredes desconexas (não contíguas), o robô pode entrar em *loop*, correndo o risco de não encontrar a saída. Apesar das desvantagens citadas acima, o algoritmo implementado foi escolhido pois é de fácil entendimento, simples de ser implementado, possui baixa complexidade e possui garantia de que encontra a solução para labirintos conectados, sendo portanto uma boa opção de trabalho para quem está tendo o primeiro contato com o ROS2 e o Gazebo.

Referências

[1] `https://github.com/marcelowds/Robotica_tf`, 2021. [Online; accessed 11-August-2021].

[2] Labirinto - ros2 - gazebo. `https://www.youtube.com/watch?v=Rjve-MeL17Q`, 2021. [Online; accessed 11-August-2021].

[3] Mohammad Aqel, Ahmed Issa, Mohammed Khdair, Majde Elhabbash, Mohammed AbuBaker, and Mohammed Massoud. Intelligent maze solving robot based on image processing and graph theory algorithms. pages 48–53, 10 2017.

[4] Bhawna Gupta and Smriti Sehgal. Survey on techniques used in autonomous maze solving robot. In *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, pages 323–328, 2014.

[5] Shih-Wei Lin, Yao-Lin Huang, and Wen-Kuei Hsieh. Solving maze problem with reinforcement learning by a mobile robot. In *2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE)*, pages 215–217, 2019.