Robot Reactivo seguidor de paredes

Uma implementação recorrendo ao meta sistema operativo ROS

Reactive wall follower robot

An implementation using the meta operating system ROS

André Humberto Trigo de Bordalo Morais
Mestrado Integrado em Engenharia Informática e Computação, Faculdade de Engenharia, UP
Porto, Portugal
ei07122@fe.up.pt

***Resumo* — Este documento descreve o processo de implementação de um robot reativo na plataforma ROS com o simulador Gazebo, seus mecanismos de decisão e aquitetura. De seguida são discutidas algumas limitações e apresentados os resultados das simulações concluindo com uma avaliaçao da implementação.**

Palavras Chave - ROS; Gazebo; reativo; arquitetura.

***Abstract* — This document describes the implementation of a reactive robot in the ROS platform combined with the Gazebo simulator, the robot's architecture and decision mechanisms. Following, some limitations of the robot are discussed and the simulation results are presented concluding with an evaluation of the implementation.**

Keywords - ROS; Gazebo; reactive; architecture.

I. INTRODUÇÃO

Na área da róbotica, relativamente à organização da arquitetura do agente, podemos distinguir entre arquiteturas reativas, arquiteturas deliberativas e arquiteturas híbridas.

Neste trabalho foi implementada uma arquitetura reativa recorrendo a uma maquina de estados finita que gere qual dos comportamentos reativos está ativo num dado instante. Cada comportamento possui um conjunto de pares condição-ação hierarquicamente definidos perfazendo uma arquitetura de subsunção.

Apesar de mais simples, as arquiteturas reativas possuem vantagens inerentes, conseguindo superar com sucesso um conjunto de tarefas com relativa facilidade.

Foi escolhido o robot Husky[1] produzido pela empresa Clearpath Robotics[2] que utiliza um mecanismo diferencial de 4 rodas para locomoção. Ao robot foi acopulado um sensor laser para que fosse possível detetar obstáculos.

II. ARQUITETURA

Numa arquitetura reativa um comportamento é o mapeamento de uma ação aos atuadores em função dos dados recebidos pelos sensores. Os comportamentos são geridos usando uma máquina de estados finita, estando cada comportamento associado a um estado.

A. Comportamentos

Foram implementados 3 comportamentos:

- Wall Searching: o robot efetua um trajeto circular, aumentando o raio da curva progressivamente até encontrar um obstáculo, dirigindo-se a ele até se encontrar próximo.
- Wall Following: o robot segue o obstáculo tentando manter uma distância constante enquanto se desloca paralelamente.
- Wall Centering: o robot ajusta-se de forma a terminar numa posição equidistante a três paredes perpendiculares entre si duas a duas (2 paralelas entre si, uma perpendicular a ambas).

III. IMPLEMENTAÇÃO E MECANISMOS DE DECISÃO

Após inicialização do Node, é criado um objeto Publisher responsável por publicar mensagens para o tópico /husky_velocity_controller/cmd_vel, responsável por alterar a velocidade angular e linear na simulação. De seguida é iniciado um loop em que é verificado se o estado actual é o mais recente. Se o estado não estiver atualizado é efetuada uma transição na máquina de estados e iniciado o respetivo comportamento.

Cada comportamento tem associado uma função de callBack que é executada sempre que o sensor publicar uma mensagem para o tópico /scan.

Na função de callBack é activado o mecanismo de decisão associado ao comportamento e é publicada uma mensagem para o tópico que controla a locomoção do robot.

B. *Wall Searching*

O robot inicia o movimento circular até encontrar um obstáculo. Quando o encontrar, move-se até estar próximo, parando e transitando para o próximo estado.

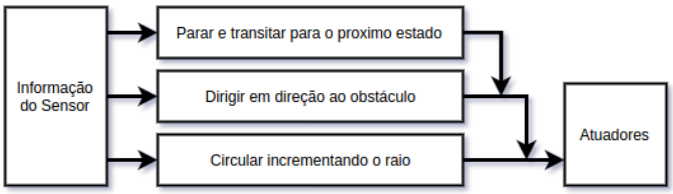

Figure 1. Arquitetura do comportamento Wall Searching

C. *Wall Following*

O robot aproxima-se do obstáculo reduzindo a velocidade para 50%, caso se aproxime mais do que uma distância de segurança, inverte o sentido de marcha para se distanciar um pouco do obstáculo.

No caso de a distância ao obstáculo se tornar maior do que o dobro da distância pretendida, a velocidade linear é reduzida para metade permitindo o robot aproximar-se mais devagar da trajetória pretendida. Se a diferença entre o ângulo pretendido e o ângulo do vetor direção for relativamente grande, a velocidade é reduzida para 40% pelo mesmo motivo.

Não se verificando algum dos casos já descritos, o robot assume a marcha normal na velocidade máxima configurada.

Em todos os casos descritos a velocidade angular é calculada usando um controlardor PD definido pela equação (1) com P=10 e D=5.

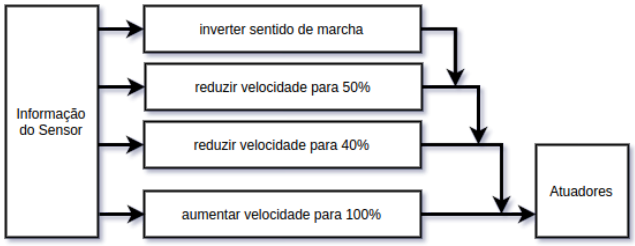

Figure 2. Arquitetura do comportamento Wall Following

$$\omega=(P*err + D*diffErr) * (\alpha - \pi/2) \quad (1)$$

ω: Velocidade angular a aplicar no ciclo presente.

P: Constante proporcional.

err: Diferença entre a distância mínima e a distância pretendida ao obstáculo no ciclo presente (erro).

D: Constante derivativa.

diffErr: Diferença entre o erro do ciclo presente e o erro do ciclo anterior.

α: Ângulo entre o vetor direção e o vetor de menor comprimento entre o obstáculo e o robot.

D. *Wall Centering*

O Robot deteta a parede oposta e tenta seguir a parede mais proxima a metade da distância da parede oposta, continuando o movimento até encontrar uma parede perpendicular ao seu deslocamento a uma distancia menor ou igual à distancia entre as duas paredes paralelas, terminando a execução.

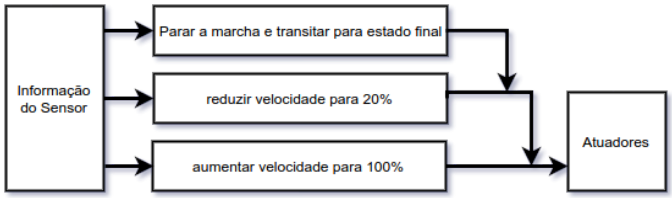

Figure 3. Arquitetura do comportamento Wall Centering

IV. LIMITAÇÕES

O uso de uma arquitetura reativa implica algumas limitações derivadas da sua simplicidade.[3]

E. *Representação Interna do mundo*

A representação do mundo é inexistente, tornando difícil analisar experiências passadas e aprender sobre elas. [4]

F. *Coordenação*

Os comportamentos podem ser difíceis de coordenar, podendo competir em vez de colaborar.

G. *Planeamento*

A ausência de um módulo de planeamento dificulta a execução de tarefas mais complexas.

V. RESULTADOS

Foram efetuadas várias simulações no simulador Gazebo obtendo os resultados apresentados na seguinte tabela:

TABLE I. RESULTADOS DE DIFERENTES SIMULAÇÕES

Mapa	**Cabeçalho de Coluna da Tabela**		
	Posição Inicial	***Posição Final***	***Observações***
Rectangular	Posiçao aleatoria dentro do quadrado	Movimento continuo paralelo à parede mais próxima	É notado um ajuste mais abrupto quando a parede alvo muda
Rectangular	Posição Aleatória for a do Quadrado	Movimento continuo paralelo à parede mais próxima	-
Forma U	Posição Aleatória externa ao U	Posição Interna equidistante de todos os obstáculos	Desvio ligeiro da posição final pretendida
Forma U	Posição Aleatória interna ao U	Posição Interna equidistante de todos os obstáculos	Desvio ligeiro da posição final pretendida

Todo o código desonvolvido para efeitos de teste e simulação pode ser consultado em [5]

VI. CONCLUSÕES

A implementação realizada permite seguir obstáculos contínuos mantendo uma distância fixa em qualquer um dos cenários. Devido à natureza do controlador PD, são verificadas algumas oscilações no percurso necessitando de uma condição de segurança prevendo a colisão eminente do robot com o obstáculo. O processo de centragem do robot na fase final não é muito preciso, obtendo uma posição aproximada da pretendida.

Apesar de ser possível seguir paredes irregulares, a parede deve ser contínua, de forma a que o robot mantenha a marcha paralela. Caso seja verificada uma falha, o robot tentará atravessar a falha de forma a contornar o obstáculo. Este comportamento seria mais simples de evitar com algum mecanismo de planeamento, trocando a arquitetura reativa por uma arquitetura híbrida.

REFERÊNCIAS BIBLIOGRÁFICA

[1] http://wiki.ros.org/Robots/Husky

[2] https://www.clearpathrobotics.com/

[3] D. Nakhaeinia1, S. H. Tang, S. B. Mohd Noor and O. Motlagh "A review of control architectures for autonomous navigation of mobile robots," International Journal of the Physical Sciences Vol. 6(2), pp. 169-174, 18 January, 2011.

[4] L. P. Reis, "Intelligent Robotics" [slides]. Retrieved from https://moodle.up.pt/mod/resource/view.php?id=24314.

[5] https://github.com/lycantropus/reactiveros