Robô Reativo Simples
Simple Reactive Robot

Eduardo Reis, João Duarte
Robótica, Faculdade de Engenharia, Universidade do Porto
Porto, Portugal
up201303832@fe.up.pt, up201303834@fe.up.pt

***Resumo* — Este documento descreve o funcionamento de um robô reativo que procura e segue uma parede de um determinado mapa. Para a implementação do robô foi utilizado a plataforma ROS com o apoio do seu pacote STDR, sendo assim possível simular um simples robô reativo em duas dimensões.**

Palavras Chave - robô; reativo; ros; stdr.

***Abstract* — This document describes how a reactive robot that follows a wall from a certain map works. To implement the robot, we used the platform ROS with the support of the STDR package, that allowed us to simulate a simple reactive robot in two dimensions.**

Keywords - robot; reactive; ros; stdr.

I. Introdução

No seguimento do Tarefa 2 da unidade curricular de Robótica do Mestrado Integrado em Engenharia Informática e Computação, foi implementado um simples robô reativo que segue uma parede, tentando, ao mesmo tempo, manter-se num certo intervalo de distâncias da parede.

Um robô reativo pode ser referido como um robô baseado em comportamento (RBC). Os RBC não têm um modelo interno do ambiente, sendo toda a informação do que os rodeia obtida pelos seus sensores. Em vez de usar cálculos pré-definidos para enfrentar uma situação, os RBC dependem da adaptabilidade.

Os RBC são caracterizados não só por usar uma estrutura baseada em blocos de comportamentos com prioridades distintas, mas também, por não guardarem informação sobre o estado do ambiente, apenas reagindo às informações do presente [1].

Para a implementação do projeto foi utilizada uma estratégia puramente reativa: arquitectura de subsunção, produzida numa plataforma e num simulador especificamente escolhidos para a resolução do problema.

II. Objetivo

O objetivo do projeto é desenvolver um simples robô reativo que, através do uso de sensores, consiga movimentar-se ao longo de uma parede mantendo-se dentro de um certo intervalo de distâncias dessa parede. Neste trabalho espera-se que o referido robô seja capaz de desempenhar a sua função num mapa com o formato de um "D" com outro "D" de menor tamanho no interior do primeiro.

III. Plataforma

Neste projeto foi utilizada a plataforma *Robot Operating System* (ROS) que é uma framework flexível que permite desenvolver software para robôs. É composto por um conjunto de ferramentas, bibliotecas e convenções que facilitam a criação de um comportamento complexo e robusto para um robô em uma grande variedade de plataformas [2].

IV. Simulador

Após uma pesquisa sobre as tecnologias existentes decidimos usar o simulador *Simple Two Dimensional Robot* (STDR).

O simulador STDR é um pacote da plataforma ROS com uma arquitetura simples (Fig. 1) e nada confusa para pessoas que se iniciam em robótica. A sua instalação é rápida e consegue logo emular um simples ambiente 2D que necessita de baixo poder computacional [3][4].

Os pacotes disponíveis do simulador STDR são os seguintes:

- *stdr_server*: Implementa funcionalidade de sincronização e coordenação;
- *stdr_robot*: Fornece ao robô a implementação de sensores;
- *stdr_parser*: Providencia uma biblioteca que faz o *parse* de ficheiros do tipo *yaml* e *xml*;
- *stdr_gui*: Disponibiliza uma interface GUI;

- *stdr_msgs*: Fornece msgs, serviços e ações para o simulador STDR;
- *stdr_launchers*: Facilita a execução de ficheiros;
- *stdr_resources*: Disponibiliza ficheiros de descrição do robô e dos sensores;
- *stdr_samples*: Dispõe de exemplos de código capazes de demonstrar as funcionalidades do simulador.

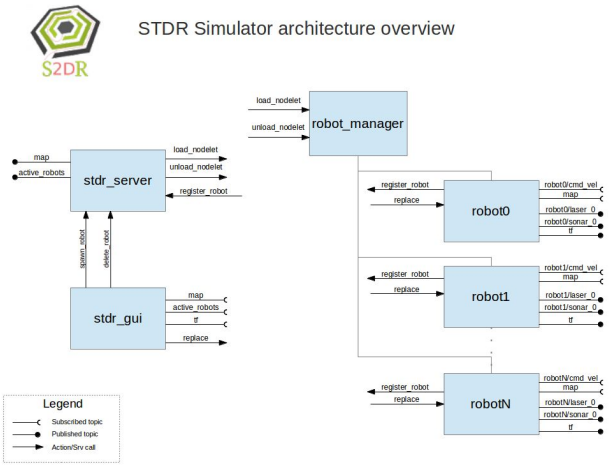

Figure 1. Visão geral da arquitetura do simulador STDR

V. Desenvolvimento

O objetivo do projeto, como já foi referido anteriormente, foi desenvolver um robô capaz de encontrar a parede mais próxima e segui-la infinitamente, mantendo-se a um certo intervalo de distâncias dessa parede.

A. Mapa

O mapa onde se deu o experimento tem a forma de um "D" com outro "D" de menor tamanho no interior do primeiro (Fig. 2). No enunciado da Tarefa 2 era exigido que o robô fosse capaz de seguir, não só, a parede do "D" maior pela parte de dentro, como também, a parede do "D" menor pela parte de fora.

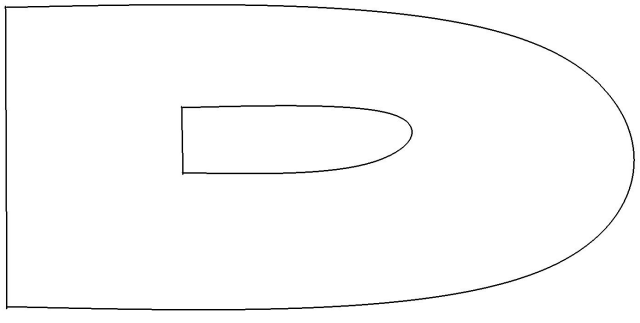

Figure 2. Mapa usado no projeto

O mapa foi desenhado pelo grupo e de seguida inserido no simulador através do pacote *stdr_parser* que permitiu o *parse* do ficheiro *yaml*.

B. Robô e sensores

O robô utilizado no projeto foi o robô *default* do simulador STDR (*robot0*). Este robô tem dois tipos de sensores: os lasers e os sonares. Na nossa implementação foram apenas usados os sonares.

Dos 5 sonares disponíveis no robô apenas foram utilizados 3 deles (Fig. 3): o da frente (*sonar0*), o da direita (*sonar2*) e o da esquerda (*sonar1*).

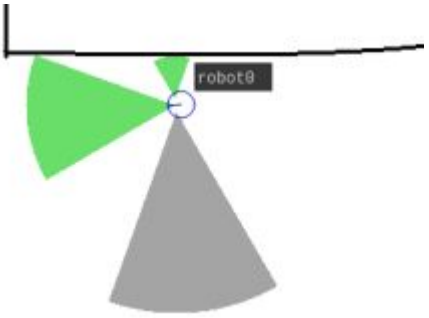

Figure 3. Robô e sonares usados

Os sonares têm um intervalo de alcance que vai dos 0.15 metros até aos 3 metros.

C. Implementação da estratégia

Dividimos a implementação em 3 fases de forma a agilizar, não só o processo de implementação do robô reativo, mas também o processo de compreensão do código elaborado. Primeiro o robô determina a parede mais perto, seguidamente tenta alcançar essa parede e, por fim, segue a mesma infinitamente.

1) Tenta determinar a parede mais perto: Num momento inicial são comparados os 3 valores medidos pelos 3 sonares:

- Caso o menor valor seja medido pelo *sonar1*, o robô roda 90 graus para a esquerda;
- Caso o menor valor seja medido pelo *sonar2*, o robô roda 90 graus para a direita;
- Caso o menor valor seja medido pelo *sonar0* ou nenhum sonar detete uma parede ou todos os valores medidos pelos sonares sejam iguais, o robô não faz qualquer rotação.

2) Move-se até a parede mais perto: Nesta fase o robô apenas desloca-se em linha reta para onde se encontrar virado enquanto a distância à parede escolhida na fase anterior for maior que 0.8 metros.

3) Segue a parede: Ao longo desta fase o robô vai seguir a parede com o seu lado direito orientado para a parede e o lado esquerdo na direção oposta. Posto isto, dividimos a

árvore de decisão em dois níveis: primeiro considera-se o valor do *sonar0* e só de seguida considera-se o valor do *sonar2*. De forma geral, o robô sempre que se aproxima demasiado da parede tenta afastar-se e sempre que se afasta de mais da parede tenta aproximar-se, para tal roda sobre si mesmo e move-se em linha reta a uma velocidade de 0.4 metros por iteração.

VI. Resultados e Limitações

Os resultados esperados eram bastante simples, o robô apenas tinha de desempenhar a tarefa detalhada no tópico Objetivo.

Após a finalização da implementação, o robô mostrou-se capaz de encontrar e seguir as paredes como era exigido, no entanto encontrámos algumas limitações, tanto na forma como a estratégia foi implementada, tanto como no "poder" dos sonares.

Falando primeiro sobre a forma como a estratégia foi implementada, o robô ao ser executado vai detetar qual a parede mais próxima, no entanto, por vezes, a parede mais próxima não está no alcance ou no campo de visão dos sonares, o que leva a este tomar uma decisão errada.

Falando agora sobre o "poder" dos sonares, é muito importante referir que a leitura dos valores a que uma parede se encontra do robô (num estado fixo) não são constantes, variando, por vezes, na casa das décimas de metro. Esta variação leva muitas vezes a uma movimentação menos constante e elegante por parte do robô.

VII. Conclusões

No fim do projeto conseguimos concluir que a implementação de um robô reativo usando ROS e STDR é algo relativamente acessível de realizar. É importante referir que este projeto levou o grupo a aprender a trabalhar na plataforma de um modo mais produtivo e rápido.

Em futuros trabalhos gostaríamos de analisar melhor outros tipos de sensores e compará-los com o sonar, pois a ideia que ficamos por agora é que o sonar não é um sensor preciso nem constante, o que nos leva a não criar muita confiança nos resultados das leituras de distâncias

Referências Bibliográfica

[1] Arkin, R. (1995), Reactive robotic systems. The handbook of brain theory and neural network.

[2] About ROS . Available at: http://www.ros.org/about-ros/ [Accessed 9 Nov. 2017].

[3] STDR Simulator. Available at: http://wiki.ros.org/stdr_simulator [Accessed 9 Nov. 2017].

[4] Tsardoulias, M., Zalidis, C. and Thallas, A. (2014). STDR Simulator. Available at: http://stdr-simulator-ros-pkg.github.io/ [Accessed 9 Nov. 2017].