O Que Pensa Speedy Gonzalez

Jorge Cardoso

***Resumo* – Este artigo descreve o algoritmo usado pelo robô simulado *Speedy Gonzalez,* que participou na competição *Ciber-Rato,* organizada pela Universidade de Aveiro, em 2002.**

***Abstract* – This paper describes the algorithm used by the simulated robot *Speedy Gonzalez,* which participated in the *Ciber-Rato* competition held at the Aveiro University, on 2002.**

I. INTRODUÇÃO

O robô *Speedy Gonzalez* foi desenvolvido por Jorge Cardoso, Nuno Ramos e Pedro Ferreira da Faculdade de Engenharia da Universidade do Porto. O programa foi desenvolvido no âmbito da cadeira de Sistemas Periciais e Robótica do 4º ano do curso de Engenharia Informática e Computação, em 2001.

Ficou em 13º lugar (i.e. último ☹) na 2ª edição do concurso Ciber-Rato organizado pela Universidade de Aveiro em 2002 (ver [2]). O robô venceu a primeira manga mas "encalhou" na segunda e acumulou penalizações que fizeram com que obtivesse aquele lugar. Este artigo descreve sucintamente os processos de raciocínio do agente robótico simulado.

Para mais informação sobre este robô ver [1].

II. PROCESSO DE RACIOCÍNIO

O agente mantém uma pilha (*stack*) de objectivos (direcções) que pretende seguir.

A primeira acção do agente é virar-se para o farol e deslocar-se nessa direcção. Se encontrar um obstáculo no seu caminho, a direcção do farol é guardada na pilha de objectivos e é escolhida uma nova direcção para o robô seguir (algoritmo na secção III). Depois de escolhida a direcção guarda-se a posição relativa do obstáculo em relação ao robô e indica-se numa *flag* que foi encontrado um obstáculo.

A posição relativa do obstáculo (direita ou esquerda em relação ao robô) é usada, enquanto o robô se desloca ao longo desse obstáculo, para determinar qual o sensor que se deve usar para verificar se o robô se está a fastar, ou aproximar, demasiado do obstáculo.

Agora o robô irá seguir ao longo do obstáculo até poder deslocar-se em direcção ao objectivo que está guardado na pilha. Uma vez acabado o obstáculo o robô dobra a esquina, retira o último objectivo da pilha e verifica se existe obstáculo na direcção do próximo objectivo da pilha, ou na direcção do farol, caso não exista nenhum objectivo na pilha. Caso exista obstáculo então, se não houver objectivos na pilha a direcção do farol é colocada na pilha e o robô irá seguir a direcção em que se encontra, ao longo do obstáculo. Se não existir obstáculo então sinaliza-se a *flag* de obstáculo e o robô irá seguir na direcção que havia retirado da pilha.

Se, enquanto está a seguir ao longo de um obstáculo surgir um outro (o robô considera um novo obstáculo a continuação do mesmo com outra direcção ou um obstáculo diferente), a direcção actual é guardada na pilha e é escolhida uma nova direcção.

Se não existirem objectivos na pilha o robô segue na direcção do farol à velocidade máxima, eventualmente corrigindo a direcção devido ao erro dos motores, até chegar ao farol ou encontrar um obstáculo.

O número máximo e objectivos que podem estar na pilha foi limitado para evitar que o robô entre em "ciclo"

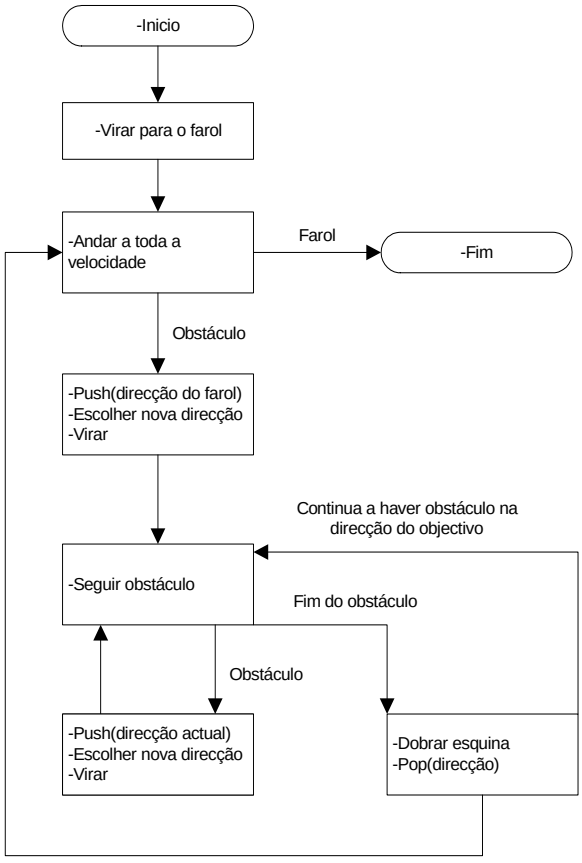

Figura 1 – Diagrama de estados do processo de raciocínio.

devido a uma má decisão. Assim sempre que a pilha estiver cheia e se tentar colocar mais um objectivo, a pilha é esvaziada. Deste modo o robô irá começar de novo a sua tentativa de encontrar o farol. Na Figura 1 é apresentado um diagrama de estados do processo de raciocínio.

III. IMPLEMENTAÇÃO

Os sensores

Na implementação do nosso robô não usamos um dos sensores disponíveis ao agente: o *bumperSensor*, que indica se o robô está a bater nalgum obstáculo. Isto porque achámos que com a correcta utilização dos outros sensores, este não teria utilidade para o algoritmo utilizado.

A escolha de uma nova direcção

Quando o robô encontra um obstáculo à sua frente é necessário decidir se o vai contornar pela esquerda ou pela direita. A escolha é feita do seguinte modo:

- Se o robô não estava a seguir nenhum obstáculo, então escolhe-se o lado indicado como mais "livre" pelos sensores laterais, ou seja, vira para o lado cujo sensor indica um valor menor.
- Se o robô já estava a seguir um obstáculo, então irá escolher o mesmo lado escolhido anteriormente.

Depois de escolher o lado o robô irá rodar enquanto o sensor da frente indicar um obstáculo (indicar um obstáculo é ter um valor superior a uma constante indicada no código do programa).

Uma vez que existem casos em que o robô pode "encalhar" (ver Figura 2) é necessário efectuar mais um passo: comparar a direcção final com a direcção inicial do robô. Se forem muito próximas é provável que o robô esteja "encalhado". Nesse caso dá-se um incremento na direcção a seguir (no nosso caso o incremento era de 45º).

Dobrar uma esquina

Quando acaba um obstáculo é necessário dobrar uma esquina.

Para dobrar uma esquina o robô começa por colocar a potência máxima no motor do lado contrário à esquina e uma potência baixa no outro. À medida que vai progredindo a potência do motor do lado da esquina é aumentada progressivamente. O movimento termina quando for detectado um obstáculo por qualquer um dos sensores, o que no caso geral significa que o sensor do lado da esquina detectou a outra parede.

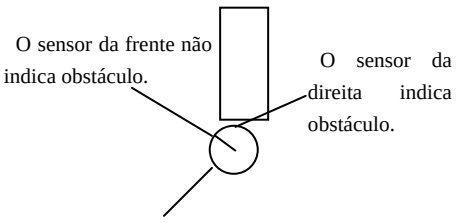

Figura 2 – O robô "encalha".

Velocidade máxima (ou quase...)

Quando o robô segue em direcção ao farol ou quando segue ao longo de um obstáculo tenta seguir à velocidade máxima permitida pelos motores.

Uma vez que os actuadores dos motores estão sujeitos a ruído é necessário efectuar correcções à rota de vez em quando. No caso de o robô seguir em direcção ao farol basta verificar se está afastado da direcção pretendida, dado pelo *beaconSensor*, com base na bússola (*compassSensor*) e rodar um pouco se estiver demasiado afastado. No caso em que o robô está a seguir ao longo de uma parede as correcções à rota actual são feitas com base na distância a que se encontra da parede. Se estiver demasiado afastado ou demasiado perto da parede corrige a direcção dando menos potência a um dos motores dependendo da situação. O robô "sabe" que está muito perto ou afastado da parede com base no sensor do lado da parede. O que define se é perto ou afastado é o valor do sensor (os valores de comparação são constantes definidas no programa e encontradas empiricamente).

CONCLUSÃO

Obviamente, pela classificação obtida no concurso, que existem problemas na implementação do robô utilizado.

O facto de não termos utilizado o sensor de obstáculo, *bumperSensor*, revelou-se fatal uma vez que, no concurso, o nosso robô bateu num outro que já tinha chegado à meta e ali ficou "encalhado" colhendo penalizações que fizeram com que obtivesse no final uma pontuação lastimável...

REFERÊNCIAS

[1] Jorge Cardoso, "*Site* do Speedy Gonzalez", http://nuieee.fe.up.pt/~ei98050/speedy/index.html.

[2] Página do concurso Micro-Rato, http://microrato.ua.pt

06/07/2002 Jorge Cardoso [jorgecardoso@ieee.org]