Robô simulado que encontra espelhos: utilizando esquemas motores

Giovane Roslindo Kuhn (FURB)
brain@netuno.com.br

Categoria: Robótica, Inteligência artificial.

Linguagem de programação: Java.

Sistema operacional: Qualquer um que suporte JRE 1.4.

Palavras-chave: Esquemas motores, inteligência artificial, *TeamBots*, robótica.

1 Contexto

O trabalho tem como objetivo desenvolver um robô que encontre um espelho no ambiente de uma casa, utilizando o simulador TeamBots (http://www.teambots.org). O robô simulado deve ser o *Nomad150* e arquitetura para desenvolvimento do robô deve ser com esquemas motores.

2 Desenvolvimento

Inicialmente foram criados três esquemas motores:

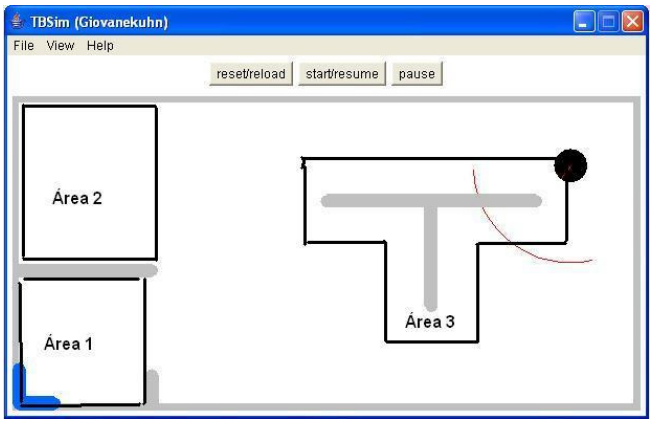

Figura 1

Esquema 1 - Atrator: colocado no objetivo do robô (espelho), desta forma o robô estará sendo atraído pelo objetivo, seu campo de atuação está restrito a **Área 1** da **Figura 1** (vide anexo) e seu fator de influência é o maior de todos os esquemas**.** No restante do ambiente, por estar sendo utilizada a classe *v_LinearAttraction_v* do *TeamBots*, este atrator gera um vetor constante no valor de 1,0.

Esquema 2 - Repulsor: colocado em todas as paredes do ambiente e utilizando a classe *v_Avoid_va* do *TeamBots*, seu campo de atuação é pequeno, porém seu fator de influência é alto para que quando o robô estiver próximo as paredes possa ser "repelido".

Esquema 3 - Noise: esquema utilizado para "desempatar" os dois esquemas descritos acima, seu fator de influência é baixo.

Este conjunto de esquemas não se mostrou eficiente pelos seguintes aspectos:

Problema 1: o vetor constante gerado pelo **Esquema 1** quando o robô está fora da **Área 1** é muito alto, fazendo com que o robô em alguns casos colida contra a parede.

Problema 2: o robô ao entrar na **Área 2** não consegue contornar a parede para entrar na **Área 1,** devido ao forte fator de influência do **Esquema 1.**

Para resolver o **Problema 1** foi alterado o **Esquema 1** criando a classe *MyAttractor*, esta classe tem as mesmas características da *v_LinearAttraction_v* do *TeamBots,* porém quando o robô esta fora da área de atuação do atrator, o vetor gerado é configurável e não mais constante em 1,0.

O **Problema 2** foi resolvido alterando o **Esquema 2**, inicialmente com a classe *v_SwirlLeft_va* do *TeamBots*, desta forma o robô ao se aproximar de uma parede não era mais "repelido" e sim "induzido" para esquerda, com isso o robô ao encontrar uma parede seguia a mesma.

Com esta nova implementação, onde o robô seguia as paredes até achar o espelho, o robô encontrou o espelho em todos os casos exceto quando se aproximava da **Área 3**. Como o robô seguia as paredes, ficava em *loop* contornando as paredes da **Área 3.** Para resolver este novo problema foi criado um sistema de navegação

Para evitar que o robô ficasse em *loop* (andando pelo mesmo caminho), foi criada a classe *MyNavigator*. O simulador *TeamBots* gera um passo do robô a cada 100ms. e a cada passo do robô o sistema de navegação verifica se o robô encontrou uma nova zona (uma área quadrada por onde o robô já caminhou) ou se já não havia passado pela ponto em que se encontra.

Caso o robô caminhe por uma zona que já tenha passado antes em um determinado tempo, ele entra no modo chamado de "**Fuga do loop".** Para isto foi criado um novo esquema (**Esquema 4 – Repulsor**) colocado em todas as paredes do ambiente e quando o robô entra no modo **"Fuga do loop"** a influência do **Esquema 2** é zerada e a do **Esquema 4** é elevada, desta forma o robô pára de seguir as paredes e passa a fugir delas. O robô se mantém neste modo por alguns segundos e depois retorna ao processo inicial.

3 Resultados

Esquemas motores se mostraram bastante eficientes para a resolução deste tipo de problema em ambientes simulados, porém o fato do robô entrar em loop em determinados pontos do ambiente, exigiu uma lógica de navegação para evitar este tipo de acontecimento, com esta implementação o robô encontrou o espelho em todos os casos submetidos e com ponto de saída em vários locais do ambiente.

4 Bibliografia

FERRARI Giulio. **Programming lego mindstorm with java**. Rochland – United States: Syngress Publishing, Inc, 2002.

MURPHY Robin. **Introductions to AI robotics.** Cambridge – United States: Mit, 2000.