R_ZERO: Um Agente Virtual "Reactivo Ma Non Troppo"

Emmanuel Lomba

I.S.E.P. – Instituto Superior de Engenharia do Porto

***Resumo* – Este artigo descreve a implementação de um robot virtual com um comportamento reactivo afinado. Este robot, de nome R_ZERO, foi submetido às provas inerentes ao concurso Ciber-Rato 2002, na Universidade de Aveiro; onde obteve o 3º lugar da competição.**

***Abstract* – This article describes the implementation of a virtual robot with a smooth reactive behavior. This robot, so-called R_ZERO, was submitted to the tests of the Ciber-Rato 2002 contest at the Aveiro University; where it obtained the 3rd place of the competition.**

I. INTRODUÇÃO

R_ZERO, é um agente robótico virtual que tem por objectivo a chegada a um certo ponto de um labirinto (virtual, também…), na menor duração de tempo possível, e incorrendo no menor número de penalizações (estas devem-se a colisões com obstáculos, por exemplo).

De início, este agente não tem conhecimento nenhum do mundo em que vai evoluir, seja das posições dos pontos de partida e de chegada (um farol, virtual), seja dos obstáculos. Estes, podem ser fixos (as paredes do labirinto), ou móveis (os agentes concorrentes, em cada manga da competição).

À semelhança de todos os agentes em prova, R_ZERO está ligado a um servidor, através de um protocolo de comunicação (aqui, UDP). Este distribui pelos agentes os respectivos resultados provenientes de um Simulador.

Este simulador simula o comportamento dos agentes no mundo onde "evoluem". Para além disto, o simulador comunica aos agentes os valores dos respectivos sensores. Com base nestes valores, os agentes (apenas) têm o poder de decidir quais as atitudes a tomar, actuando nos respectivos motores.

Todos os agentes em prova, são idênticos, relativamente ao simulador, a menos dos ruídos gerados e injectados por este nos valores dos sensores dos robots. Ou seja, todos têm as mesmas características técnicas (virtuais). Todos têm, virtualmente, o aspecto que se apresenta na figura 1.

Em suma, R_ZERO é o algoritmo de controlo de uma plataforma robótica, circular e com motorização diferencial. Esta motorização simplifica o modelo do sistema, e permite uma rotação da plataforma sem que haja necessariamente uma translação.

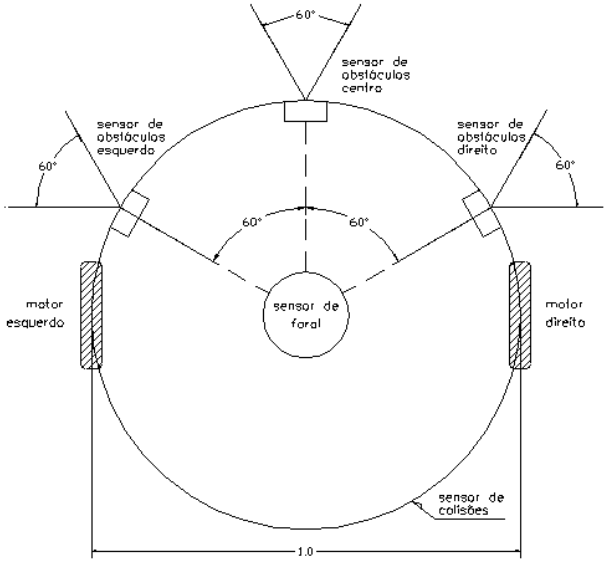

Fig. 1 – Constituição do robot virtual

II. O SIMULADOR [1]

O simulador é responsável por determinar quais os estados de cada agente no labirinto. Entende-se por estados, os valores das seguintes variáveis:

- sensores de proximidade de obstáculos (real);
- sensor de colisões (binário);
- detector de área de chegada (binário);
- detector de orientação, face ao farol (inteiro);
- bússola (inteiro).

Estas variáveis são comunicadas aos respectivos agentes, a intervalos constantes de tempo (80ms).

Para além do cálculo dos valores destes estados para cada robot em prova, o simulador também precisa saber as respectivas posições e orientações de cada agente. Isto, para ser comunicado a uma aplicação de visualização (um Viewer) do labirinto com os robots em prova.

Com base nos estados, os agentes têm de tomar decisões sobre as atitudes a tomar. Isto é, cada agente tem de comunicar ao simulador os valores de Potência a aplicar aos respectivos motores (direito e esquerdo), por forma a obter o movimento/deslocamento desejado do robot, no mundo.

Trata-se então, de um sistema fechado, conforme se ilustra na figura 2.

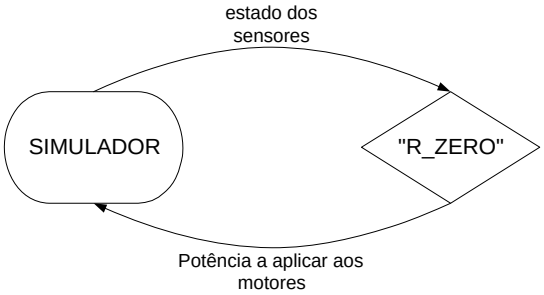

Fig. 2 - Sistema relacional Simulador/Agente

III. O AGENTE

O agente desenvolvido tem como ponto de partida o programa exemplo, fornecido pela organização do concurso Ciber-Rato [2]. Trata-se de um agente (SampleRob), desenvolvido em linguagem C, sob Linux, e com um comportamento reactivo… bastante acentuado.

Comportamento bastante reactivo... Será possível quantificar o quão reactivo é o comportamento de um sistema?

A. Sistema inteligente

Um sistema inteligente é aquele que possua no mínimo uma aptidão à aprendizagem.

Neste caso concreto, do Ciber-Rato, aprender pode ser a capacidade de memorizar um caminho empreendido por forma a poder reconhecê-lo mais tarde. O agente poderá, assim, responder à pergunta: "Será que já passei por aqui?".

Neste contexto, um sistema inteligente deve ser capaz de responder (a qualquer momento) às seguintes questões [3]:

- Onde estou?
- Para onde vou?
- Como vou lá ter?

Estas questões podem por sua vez subdividir-se em perguntas mais elementares que conduzem à elaboração de rotinas dedicadas. Um exemplo será uma rotina que ao longo do percurso do agente, no mundo, determine a posição do farol, relativamente ao ponto de partida, por triangulação. Este processo é útil, no caso de nem sempre se ter o farol à vista...

No entanto, a relativa inteligência que se possa desenvolver para um sistema passa necessariamente por um conhecimento aprofundado do modelo matemático do próprio sistema [4]. Assim, pode ser (por exemplo) elaborado um algoritmo que na decisão possa prever o resultado (simulação in-loco, teste) de uma hipotética acção. Este algoritmo pode assemelhar-se ao de um filtro de Kalman em que haja um processo recursivo de previsão-correcção.

B. Sistema reactivo

Este tipo de sistema é carente de inteligência…

Trata-se de um sistema treinado para desempenhar tarefas pré-definidas mediante o teste de condições de entrada pré-estabelecidas (ou previsíveis). Daqui, pode-se dizer que se trata de uma tipologia mais simples de implementar. Pois exige "apenas" a enumeração dos possíveis estados de entrada, e a elaboração das respectivas acções de resposta para a saída do sistema.

A implementação de um comportamento reactivo em um agente pode ser exemplificado com o excerto abaixo.

```
ENQUANTO (não termina) {
    SE (condição 1) ENTÃO (acção 1)
    SE (condição 2) ENTÃO (acção 2)
    …
    SE (condição n) ENTÃO (acção n)
}
```

Esta abordagem tem obviamente os seus inconvenientes no contexto do Ciber-Rato (e não só!). E a extensão do código a desenvolver pode crescer exponencialmente(?) face ao número possível de estados de entrada.

O resultado de uma implementação deste género está à vista, quando se executam as aplicações relativas ao Ciber-Rato (Server, Viewer, etc…), e se carrega o 'SampleRob' na linha de partida. O referido agente, basicamente, procura atingir a farol, virando-se para este, e evitando os obstáculos. Mas para tal, e devido às constantes em jogo no código fonte, este agente executa um zigue-zague bem pronunciado e "segue" as paredes a uma distância quase igual ao seu próprio diâmetro.

Este agente (o SampleRob) é, portanto, inviável em certas configurações de labirinto. Por exemplo, labirintos com passagens estreitas ou becos com saída estreita são quanto basta para que este agente entre em um ciclo fechado, sem saída.

IV. R_ZERO

R_ZERO é uma evolução de SampleRob; pois como já foi dito, este agente teve por ponto de partida o código fonte de SampleRob.

Assim, R_ZERO continua a ter um comportamento reactivo mas, discreto ou mais disfarçado…

Este "disfarce", é visível ao ver este agente evoluir dentro do labirinto quando se nota que a amplitude dos zigue-zagues diminuiu consideravelmente (comparativamente a SampleRob).

Para chegar a esta relativa linearidade no movimento de R_ZERO é, em primeiro lugar, necessário compreender como funcionam os sensores. Isto é conhecer a resposta destes face às possíveis condições de utilização. Em segundo lugar, é preciso conhecer a resposta do agente perante a ordem de accionamento dos motores. Isto é, em cada CycleTime [1], quantificar o movimento do robot, em termos de translação e rotação, em função dos dados de comando dos motores (lPow e rPow) [2].

Finalmente, após um estudo do código fonte de SampleRob, o enriquecimento deste (segundo o exposto mais adiante…) conduz ao produto que foi designado R_ZERO.

A. Resposta dos sensores

Este ponto apenas trata dos sensores de obstáculos.

A análise de resposta destes sensores deve começar por uma observação destes, em conjunto, no robot. Segundo o que vem estipulado nas Regras e especificações técnicas [1], pode-se desenhar o esquema da figura 3.

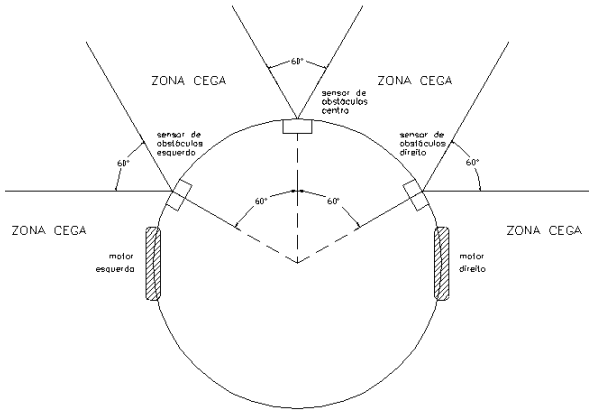

Fig. 3 – Sensores de obstáculos do agente

Pela observação desta figura, depara-se com o facto de que existem três "zonas cegas" (identificadas na figura), a interromper a área de detecção de obstáculos.

Sabe-se (da referência [1]) que estes sensores, a menos do ruído, indicam (cada um) um valor que corresponde à menor distância detectada entre o obstáculo e o próprio sensor.

Então, um primeiro teste a realizar, é colocar o robot junto a uma esquina (por meio do ficheiro MicMouseServer.conf), conforme apresenta a figura 4. E seguidamente, sem que haja movimento do robot, ler os valores retornados pelos sensores.

Este teste pode, e deve, ser repetido com a frente do robot orientada de diversos ângulos relativamente ao cenário (figura 5).

Este processo permite chegar à conclusão de que dois sensores vizinhos podem ajudar a seguir uma parede. Trata-se então de fundir os respectivos valores para se estimar um valor (mais) correcto. Para além disto, também permite estimar qual o intervalo (prático) de valores possíveis, colocando o robot a distâncias conhecidas, da parede (através do ficheiro MicMouseServer.conf).

B. Resposta do agente

A resposta do agente é o deslocamento efectuado pelo mesmo, dados os valores de Potência a aplicar aos

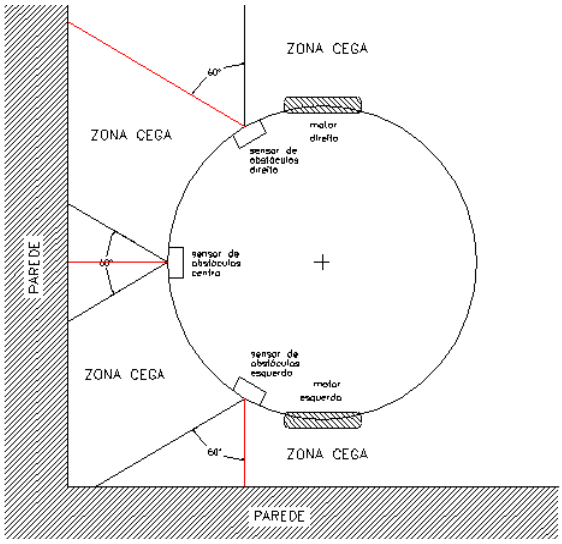

Fig. 4 – Colocação do robot, junto a uma esquina do labirinto.

motores. Esta resposta subdivide-se em duas componentes: uma componente de translação, ou linear, e uma componente de rotação, ou angular.

Estas componentes são determinadas pelas expressões abaixo [1]:

- Componente Linear:

$$Lin = (rPow + lPow) / 2$$

- Componente Angular:

$$Rot = (rPow - lPow) / (2d)$$

onde *d* é o diâmetro do robot.

Os resultados destas operações permitem de avanço, estimar a posição e orientação do robot, no fim do ciclo (de CycleTime = 80ms). Fica-se, antecipadamente, com uma ideia de: "se o robot vai ou não colidir com um obstáculo, no fim do ciclo".

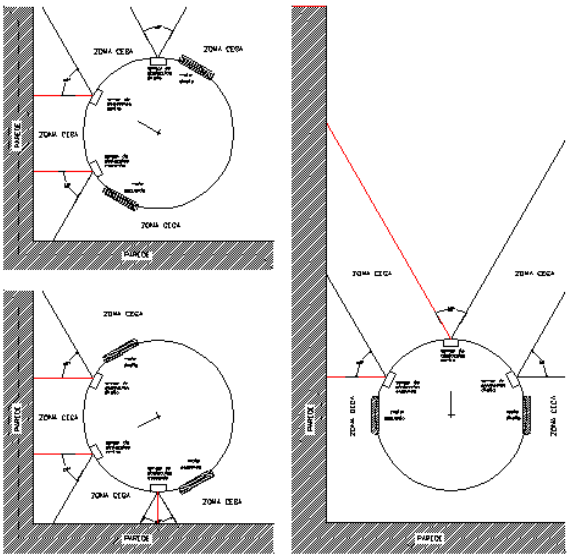

Fig. 5 – Robot em diversas orientações (conhecidas)

C. Filtragem dos valores dos sensores

A filtragem dos valores retornados pelos sensores é um processo fundamental para se convergir para um resultado real.

Este processo é um dos elementos-chave que permite reduzir as oscilações (zigue-zagues) do robot, no seu percurso, quando está seguindo uma parede (por exemplo).

Uma vez que o ruído tem uma distribuição uniforme, para "filtrar" este, R_ZERO assume o valor "corrigido" como sendo o resultado da média algébrica dos 4 últimos valores retornados pelos respectivos sensores.

Obviamente, na fase inicial da "corrida", a quantidade de leituras dos sensores é inferior a 4. No entanto, o resultado é sempre mais aceitável do que sem tratamento algum.

Esta "filtragem" aplica-se apenas aos sensores com resposta não-binária (sensores de obstáculos, sensor de farol e bússola).

D. O algoritmo

Tratando-se apenas do algoritmo de navegação, este segue a seguinte regra: "Ir direito ao farol, desviando dos obstáculos pelo lado mais próximo a este".

Para tal definiram-se ordens de trabalho para R_ZERO, ou "melhor dizendo", definiram-se tarefas. É uma combinação cuidada destas tarefas que permite a R_ZERO atingir o objectivo final.

As tarefas são:

- Segue direcção;
- Contorna pela direita;
- Contorna pela esquerda;
- Fica parado (Pára!).

Para cada uma destas tarefas, são definidos valores mínimos e máximos para variáveis como a velocidade e/ou a distância entre R_ZERO e uma parede que esteja a seguir ou a contornar. Estes valores são as constantes de navegação de R_ZERO. Note-se no entanto que, nesta versão, R_ZERO não distingue as paredes de outros agentes em prova.

R_ZERO, desde o início da prova, adquire os valores retornados pelos sensores filtrando-os. Com estes resultados (desde o primeiro ciclo) orienta-se para o farol e segue a direcção definida por este e pelo próprio.

Quando R_ZERO detecta um obstáculo dentro do seu "espaço mínimo" (uma das variáveis previamente definidas) ou vizinhança, desde logo sabe de que lado este se encontra. Pois o lado do obstáculo corresponde ao sensor cujo valor transpôs um dos limites. R_ZERO "decide" então por que lado vai contornar o obstáculo.

A ordem de contorno mantém-se até que o obstáculo tenha desaparecido de dentro do "espaço mínimo", sendo sucedida pela ordem de ir em direcção ao farol.

Até aqui, o algoritmo de R_ZERO parece muito semelhante ao exemplo original (SampleRob).

A principal diferença é que quando R_ZERO está a contornar um obstáculo (por exemplo, seguir uma parede), ele passa a seguir uma nova direcção, mantendo este rumo até que o obstáculo "desapareça". Isto faz com que o robot não vagueie no seu caminho, não ande aos zigue-zagues, tentando desesperadamente atingir o farol.

D. O treino

R_ZERO, como qualquer outro protótipo, foi submetido a sessões de treino, estas são testes e ensaios para o *debug* do código fonte, bem como para a afinação das constantes de navegação.

Estes treinos têm por objectivo melhorar o comportamento de R_ZERO, dentro do labirinto, e obriga a duas exigências fundamentais:

- Reduzir as oscilações ao seguir um rumo;
- Andar o mais próximo possível das paredes.

Para o acompanhamento da evolução dos treinos, R_ZERO transmite a uma aplicação monitora externa, a correr em paralelo, todos os valores de todas as variáveis, bem como os estados do agente e as "intenções" deste.

A transmissão em causa é realizada através de uma FIFO implementada sob o protocolo IPC system V, em Linux, tal como o próprio agente.

IV. CONCLUSÕES

O pacote de software relativo ao concurso Ciber-Rato [2] revela ser uma poderosa plataforma de simulação robótica, para sistemas simples com objectivos simples.

Esta ferramenta, embora com uma arquitectura sensorial ainda idealista (na versão 0.6), permite testar aplicações de controlo, viáveis para a maioria de pequenos robots móveis autónomos reais (isto é, não virtuais).

R_ZERO [5] é uma das muitas aplicações já desenvolvidas no âmbito do referido concurso. Embora tenha alcançado o terceiro lugar na classificação final, muito trabalho ainda está por desenvolver; nomeadamente dotá-lo de um mínimo de inteligência.

REFERÊNCIAS

[1] "Concurso Micro-Rato 2002 – Regras e especificações técnicas da modalidade Ciber-Rato", http://microrato.ua.pt, Universidade de Aveiro, 08/01/2002.

[2] "MicMouseServer-0.6.tgz", http://microrato.ua.pt/main/docs/tec_docs, Universidade de Aveiro, 11/05/2001.

[3] Johann Borenstein, H.R. Everett, Liqiang Feng, "Navigating Mobile Robots – Systems and techniques", A.K. Peters, Ltd., 1996.

[4] Luís Almeida, "Modelização de pequenos robots autónomos: um exemplo", Revista do DETUA, vol. 2, nº 1, Setembro 1997.

[5] "R_ZERO", ficheiro executável do agente robótico para Ciber-Rato disponível em http://www.dee.isep.ipp.pt:/~emmanuel.