Relatório Laboratório 01 - Máquina de estados Finita e Behavior Tree

Instituto Tecnológico de Aeronáutica – ITA
Inteligência Artificial para Robótica Móvel – CT-213
Nicholas Scharan Cysne
Turma 22

1. Introdução

Neste laboratório foram aplicados dois métodos para modelar o comportamento de um Roomba (robô de limpeza), ver Figura 1. O primeiro utilizando uma Máquina de Estados Finita (FSM) e o segundo Behavior Tree.

Figura 1: Um gato vestido de tubarão pegando carona em um Roomba.

2. Implementação do Comportamento de um Roomba

Para a implementação no código de tal comportamento, seguiu-se o fluxograma de atividades abaixo, Figura 2, que retrata o comportamento do Roomba, que independe do método utilizado, se FSM ou Behavior Tree.

Sucintamente, o Roomba alterna entre quatro estados possíveis: *Move Forward, Move in Spiral, Go Back* e *Rotate*, alternando entre eles por interrupções do sistema. Entre o *Move Forward* e *Move in Spiral* a única condição de alternância entre esses dois é o tempo dispendido em cada ação, passado um determinado tempo em cada situação há uma troca. A ação *Go Back* ocorre sempre que o “bumper” do robô indica contato com um

agente externo, podendo ser antecedida por qualquer outra ação de movimento. Após determinado tempo movendo-se para trás o robô rotaciona em um ângulo aleatório e recomeça o ciclo de movimentos, iniciando com o *Move Forward*.

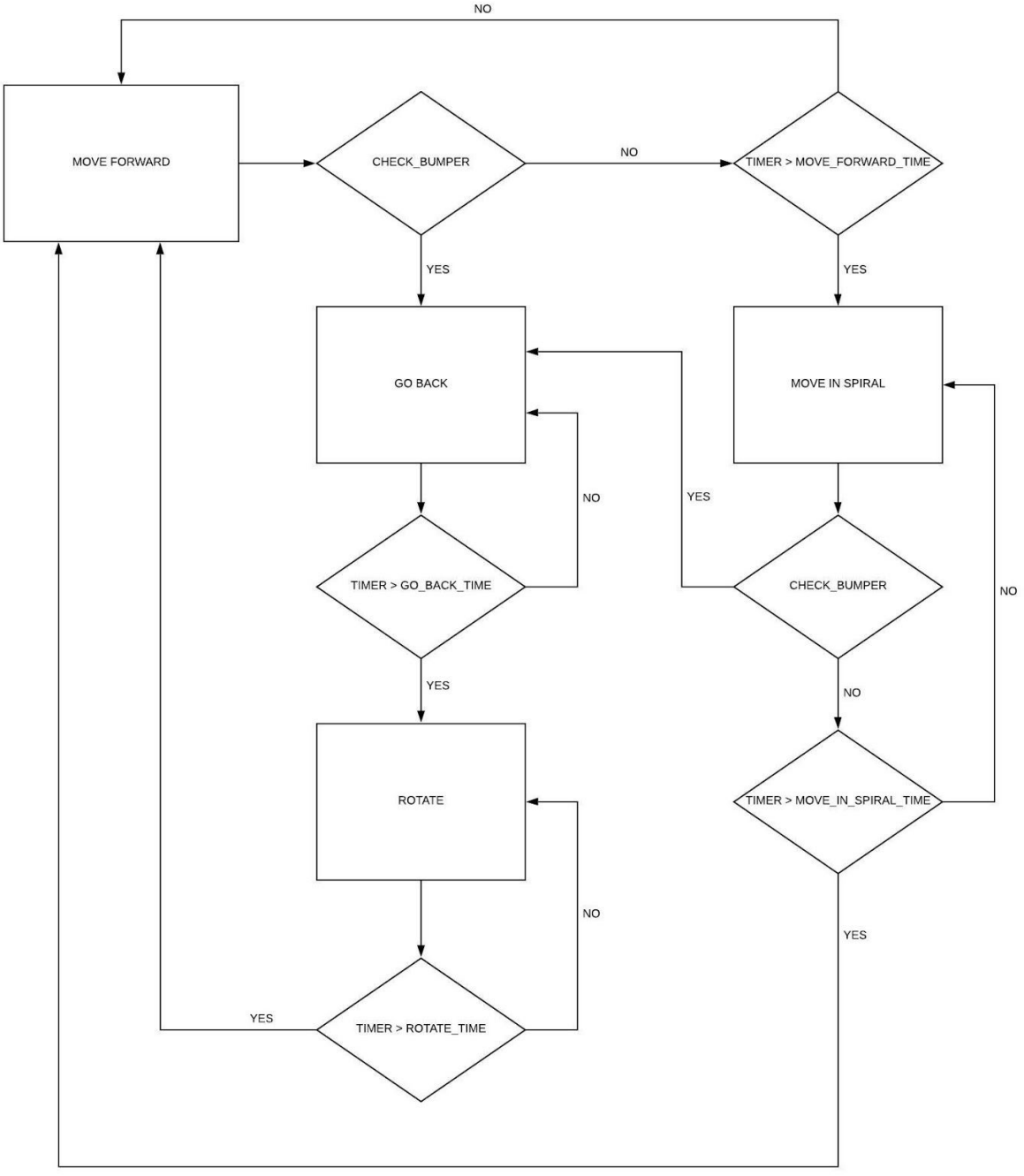

Figura 2: Fluxograma de Ações do Roomba

2.1 Cálculo do ângulo aleatório

A rotação do robô após voltar alguns segundos para trás requer um ângulo de giro aleatório. Para gerar este ângulo aleatório foi utilizado a biblioteca *random* da linguagem python3, devolvendo um número entre os valores [1, 100]. Este número aleatório foi aplicado à função $f(x)$ dada abaixo:

$$f(x) = \frac{\pi(2x - 101)}{99} \qquad \text{Eq. 1}$$

Para valores de x pertencentes ao intervalo [1, 100], a saída da função pertence ao intervalo $[-\pi, \pi]$. Assim, pode-se obter um ângulo aleatório válido para a rotação do Roomba.

2.2 Cálculo do movimento em espiral

Para o movimento em espiral do Roomba, a velocidade linear do robô deve manter-se constante enquanto sua velocidade angular, em torno do próprio eixo, deve variar a uma taxa fixa. Para uma espiral crescente como a desejada esta taxa deve ser um número positivo.

Assim, a cada atualização do behavior do robô, a velocidade angular deste é dada por:

$$\omega = \frac{v}{(r(0) + kt)} \qquad \text{Eq. 2}$$

Onde,

v - Velocidade linear do robô
$r(0)$ - Raio inicial da espira
k - Fator de crescimento do raio da espira
t - Tempo

O tempo no código foi discretizado e este apresentava uma frequência de 60 Hz.

3. Implementação da FSM

Na implementação da máquina de estados finita, foram utilizadas 6 classes representando os estados possíveis do robô, entre essas sendo 4 de ações, *Move Forward, Move in Spiral, Go Back* e *Rotate,* e outras duas para simbolizar a FSM.

Cada estado possui uma função de inicialização, uma função de continuamente checar o estado em que se encontra e prever a necessidade de troca de estado e por fim uma terceira que executava a ação daquele estado atual.

O estado do agente, IA do robô, era determinado pela classe que ele instanciava naquele momento, quando um evento acontecia era esta classe que era substituída, seguindo o raciocínio do fluxograma. A classe do agente apenas executa em seguida as funções de checagem do bumper e execução de ação, o que cada uma destas funções

representa depende do estado do agente, estas também se encarregam da mudança de estado.

Abaixo uma breve descrição da implementação de cada função em cada estado:

- MoveForwardState
  - Init: Aciona o timer de permanência naquele estado.
  - Check Transition: Checa a cada momento se o robô colidiu com algum obstáculo ou se o tempo despendido passou o limite daquele estado.
  - Execute: Atualiza o tempo naquele estado e dá ao robô uma velocidade linear inicial.

- MoveInSpiralState:
  - Init: Aciona o timer de permanência naquele estado.
  - Check Transition: Checa a cada momento se o robô colidiu com algum obstáculo ou se o tempo despendido passou o limite daquele estado.
  - Execute: Atualiza o tempo naquele estado e atualiza também a velocidade angular do robô para a formação da espiral.

- GoBackState:
  - Init: Aciona o timer de permanência naquele estado.
  - Check Transition: Checa a cada momento se o tempo despendido passou o limite daquele estado.
  - Execute: Atualiza o tempo naquele estado e dá ao robô uma velocidade linear inicial.

- RotateState:
  - Init: Aciona o timer de permanência naquele estado, define o ângulo inicial do robô como zero e gera um ângulo aleatório de rotação.
  - Check Transition: Define-se a orientação da velocidade angular baseado no ângulo aleatório gerado, se maior ou menor que zero, então atualiza o ângulo do robô e checa este ângulo até ele chegar no objetivo.
  - Execute: Atualiza o tempo naquele estado e dá ao robô uma velocidade angular constante.

As figuras 3 e 4 representam o funcionamento do Roomba seguindo a FSM, em que se nota a lógica de execução de movimentos deste.

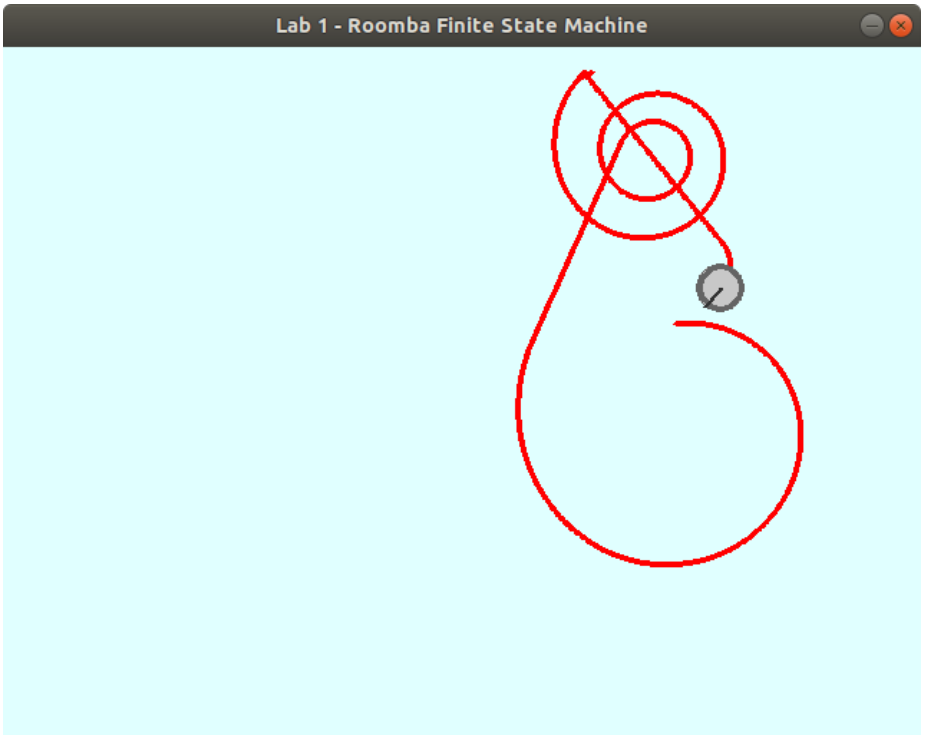

Figura 3: Movimento em espiral seguido de uma colisão de um robô Roomba.

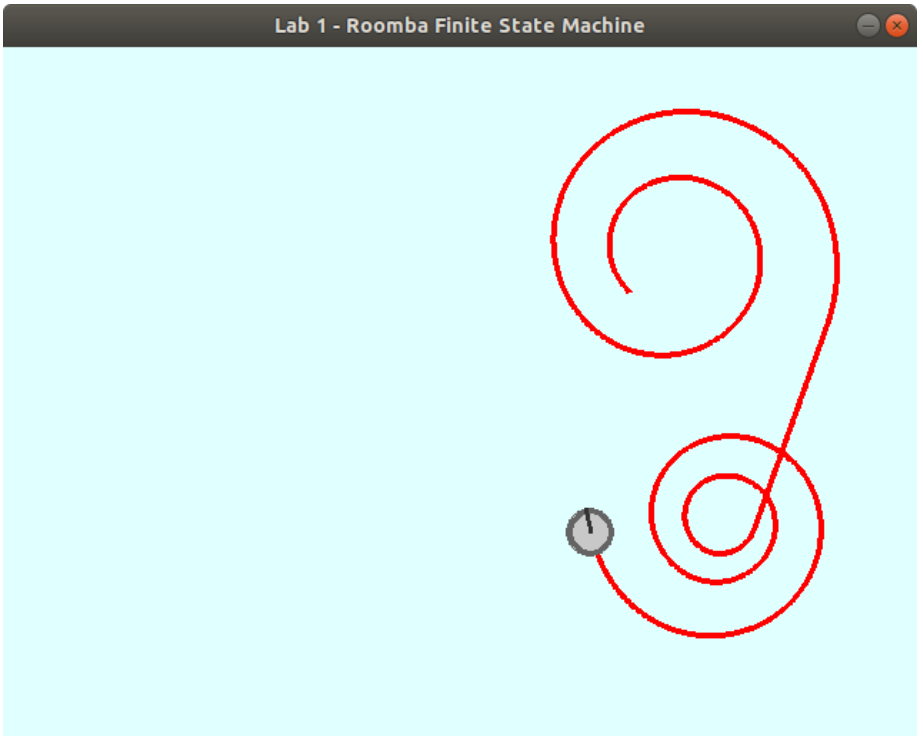

Figura 4: Sequência de movimentos em espiral e em frente.

4. Implementação do Behavior Tree

Na inicialização da Behavior Tree construímos uma árvore em que cada nó age como ação ou como tomada de decisão. Os nós que são internos à árvore são todos tomadas de decisão, *Composite Nodes*, em especial neste laboratório a raiz é do tipo *Selector* e esta possui dois filhos *Sequence*. Os nós externos, ou *Leaf Nodes,* são as ações a serem executadas. Cada nó *Sequence* possui dois filhos *Leaf Nodes* que abrangem as quatro ações possíveis do robô.

Cada nó de ação possui em si duas funções, a primeira de lógica de entrada, ações que ele executa sempre que entra no nó, e a segunda uma função de execução da ação propriamente dita.

Abaixo uma breve descrição da implementação de cada função em cada nó:

- MoveForwardNode
  - Enter: Aciona o timer de permanência naquele estado e dá ao robô uma velocidade linear inicial.
  - Execute: Atualiza o tempo naquele estado, checa a cada momento se o robô colidiu com algum obstáculo ou se o tempo despendido passou o limite daquele estado.

- MoveInSpiralNode:
  - Enter: Aciona o timer de permanência naquele estado e dá ao robô uma velocidade linear inicial.
  - Execute: Atualiza o tempo naquele estado, checa a cada momento se o robô colidiu com algum obstáculo ou se o tempo despendido passou o limite daquele estado, também varia a velocidade angular do robô para a formação da espiral.

- GoBackNode:
  - Enter: Aciona o timer de permanência naquele estado e dá ao robô uma velocidade linear inicial.
  - Execute: Atualiza o tempo naquele estado, checa a cada momento se o tempo despendido passou o limite daquele estado.

- RotateNode:
  - Enter: Aciona o timer de permanência naquele estado, define o ângulo inicial do robô como zero, gera um ângulo aleatório de rotação e dá ao robô uma velocidade angular constante.
  - Execute: Atualiza o tempo naquele estado, define-se a orientação da velocidade angular baseado no ângulo aleatório gerado, se maior ou menor que zero, então atualiza o ângulo do robô e checa este ângulo até ele chegar no objetivo.

Abaixo as figuras 5 e 6 descrevem a execução da Behavior Tree no robô Roomba.

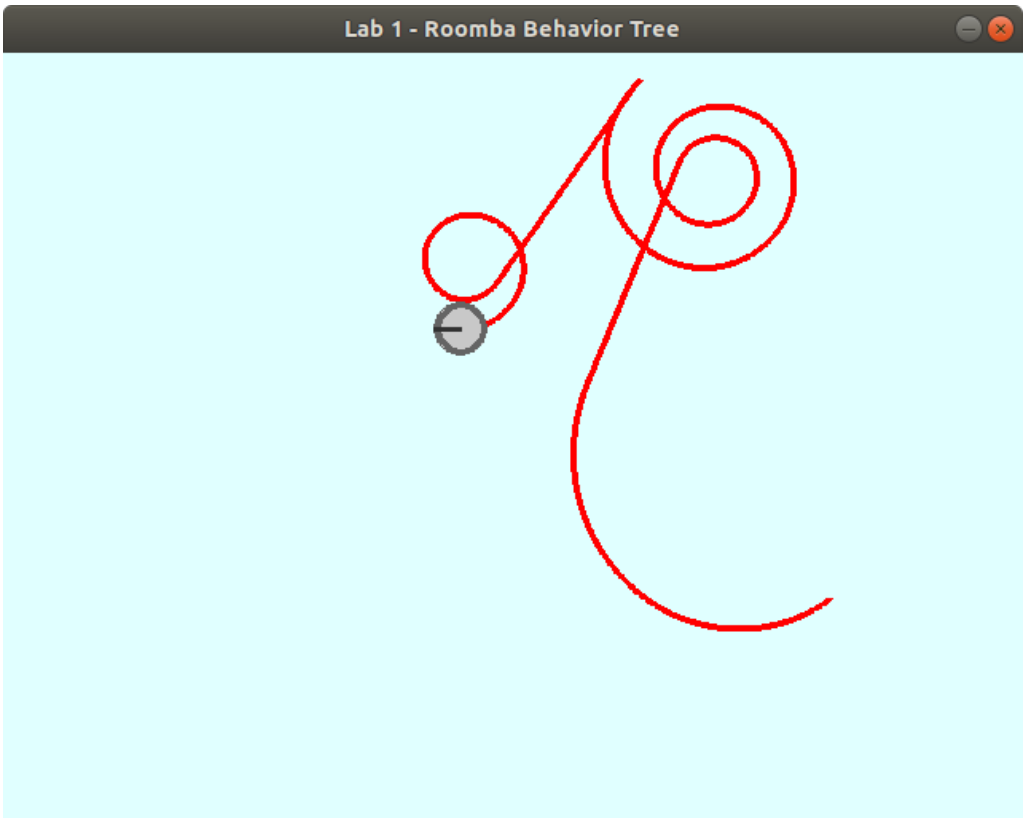

Figura 5: Sequência de movimentos de MoveInSpiral, GoBack, Rotate e MoveForward.

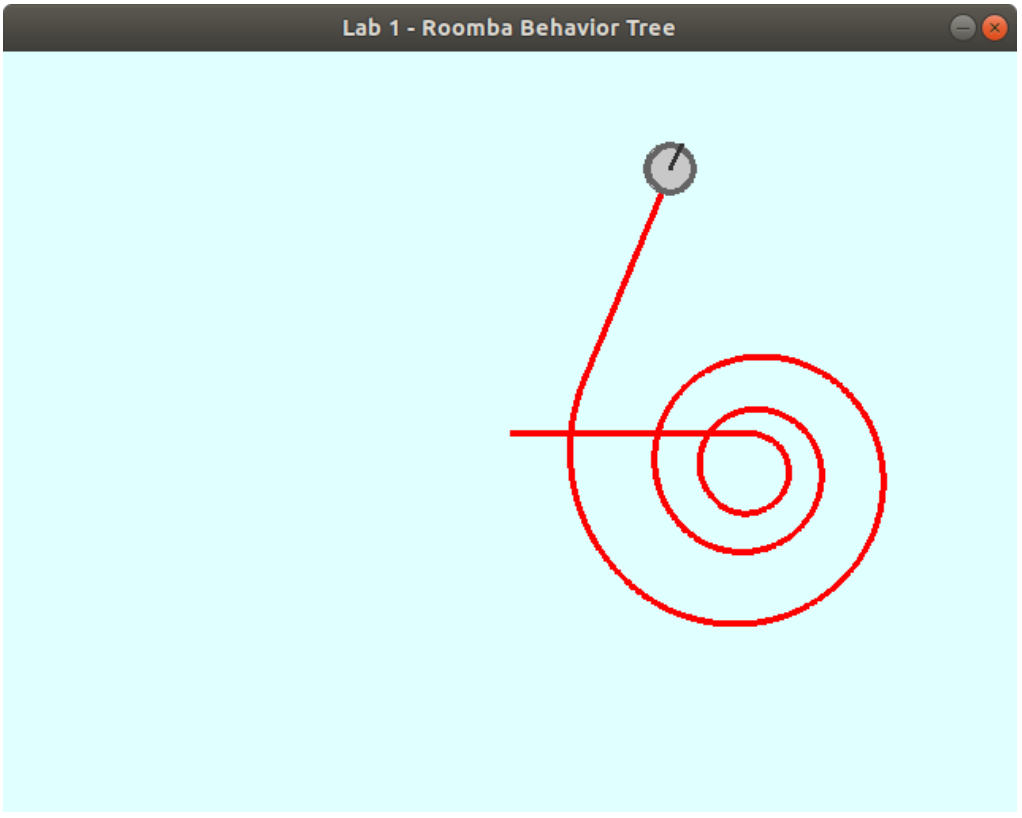

Figura 6: Sequência de movimentos de MoveInSpiral e MoveForward.