Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT213
Aluno: Douglas Bergamim Fernandes

Relatório do Laboratório 1 - Máquina de Estados Finita e *Behavior Tree*

1 Breve Explicação em Alto Nível da Implementação

1.1 Máquina de Estados Finita

A Máquina de Estados Finitos do robô Roomba é composta por quatro estados principais: `MoveForwardState`, `MoveInSpiralState`, `GoBackState` e `RotateState`. Cada estado é representado por uma classe que herda da classe base `State`, que define a estrutura comum, com métodos para verificar transições (`check_transition`) e executar a lógica do estado (`execute`). A FSM é gerenciada pela classe `FiniteStateMachine`, que controla o estado atual e as transições entre eles. Todas as classes de estado são iniciadas com um contador de tempo (`self.time`), que é incrementado a cada frame da simulação com base em `SAMPLE_TIME`. Esse contador é usado para determinar quanto tempo o robô permanece em cada estado antes de transitar para outro.

A mudança de estado é realizada pelo método `change_state` da classe `FiniteStateMachine`, chamado dentro de `check_transition` de cada estado quando as condições para transição são atendidas. O contador de tempo (`self.time`) é reinicializado para `0.0` sempre que o robô entra em um novo estado, garantindo que o tempo seja contado corretamente a partir do início. Essa estrutura modular permite que o robô alterne dinamicamente entre comportamentos, como mover-se para frente, recuar, girar e mover-se em espiral, com base em colisões e tempo gasto em cada estado.

Em resumo, a FSM do Roomba utiliza um contador de tempo baseado em frames para controlar as transições entre estados, garantindo que o robô execute comportamentos específicos por um período determinado ou em resposta a colisões.

1.2 *Behavior Tree*

A Behavior Tree do Roomba começa com um nó raiz do tipo **SelectorNode**, que decide qual subcomportamento deve ser executado. Esse nó raiz contém dois **SequenceNode**: um para a sequência de limpeza (`CleaningSequence`) e outro para a sequência de desvio de obstáculos (`TurningSequence`). A sequência de limpeza inclui nós folha como `MoveForwardNode` (mover para frente) e `MoveInSpiralNode` (mover em espiral), enquanto a sequência de desvio inclui `GoBackNode` (recuar) e `RotateNode` (girar). Os nós folha são responsáveis por ações específicas, enquanto os nós compostos organizam a execução dessas ações de forma sequencial ou seletiva.

Cada ação específica do Roomba é modelada como um nó folha, com atributos e métodos bem definidos. Todos os nós folha inicializam um contador de tempo (`self.time`) para controlar a duração da ação, e alguns, como o `RotateNode`, também inicializam parâmetros específicos, como o ângulo de rotação (`self.angular_rotation`). O método `enter` é chamado quando o nó é iniciado e serve para reinicializar os atributos, garantindo que a ação comece do zero. Já o

método `execute` é chamado a cada frame para executar a lógica da ação, verificando condições (como colisões ou tempo decorrido) e retornando um status (`SUCCESS`, `FAILURE` ou `RUNNING`).

A Behavior Tree oferece, em relação à Máquina de Estados Finitos, uma maior modularidade e flexibilidade. A estrutura hierárquica da Behavior Tree facilita a adição de novos comportamentos sem modificar a estrutura existente, e a organização em nós torna o código mais legível e intuitivo, especialmente para comportamentos complexos. Em comparação, a FSM é mais simples de implementar para comportamentos lineares e sequenciais, mas pode se tornar difícil de manter à medida que o número de estados e transições aumenta.

2 Figuras Comprovando Funcionamento do Código

2.1 Máquina de Estados Finita

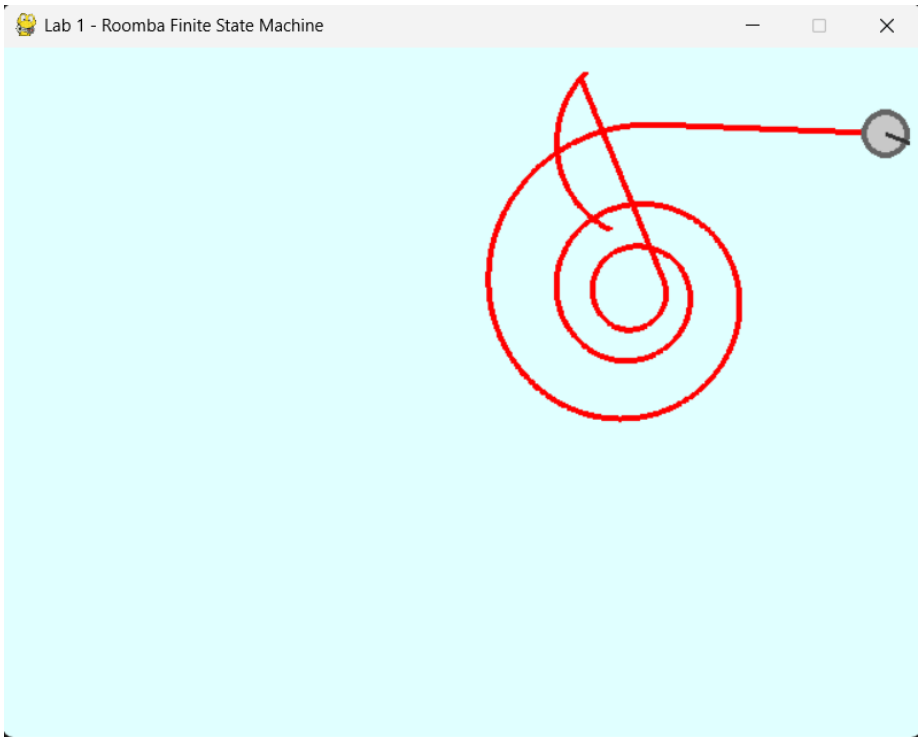

Figura 1: Movimento do Roomba implementado por uma Máquina de Estados Finita.

2.2 *Behavior Tree*

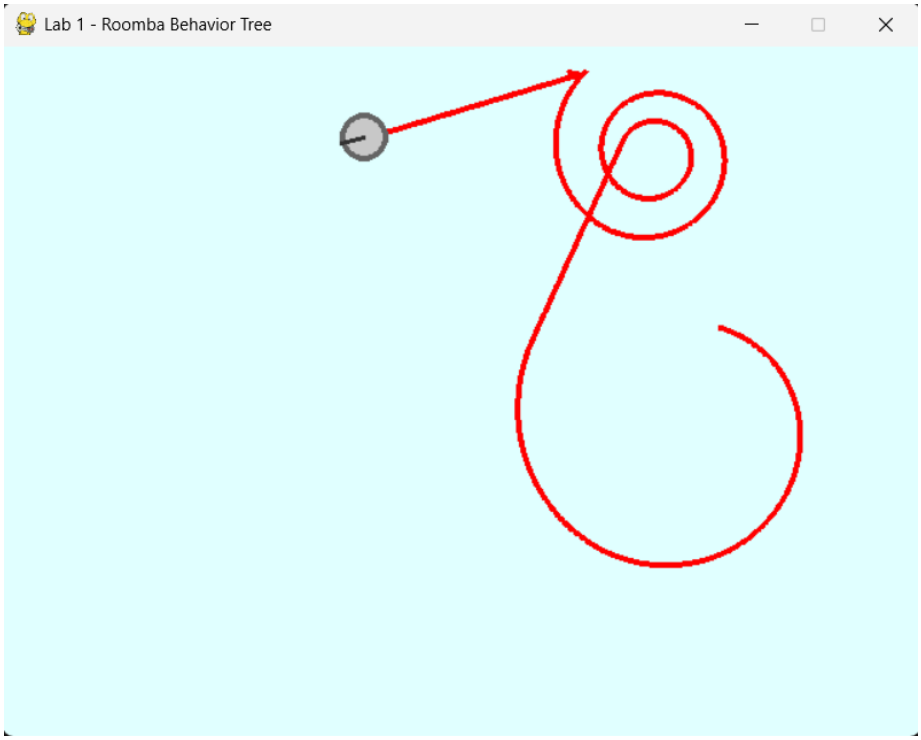

Figura 2: Movimento do Roomba implementado por uma Behavior Tree.