Controle para Rastreio de Trajetória por um Robô com Acionamento Diferencial Utilizando Retroalimentação por Câmera

Igor Mourão Ribeiro
Instituto Tecnológico de Aeronáutica
Rua H8C, 317, CTA
12.228-462 - São José dos Campos/SP
Bolsista PIBIC - CNPq
igormr98mr@gmail.com

Paulo Marcelo Tasinaffo
Instituto Tecnológico de Aeronáutica
Divisão de Ciência da Computação
Praça Marechal Eduardo Gomes, 50
12.229-900 – São José dos Campos / SP
tasinaffo@ita.br

Marcos Ricardo Omena de Albuquerque Máximo
Instituto Tecnológico de Aeronáutica
Divisão de Ciência da Computação
Praça Marechal Eduardo Gomes, 50
12.229-900 – São José dos Campos / SP
maximo.marcos@gmail.com

Resumo: *Devido à alta complexidade existente nos algoritmos de funcionamento de um robô humanoide, é fundamental possuir ferramentas de interface com o usuário para o desenvolvimento do sistema. Estas ferramentas possuem a finalidade de fornecer uma visualização dos algoritmos envolvidos de forma clara para o usuário. Após uma fase de pesquisa, foi decidido utilizar o framework rqt, uma plataforma baseada em Qt que permite o desenvolvimento de interfaces gráficas para ROS (Robot Operational System), assim possibilitando criar um sistema de telemetria para o robô. Com este intuito, foram desenvolvidas três ferramentas iniciais. A primeira, feita antes da pesquisa sobre o framework rqt, foi escrita utilizando apenas a plataforma Qt. Ela é uma interface que permite anotar imagens e gerar tabelas de cores utilizando as funções de treinamento de redes neurais presentes no software MATLAB. A segunda utiliza o framework rqt e é uma ferramenta de teste e calibração do algoritmo de visão computacional do robô humanoide. A última ferramenta é um controle remoto do robô que permite variar a sua velocidade para facilitar a depuração do código.*
Palavras-chave: *robótica, estratégia, tomada de decisão*

no max 4000 caracteres no resumo artigo de no máx 10 páginas tamanho máximo do pdf: 2,5 MB

1. INTRODUÇÃO

A ITAndroids é uma equipe de alunos do ITA, supervisionada por um professor, que participa de diversas competições de robótica nacionais e internacionais. Uma das categorias em que a ITAndroids participa é a do robô humanoide, que consiste em desenvolver um time de robôs capazes de jogar futebol. Esta tarefa envolve uma série de desafios complexos que variam desde a construção do robô até a sua tomada de decisões.

Neste contexto, é fundamental a presença de algoritmos robustos para a realização das diversas ações do robô, tornando essenciais as ferramentas de testes, calibração e depuração. Equipes reconhecidas no cenário internacional, como a B-Human e a Nimbro-OP, possuem várias ferramentas com este intuito, como são apresentadas em Roffer et al. (2013) e em Allgeuer et al. (2013).

Essas ferramentas devem possibilitar o teste dos algoritmos em tempo real, criando um sistema de telemetria com o robô. Além disso, é interessante que elas também possibilitem testes sem a presença do robô, assim facilitando a depuração e calibração do código.

1.1 Behavior Tree

Para se construir uma BT, é preciso primeiro entender o básico de sua estrutura. Uma BT é uma Árvore no sentido dado pela teoria dos grafos, como explicado em West *et al.* (2001). Por definição, cada nó dessa árvore será chamado de behavior.

Os nós de uma Behavior Tree são diferenciados em dois tipos: os nós folhas, que são os nós da árvore que não tem filhos e os nós compostos, que podem ter um ou mais filhos.

Todos os nós retornam, no final de sua execução, um dos três estados a seguir: TRUE, FALSE ou RUNNING, para indicar, respectivamente, que o nó foi terminado com sucesso, que foi terminado sem sucesso ou que o algoritmo deve

executar esse nó por pelo menos mais uma iteração.

1.1.1 Nó Folha

Os nós folha são os nós terminais da Árvore e eles que representam as ações mais baixo níveis e concretas da estratégia. São divididas em dois tipos: nós de ação e nós condicionais.

Nó de Ação São os nós que de fato realizam uma ação, como se movimentar até a bola.

Nó condicional São nós que apenas retornam um estado sem realizar nenhuma ação concreta, como checar se o time está atacando. São usados para controlar quais nós serão executados em conjunto com os nós compostos.

1.1.2 Nó composto

Nós compostos são os nós intermediários da árvore, que são responsáveis por escolher quais os nós que serão executados. São completamente reutilizáveis no sentido que apenas uma implementação de cada tipo de nó composto deve existir e ela poderá ser usada em diversas partes do código. Os principais tipos desses nós que foram usados para a estratégia estão listados a seguir:

Selector Behavior Nesse behavior, é selecionado um de seus filhos para ser acessado a seguir. Ele é usado quando tem-se várias maneiras de se realizar uma mesma ação e deve-se escolher a melhor delas em uma determinada situação. Ele funciona executando os filhos em uma determinada ordem e, assim que um dos filhos retorna TRUE ou RUNNING, esse nó retorna o mesmo valor. Se um filho retornar FALSE, o próximo filho será executado. Caso nenhum filho seja sucedido ou retorne que deve ser executado mais um vez, o nó retorna FALSE.

Sequence Behavior Esse nó executa cada um dos seus filhos em uma sequência bem definida e fixa. Ele é usado para fazer ações sequenciadas para completar um objetivo maior, por exemplo: chutar a bola para o gol oponente requer que o robô aliado esteja atrás da bola, o que significa que chegar atrás da bola e, em seguida, chutar a bola representam ações sequenciadas. Ele funciona de forma que, quando um filho retorna RUNNING ou FALSE, o behavior retorna o mesmo valor e, quando um filho retorna TRUE, ele avança para o próximo filho. Se todos os filhos retornarem TRUE, o behavior é bem sucedido e também retorna TRUE.

Parallel Behavior O objetivo desse nó é executar todos os seus filhos ao mesmo tempo, fato que pode ser obtido ao utilizar-se de processamento paralelo ou não. Na implementação desse trabalho foi usado a versão não paralela do algoritmo. Se um filho retorna FALSE, o behavior retorna FALSE. Se todos os filhos retornarem TRUE, o behavior retorna TRUE. Em todas as outras situações, o behavior retorna RUNNING e continua sua execução por pelo menos outra iteração.

Decorator O nome para esse behavior é inspirado no conceito de Decorator em Software Design Pattern, que pode ser explicado em Hunt (2013). No contexto da Behavior Tree, esse nó tem apenas um filho e ele modifica o comportamento do filho de alguma maneira. Um decorador pode ser diversos tipos, mas os mais usados no projeto foram:

1. AlwaysFail e AlwaysSucceed: que fazem que o fiho sempre retorne FALSE ou sempre retorna TRUE respectivamente.

2. UntilFail e UntilSucceed: fazem que o fiho sempre retorne RUNNING a não ser que ele retorna FALSE ou TRUE, respectivamente.

3. Invert: muda o retorno do filho de FALSE para TRUE e TRUE para FALSE.

1.2 Técnico

Uma Behavior Tree, que embora consegue modelar comportamentos complexos com facilidade, nem sempre é a melhor escolha para determinadas situações. Por isso, a estratégia foi dividida em níveis de abstrações diferentes, em que a um Técnico, representado pela classe Coach, seria responsável por escolher e administrar quais árvore cada um dos três jogadores deveria usar no momento. Com essa implementação, tem-se um agente externo controlando a função dos jogadores por meio da escolha de uma BT, que pode ser uma árvore para um goleiro, para um principal e para um auxiliar. Isso significa que, em um determinado momento, o técnico pode escolher que o time tenha 1 goleiro, 1 principal e 1 auxiliar a até mesmo 3 principais ao mesmo tempo. Os requisitos esperados para cada um desses papéis será abordado a seguir. Além disso, também é função do técnico escolher o papel de cada jogador a cada iteração do código, de modo que as posições dos jogadores não sejam fixas ao decorrer do jogo.

2. RESULTADOS OBTIDOS

Os resultados a seguir foram obtidos estudando o comportamentos de equipes adversárias na competição, além de diversos testes com o time de robôs da ITAndroids contra si mesmo em simulações computacionais em um simulador feito pela própria equipe, como mostrado na Figura 1.

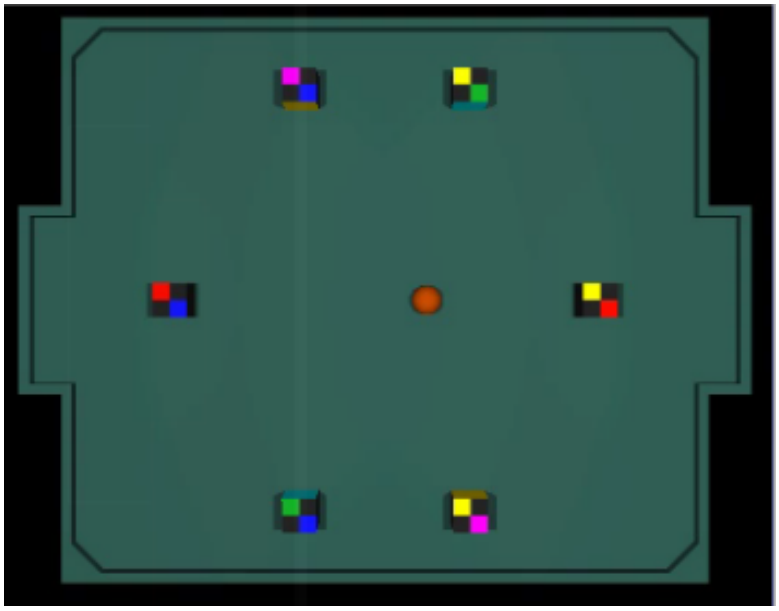

Figura 1: Simulador da ITAndroids.

2.1 Goleiro

O goleiro é o jogador que deve ficar próximo ao gol aliado com o objetivo de proteger o gol de ataques oponentes. A BT criada para o goleiro está representada na Figura 2.

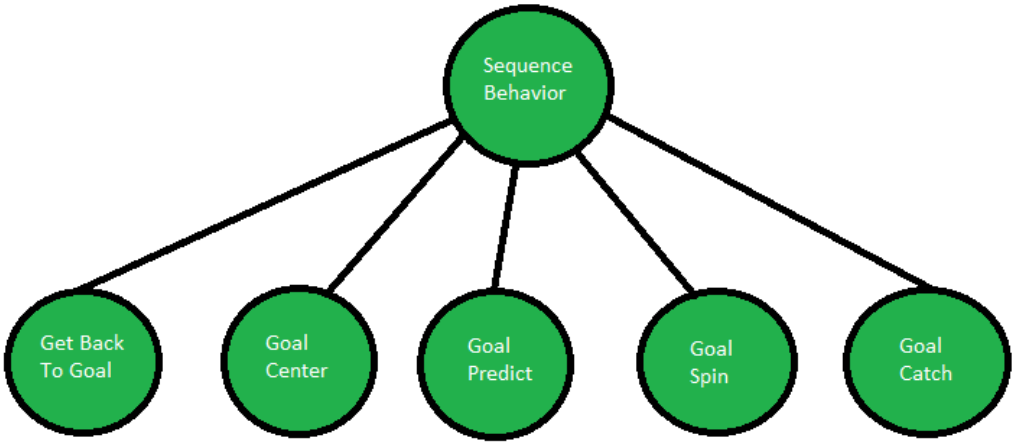

Figura 2: Behavior Tree para o goleiro.

Como visto, ela é composta por um nó do tipo Sequence Behavior, que irá executar os seus filhos em sequência. Esse papel, então, executa as seguintes ações prioritando as primeiras a aparecerem na seguinte lista:

- **Get Back to Goal Behavior** Volta para o gol, caso que, se por algum motivo ele esteja fora do próprio.
- **Goal Center** Fica centralizado no gol quando a bola estiver longe, de forma que o jogador possa rapidamente ir para qualquer um dos lados quando a bola se aproximar.
- **Goal Predict** Prediz para onde a bola irá quando ela estiver rápida e longe do gol.
- **Goal Spin** Gira quando está perto da bola e não tem oponente perto da bola para jogá-la para longe.
- **Goal Catch** Comportamento que o goleiro deverá fazer quando não fizer nenhum outro, por isso sua última posição na sequência. Ele acompanha o movimento da bola com o goleiro sobre a linha do gol.

2.2 Principal

O jogador com o papel de Principal é o jogador mais ativo do time, que deve estar constantemente avançando em direção a bola. Esse papel e o goleiro são as únicas posições que efetivamente deverão tocar na bola. A árvore 3 foi desenvolvida para o principal, sendo a árvore mais complexa dentre os papéis.

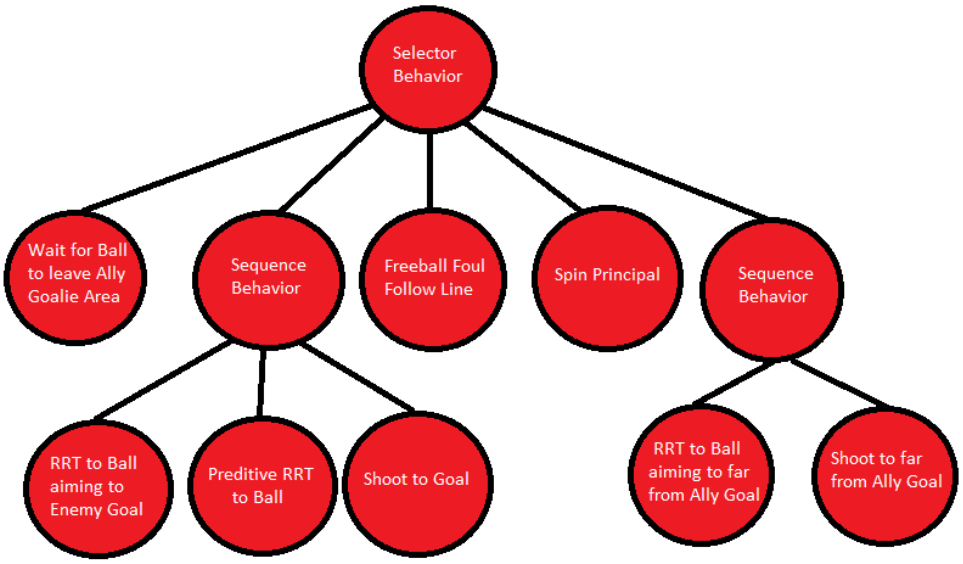

Figura 3: Behavior Tree para o principal.

Conforme a Figura 3, a raiz da árvore do principal é um Selector Behavior que escolhe uma das ações a ser realizada. Essa BT tem mais dois outros nós compostos, que são dois Sequence Behavior usados para o posicionamento atrás da bola, seguido pelo chute. Uma descrição dos nós folha utilizados se encontra abaixo:

- **Wait for Ball to leave Ally Goalie Area** Esse Behavior utiliza a Triangução de Delaunay, com a mesma interface desenvolvida na Figura **??**. Esse comportamento deve evitar que o jogador entre na área do goleiro para não sofrer penâlti, conforme descrito na Figura **??**. O jogador se pocionará de acordo com posições escolhidas pelo usuário, calibradas com o arquivo de configuração do mesmo modelo da Figura **??**.
- **RRT to Ball aiming to Enemy Goal** Irá usar o algoritmo de planejamento de trajetórias RRT para se deslocar atrás da bola com direção apontando para o gol oponente. Usado para se aproximar da bola e, em seguida, trocar para o próximo behavior, o Predictive RRT to Ball.
- **Predictive RRT to Ball** Usa uma predição linear considerando que a bola continuará na mesma velocidade. Usado para se chegar na bola com mais precisão quando próximo a ela. Chega atrás da bola mirando para o gol.
- **Shoot to Goal** Esse behavior só é chamado quando o jogador já está alinhado com a bola e o gol oponente. Esse comportamento acelera rapidamente o robô em linha reta para chegar no gol com uma velocidade alta.
- **Free Ball Foul Follow Line** Esse behavior é um específico para situações de falta do tipo Bola Livre, conforme descrito em **??**. Quando for detectado uma dessas posições, o jogador deve acelerar o mais rápio possível em direção à bola para ter o controle dela antes do oponente.
- **Spin Principal** Esse nó deve ser chamado quando a bola estiver em um dos quatros campos do campo. Caso seja um campo defensivo, o jogador irá girar para jogar a bola para frente; caso seja um na zona de ataque, o jogador girará para por a bola no centro do campo e continuar o ataque.
- **RRT to Ball aiming to far from Ally Goal** Irá usar o algoritmo de planejamento de trajetórias RRT para se deslocar atrás da bola com direção apontando para a zona de ataque (longe do gol aliado). Usado para se aproximar da bola e se posicionar para, em seguida, chamar o Behavior Shoot to Far from Ally Goal.
- **Shoot to Far from Ally Goal** Esse behavior só é chamado quando o jogador já está alinhado com a bola e o para frente (para longe do gol alidado). Esse comportamento acelera rapidamente o robô em linha reta para por a bola na zona de ataque.

2.3 Auxiliar

O auxiliar é um papel que tem apenas uma função: posicionar-se da melhor forma possível para manter um ataque consistente. A ideia é que, caso a situação seja oportuna, o auxiliar se torne o principal e comece a atacar a bola. Para escolher a posição que o auxiliar deverá ficar em uma determinada situação de jogo, foi utilizado a Triangulação de Delaunay (1934) sobre o grafo de Voronoi (1908). Esse algoritmo foi aplicado similarmente a Akiyama and Noda (2007), relativo também à futebol de robôs.

A Behavior Tree do auxiliar, por sua vez, é muito simples, representada por apenas um nó que tem como função se posicionar em lugar ótimo. Sua árvore, é representada simplesmente pela Figura 4.

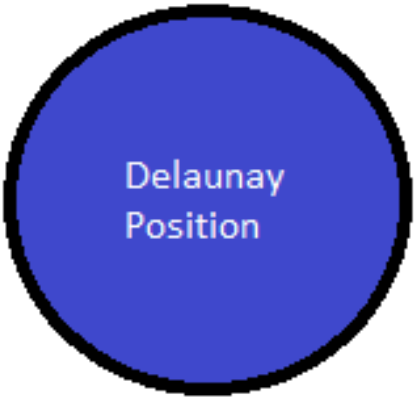

Figura 4: Behavior Tree para o auxiliar.

3. CONCLUSÕES

O projeto de Iniciação Científica apresentou bons resultados, especialmente quanto à vitória da equipe do ITA até às quartas de final da CBR 2017, rendendo o sétimo lugar à equipe dentre mais de 25 equipes participantes. Além do fato de que a ITAndroids conquistou o primeiro lugar na competição nacional de VSSS Copa Turing 2017, que ocorreu no final de setembro de 2017.

Do ponto de vista técnico, o uso do algoritmo da Behavior Tree, já consagrado e usado por times de nível mundial, se mostrou factível e funcional em partidas reais de competições. Contudo, problemas como pouca escalabilidade e pouca reutilizabilidade se mostraram presentes na estratégia desenvolvida. Isso significa que mudanças ainda deverão ser feitas nas BT, principalmente a respeito do uso de mais nós do tipo paralelo para remover esses problemas citados.

4. AGRADECIMENTOS

Agradeço ao CNPq, pelo apoio financeiro e motivacional.

Agradeço à ITAndroids, equipe que representa o ITA na competição da LARC/CBR, pela ideia do projeto, pela disponibilidade do robô real para implementação e oportunidade de aplicação dos métodos estudados.

Agradeço ao professor doutor Paulo Marcelo Tasinaffo, meu orientador, e ao professor doutor Marcos Ricardo de Omena Máximo, co-orientador, ambos da Divisão da Ciência da Computação do ITA, pelo apoio nos estudos e no desenvolvimento do projeto.

5. REFERÊNCIAS

Akiyama, H. and Noda, I., 2007. "Multi-agent positioning mechanism in the dynamic environment". In *Robot Soccer World Cup*. Springer, pp. 377–384.

Delaunay, B., 1934. "Sur la sphére vide". *Bulletin de l'Académie des Sciences de l'URSS*, Vol. 6, pp. 793–800.

Hunt, J., 2013. "Gang of four design patterns". In *Scala Design Patterns*, Springer, pp. 135–136.

Voronoi, G., 1908. "Nouvelles applications des paramètres continus à la théorie des formes quadratiques". *Journal für die Reine und Angewandte Mathematik*, Vol. 133, pp. 97–178.

West, D.B. *et al.*, 2001. *Introduction to graph theory*, Vol. 2. Prentice hall Upper Saddle River.

6. RESPONSABILIDADE AUTORAIS

Os autores são os únicos responsáveis pelo conteúdo deste trabalho.