INSTITUTO TECNOLÓGICO DE AERONÁUTICA
CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO - CNPq

CNPq
CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA - PIBIC

Otimização de parâmetros por meio de Algoritmos Evolutivos em Futebol de Robôs

Igor Mourão Ribeiro

RELATÓRIO PARCIAL DE ATIVIDADES

Orientador: Celso Massaki Hirata

Março / 2019

INSTITUTO TECNOLÓGICO DE AERONÁUTICA
CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO - CNPq

CNPq
CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA - PIBIC

Relatório Parcial de Atividades

Otimização de parâmetros por meio de Algoritmos Evolutivos em Futebol de Robôs

São José dos Campos, __ / __ / 2019

Nome do aluno	Igor Mourão Ribeiro
Assinatura do aluno	

Nome do orientador	Celso Massaki Hirata
Assinatura do orientador	

INSTITUTO TECNOLÓGICO DE AERONÁUTICA

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA - PIBIC

Formulário de Aprovação de Relatório pelo Orientador

Relatório: [X] Rel. Parcial [ ] Rel. Final

1- CONSIDERO O RELATÓRIO APROVADO COM BASE NOS SEGUINTES ASPECTOS

2- APRECIAÇÕES DO ORIENTADOR SOBRE O DESEMPENHO DO BOLSISTA NA EXECUÇÃO DO TRABALHO DE INICIAÇÃO CIENTÍFICA

Local e data: São José dos Campos, __/__/2019

Assinatura do Orientador:

Sumário

1 Resumo do Plano Inicial **5**

2 Resumo das Atividades Realizadas **5**

3 Descrição do Problema **6**

3.1 Covariance Matrix Adaptation Evolution Strategy 9

3.1.1 O algoritmo 10

3.2 O simulador 11

3.3 Univector Field 12

3.4 Papéis 13

3.4.1 Goleiro 13

3.4.2 Principal 14

3.4.3 Auxiliar 16

3.5 Táticas de Jogo 18

4 Resultados Obtidos **18**

4.1 Papéis 19

4.1.1 Goleiro 19

4.1.2 Principal 20

4.1.3 Auxiliar 21

4.2 Técnicos 22

4.2.1 Dois principais 22

4.2.2 Um principal e um auxiliar 22

4.2.3 Troca de Goleiro 23

5 Conclusões **24**

6 Agradecimentos **24**

7 Bibliografia **24**

1 Resumo do Plano Inicial

O objetivo deste trabalho é desenvolver e implementar um algoritmo para a otimização de parâmetros de um time composto por três robôs diferenciais. O contexto é uma partida de futebol segundo a Competição Brasileira de Robótica (CBR), categoria IEEE Very Small Size Soccer (VSSS). Por meio da criação de métricas e testes automatizados para medir o desempenho do time de robôs em diversas situações de jogo, pode-se utilizar algoritmos evolutivos para fazer a otimização de parâmetros relevantes, como os relativos ao planejamento de trajetória. Visa-se utilizar os algoritmos aqui implementados pela equipe da ITAndroids, que representa o ITA em competições nacionais e internacionais, como na competição Latin America Robotics Competition (LARC)/CBR.

Planejamento:

- 1o Bimestre (ago / set): Estudo da literatura e revisão bibliográfica sobre o tema de otimização de parâmetros para robôs jogadores de futebol.
- 2o Bimestre (out / nov): Seleção de métricas a serem utilizadas e implementação inicial do código.
- 3o Bimestre (dez / jan): Implementação dos testes e simulações para as métricas escolhidas. Redação do relatório parcial.
- 4o Bimestre (fev / mar): Análise e implementação do algoritmo escolhido.
- 5o Bimestre (abr / mai): Análise do desempenho da otimização para o planejamento de trajetórias. Ajuste do algoritmo para melhores resultados.
- 6o Bimestre (jun / jul): Redação do relatório final. Redação do artigo para o ENCITA. Teste final para o software completo no robô real, com todas as áreas do projeto operantes.

2 Resumo das Atividades Realizadas

Ao longo dos dois primeiros meses, foram realizados estudos e foi decidido que será usado o algoritmo Covariance Matrix Adaptation Evolution Strategy (CMA-ES) para a otimização de parâmetros. Esse algoritmo foi escolhido dentre de opções como NEWUOA

e o algoritmo de Broyden–Fletcher–Goldfarb–Shanno (BFGS). Mais detalhes sobre esses dois últimos métodos pode ser encontrada, respectivamente, em [1] e em [2]. Para uma descrição sobre a eficácia do algoritmo CMA-ES no contexto de futebol de robôs, pode-se encontrar na referência [3].

Ao longo do segundo e terceiro bimestres, a base do algoritmo foi implementada utilizando Matlab [4], que é uma linguagem especializada em algébra linear utilizada pela equipe. Foi, em seguida, realizada a escolha e codificação das métricas a serem utilizadas para medir o desempenho de uma determinada situação de jogo.

O algoritmo foi, em seguida, implementado e integrado com o restante do código da equipe. Após algumas iterações, os parâmetros relativos ao planejamento de trajetória do robô foram otimizados e testados em simulações. Os resultados se mostraram satisfatórios e condizentes com o previsto.

Os resultados obtidos foram efetivos como mostrado na LARC, ocorrida entre os dias 6 a 10 de novembro de 2018. O algoritmo mostrou que o planejamento de trajetórias da equipe, em conjunto com o controle do robô, estavam muito precisos, fato que concedeu à equipe uma vantagem decisiva na competição. A equipe da ITAndroids conquistou o primeiro lugar da competição, sendo a sua classificação a mais alta na categoria Very Small Size Soccer na história.

3 Descrição do Problema

No contexto do futebol de robôs, a otimização de parâmetros traz muitos desafios. Tal problema consiste no fato de se implementar um algoritmo que consiga encontrar as melhores constantes a serem usadas no código para melhorar o desempenho do robô em uma partida real. Entretando, para encontrar esses parâmetros de forma automatizada, é necessário, além do algoritmo evolutivo em si, a existência de simulações fiéis à realidade. A figura 1 mostra a placa eletrônica do robô utilizado.

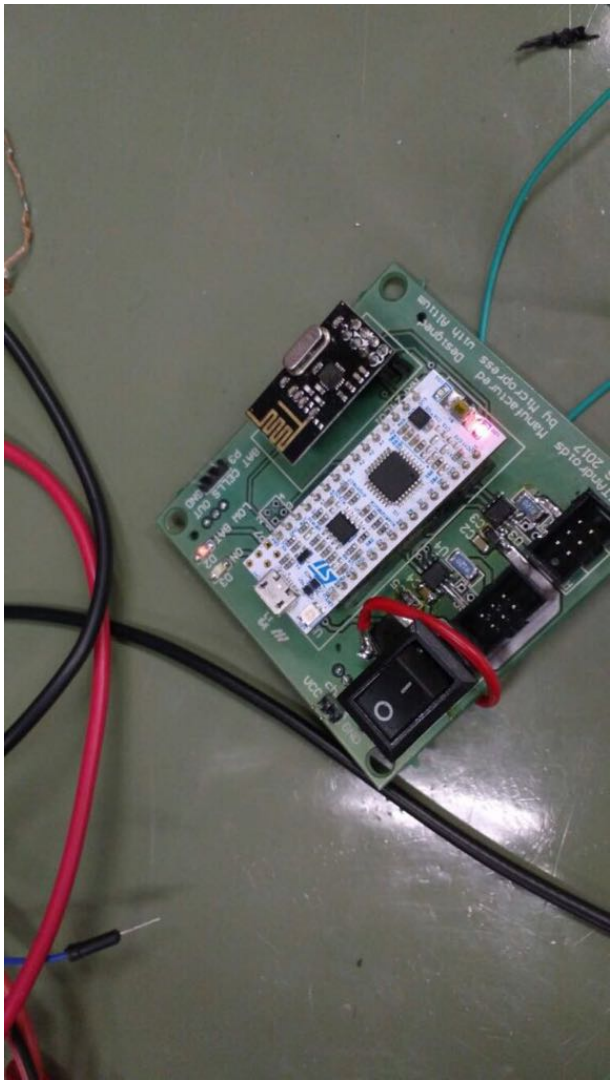

Figura 1: Placa eletrônica de um robô Very Small Size.

Nesse projeto, trabalhou-se com os robôs da competição IEEE Very Small Size (VSS): uma competição de futebol de robôs completamente automatizada em que cada robô tem como tamanho máximo um cubo de 7,5 cm de aresta. O campo de futebol possuil 1,5 m de comprimento e 1,3 m de largura. Cada time tem 3 jogadores: que podem assumir posições dinâmicas, como goleiro e atacante, ao decorrer de uma partida. Uma câmera proporciona a visão aérea com as posições de todos os elementos da partida. As regras completas podem ser lidas em [5].

Figura 2: Robôs da ITAndroids da categoria IEEE VSSS.

A figura 2 mostra os robôs usados nas competições. A visão computacional utilizada pela equipe ITAndroids pode ser encontrada com mais detalhes em [6] e utiliza uma câmera no topo do campo que repassa informaçãoes para um computador processar, como visto na figura 3.

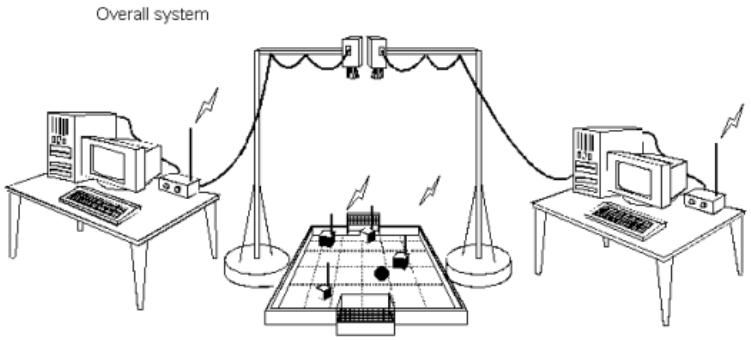

Figura 3: Representação do funcionamento técnico de um jogo de VSS.

Cada robô foi projetado com duas rodas laterais e duas esferas livres à frente e atrás para manter o equilíbrio. Essa característica o torna um robô diferencial. Ou seja, temos o controle sobre as velocidades linear e angular, mas não é possível mover o robô para os lados. A linguagem utilizada para o projeto foi C++; diferente da utilizada nesse trabalho, que é Matlab; pois é uma linguagem que tem uma velocidade de execução alta e tem escalabilidade descente para um projeto grande.

Nesse contexto, surgem várias questões a serem resolvidas para se ter um time vencedor. Nesse projeto, será abordado o seguinte problema de otimização: dado um simulador feito pela equipe que simula a física e a lógica de uma partida oficial, deve-se encontrar

os melhores valores possíveis para as constantes do planejamento de trajetórias, de modo a maximizar o desempenho do time em uma partida com os robôs reais.

Sobre o assunto de planejamento de trajetórias, o utilizado pela equipe é o Univector Field [7], que é muito eficiente e já demonstrou bons resultados em competições oficiais com algumas das melhores equipes utilizando-o. Uma das suas desvantagens, é a existência de até dez parâmetros diferentes para serem otimizados e de que essas constantes não são intuitivas para uma pessoa tuná-las sem ajuda de um computador. Isso acontece devido ao comportamento do algoritmo ser relativamente complexo quando comparado com outras alternativas como o algoritmo Campos Potenciais [8].

Além disso, é importante ressaltar que o tempo de execução do algoritmo deve ser o mínimo possível. Nota-se que não é possível simular uma partida completa de futebol já que isso demoraria bastante e o algoritmo precisa de em torno de 100 iterações para convergir. Para isso, técnicas para executar simulações aceleradas e executar situações específicas de jogo foram desenvolvidas, com o objetivo de reduzir o tempo total da otimização.

Ademais, outro problema que será enfrentado no desenvolvimento desse projeto é a criação de interfaces de comunicação entre o algoritmo de otimização, a ser codificado em Matlab, com o código do projeto e das simulações, ambos codificados na linguagem de programação C++.

Para resolver esse problema, foi escolhido como algoritmo para a otimização de parâmetros o Covariance Matrix Adaptation Evolution Strategy, já que ele consegue otimizar problemas não lineares e não convexos, que, devido à elevada complexidade, é o caso do futebol de robôs.

3.1 Covariance Matrix Adaptation Evolution Strategy

O algoritmo CMA-ES é uma estratégia evolutiva. Um algoritmo evolutivo é baseado no princípio da evolução biológica, em que conceitos de recombinação e mutação de genes são aplicados para selecionar melhores indivíduos. Mais formalmente: em cada geração novos indivíduos são criados a partir de seus pais, com o objetivo de gerar melhores indivíduos segundo algum critério.

No contexto da computação, cada geração será considerada uma iteração. Em cada iteração existirá uma população composta de indivíduos (ou candidatos) que deverá ser

alterada segundo um algoritmo para diminuir uma determinada função de custo.

Para o caso do CMA-ES, os novos indivíduos são escolhidos com base em uma distribuição gaussiana de múltiplas dimensões. A nova população será gerada a partir da média dos melhores indíviduos da população anterior, em combinação com uma pertubação gaussiana estocástica. Dependência entre diferentes parâmetros seram representadas por uma matriz de covariâncias.

Nota-se que o algoritmo, brevemente descrito, não usa derivadas nem valores das funções de custo, apenas importa a comparação entre os desempenhos dos indíviduos de uma população.

3.1.1 O algoritmo

O algoritmo do CMA-ES pode ser separado em alguma etapas:

1. Escolha dos parâmetros iniciais para a otimização: o tamanho do passo, o ponto inicial, o tamanho da população e as condições de parada do algoritmo (que podem ser número de iterações ou ter um custo menor que um certo valor).

2. Enquanto a condição de parada não for atingida:

   (a) Gera novas soluções a partir da média dos melhores indivíduos da iteração anterior. Para isso usa-se uma distribuição gaussiana normal centrada em tal média e variância inicial igual ao tamanho do passo.

   (b) Cálculo dos custos de cada novo indivíduo por meio de uma função chamada de fitness.

   (c) Ordenação dos indíviduos, com base nos seus custos.

   (d) Atualização dos valores de variáveis internas.

Não será abordado os detalhes técnicos do item 2d, pois a implementação desse algoritmo não é o foco desse trabalho. Essa decisão ocorreu porque, além de tal passo ser bastante complexo estatisticamente, esse item apenas otimiza a velocidade de convergência do algoritmo e não é necessário para o entendimento inicial da estratégia evolutiva. Uma descrição mais profunda e detalhada de como otimizar o CMA-ES se encontra em [9].

3.2 O simulador

O simulador é uma aplicação separada do código principal do time, desenvolvida e idealizada pela própria equipe, que cria um ambiente com física que simula um jogo. Tal aplicação se comunica com o código principal por meio de uma biblioteca de comunicação [10], em que se transfere informações sobre as posições do robôs para o código principal, enquanto que esse último transmite informações de velocidade das rodas de cada robô para o simulador.

A simulação foi desenvolvida em C++ com o auxílio da biblioteca Odeint [11], para fazer as computações das equações diferenciais da física e das colisões dos movimentos, e da biblioteca Ogre3D [12], responsável por renderizar os objetos na tela em três dimensões. A figura 4 mostra a interface visual do simulador logo antes de se começar uma partida.

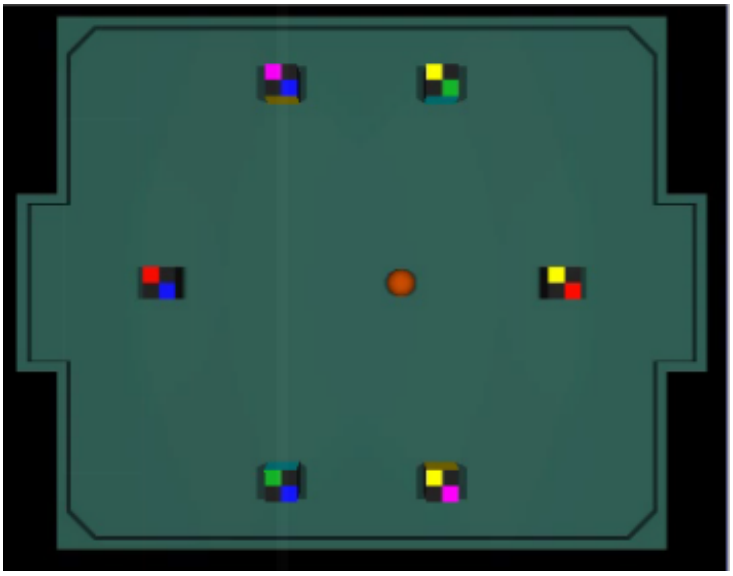

Figura 4: Simulador da ITAndroids.

Com isso, é possivel ter uma aproximação do comportamento do time em um jogo real. Dado que o simulador é desenvolvido pela própria equipe, é possível adicionar situações especiais de jogo, além de se adicionar condições de paradas específicas para o que se deseja testar. Essas funcionalidades são muito importantes para a criação de situações específicas de jogo e, consequentemente, de funções de custo mais rápidas e otimizadas, o que garante uma convergência mais rápida de determinados parâmetros.

Além disso, o simulador também tem um modo de funcionar que é o chamado: modo acelerado. Normalmente, o código tem algumas funções para esperar um determinado

tempo, com o objetivo de fazer que ele se pareça mais com a realidade (se ele não tivesse, a física seria muito rápida). Porém, no modo acelerado, esses tempos ociosos são removidos e a comunicação com o código principal funciona de forma assíncrona, o que garante uma velocidade de executação e simulação muito mais rápida, chegando a ser até dez vezes mais rápida que o modo de execução normal. Logo, o modo acelerado é fundamental para garantir que a função de custo a ser desenvolvida seja mais eficiente e veloz.

3.3 Univector Field

O Univector Field é o planejamento de trajetórias usado pela equipe. O problema a ser resolvido é: como chegar de um ponto à ponto dentro do campo, desviando de obstáculos e considerando as limitações de controle e físicas do robô. Não será detalhado o funcionamento completo do algoritmo do Univector Field, já que isso pode ser visto em [7].

Entretando, será descrito o que cada constante que o algoritmo espera receber faz, de modo a ajudar a visualizar os resultados que as otimizações trarão. Tem-se as seguinte constantes a serem otimizadas:

1. de:
2. kr:
3. sigma: Parâmetro que define
4. dmin:
5. K0:
6. sigmaLine:
7. dminLine:

O Univector Field é integrado com a interface de depuração da equipe, o que ajuda, qualitativamente, a verificar o quão otimizado é uma determinada trajetória gerada pelo algoritmo. Isso faz com que seja simples e rápido fazer uma checagem manual para saber se as constantes que o algoritmo convergiu não são absurdas. Uma visualização da interface pode ser vista em 5.

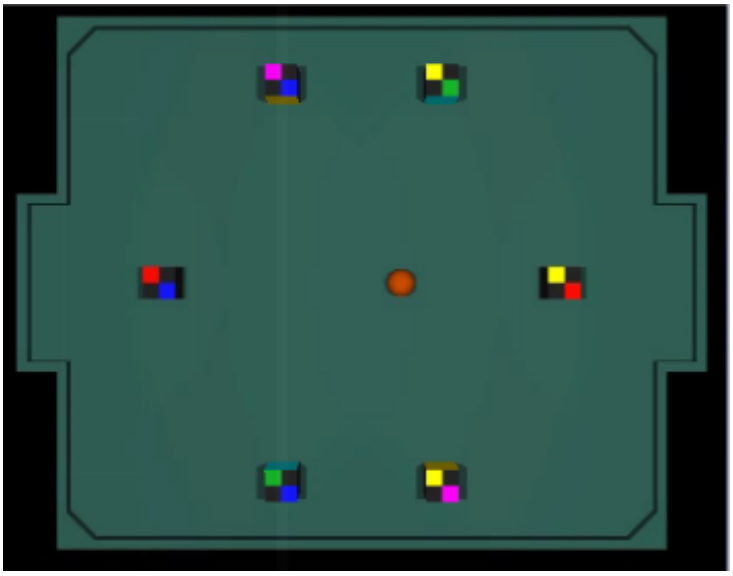

Figura 5: Interface gráfica do Univector Field

3.4 Papéis

Dentre os papéis que o técnico pode escolher, os escolhidos foram Goleiro, Principal e Auxiliar, conforme proposto por [13] como papéis de alta performance.

3.4.1 Goleiro

O goleiro é o jogador que deve ficar próximo ao gol aliado com o objetivo de proteger o gol de ataques oponentes.

1. Ele deve ser capaz de defender bolas com velocidade lenta em direção ao gol, ao acompanham o movimento da bola seguindo a linha do gol.

2. Essa posição deve conseguir proteger o gol de bolas rápidas, usando para isso um modelo de predição que supõe que a bola continuará em linha reta. Essa predição é necessária, pois, a altas velocidades, a taxa de amostragem da câmera não é alta o suficiente para conseguir comandar o robô apenas para seguir o movimento da bola, sendo necessário antecipar onde ela irá. A figura 6 mostra como essa predição linear funciona, indicando a direção da bola e o movimento que o goleiro deverá fazer para proteger o gol.

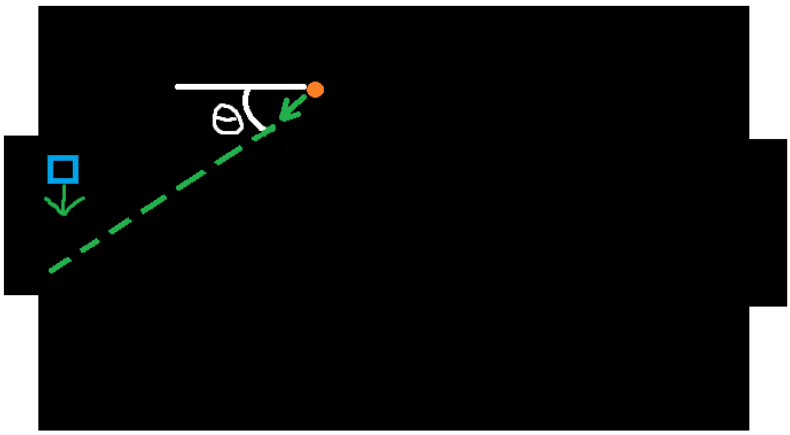

Figura 6: Predição usada pelo goleiro para bolas rápidas.

3. Deve jogar a bola para longe do gol, por meio de rotações em torno do próprio eixo quando chegar próximo da bola para diminuir o risco de gol.

4. Deve sair do gol quando a situação não é de risco e ele chegará na bola antes de qualquer oponente. Nesse caso, outro jogador deverá se tornar um goleiro.

3.4.2 Principal

O jogador com o papel de Principal é o jogador mais ativo do time, que deve star constatemente avançando em direção a bola. Essa e o goleiro são as únicas posições que efetivamente deverao tocar na bola. Esse papel deve atender os seguintes requisitos:

1. deve atacar em direção ao gol oponente sempre que a bola não esteja muito próxima da parede. Isso será feito usando o planejamento de trajetórias RRT para se posicionar atrás da bola, seguido de uma rápida aceleração com a bola em linha reta em direção ao gol, de forma que o jogador leve a bola com a maior velocidade possível ao gol.

2. deve atacar pelas laterais do campo quando a bola estiver nos lados do campo de forma que ele jogue a bola para frente, sem necessariamente ter como objetivo fazer o gol, mas garantindo que o time agora esteja atacando. Para isso acontecer, será usado o RRT para se posicionar atrás da bola, seguida de uma rápida aceleração para frente.

3. deve girar em torno do próprio eixo quando estiver em um dos quatro cantos do campo. Caso esteja no ataque, esse giro fará que a bola se desloque para o meio do campo, o que pode significar que outro jogador possa pegar essa bola e levar para o gol, qualificando essa jogada com um passe entre jogadores. Caso a bola esteja em um dos cantos da defesa, o giro, além de tirar a bola da zona defensiva e a por numa posição de ataque, impedirá que a bola fique presa, fato que caracteriza em falta tiro livre segundo [5]. A descrição da penalidade bola livre pode ser vista na imagem 7.

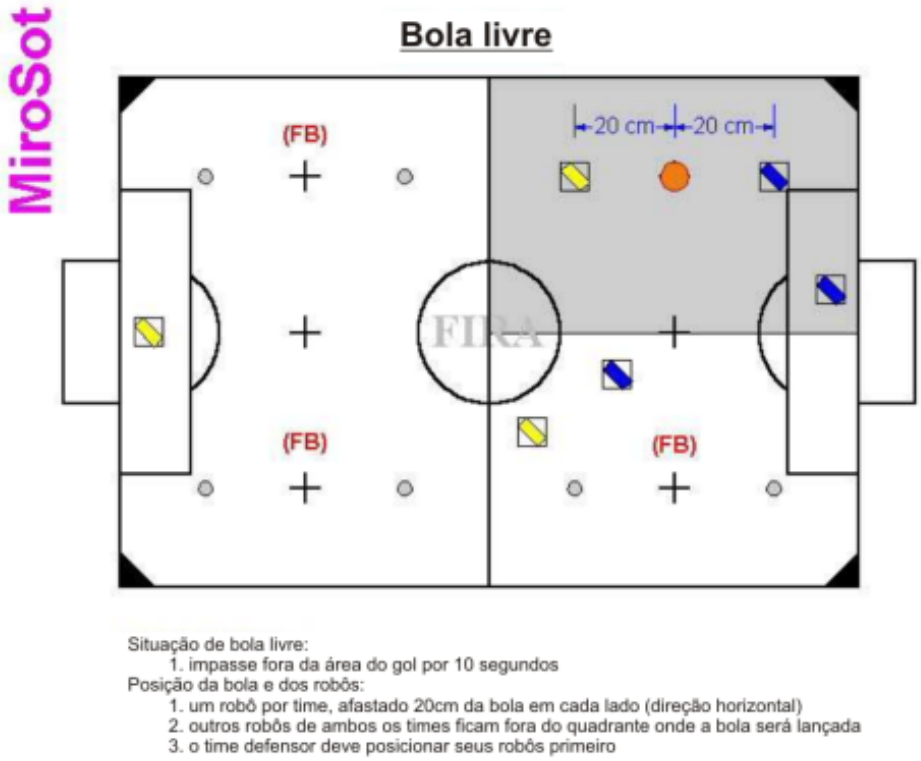

Figura 7: Descrição da penalidade bola livre.

4. deve ter comportamentos fixos e especiais para cada situações de falta, como por exemplo em um penalty, que ele deve levar a bola o mais rápido possível para um dos lados do gol para o goleiro não ter tempo de reagir e simplesmente atingir ou em um tiro livre, em que ele deve acelerar o máximo possível para chegar na bola antes do oponente. A descrição da falta do tipo penalty se encontra na figura 8.

Figura 8: Descrição da penalidade penalty.

3.4.3 Auxiliar

O auxiliar é um papél 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 determiada situação de jogo, foi utilizado a triangulação de Delaunay [14] sobre o grafo de Voronoi [15]. Esse algoritmo foi aplicado similarmente a [16], relativo também à futebol de robôs.

Em seguida, uma interface foi desenvolvida para que fosse possível escolher as melhores posições dos jogadores aliados para cada posição da bola. A implementação para escolher a posição do auxiliar, no contexto da ITAndroids, funcionará seguindo os passos a seguir:

1. Escolhe-se uma posição estratégica para a bola no campo.

2. É escohido a melhor posição para um ou mais auxiliares quando a bola estiver nessa posição. A interface desenvolvida em C++, com a biblioteca Qt[17], foi usada para esses dois passos. A visualização pode ser vista na figura 9.

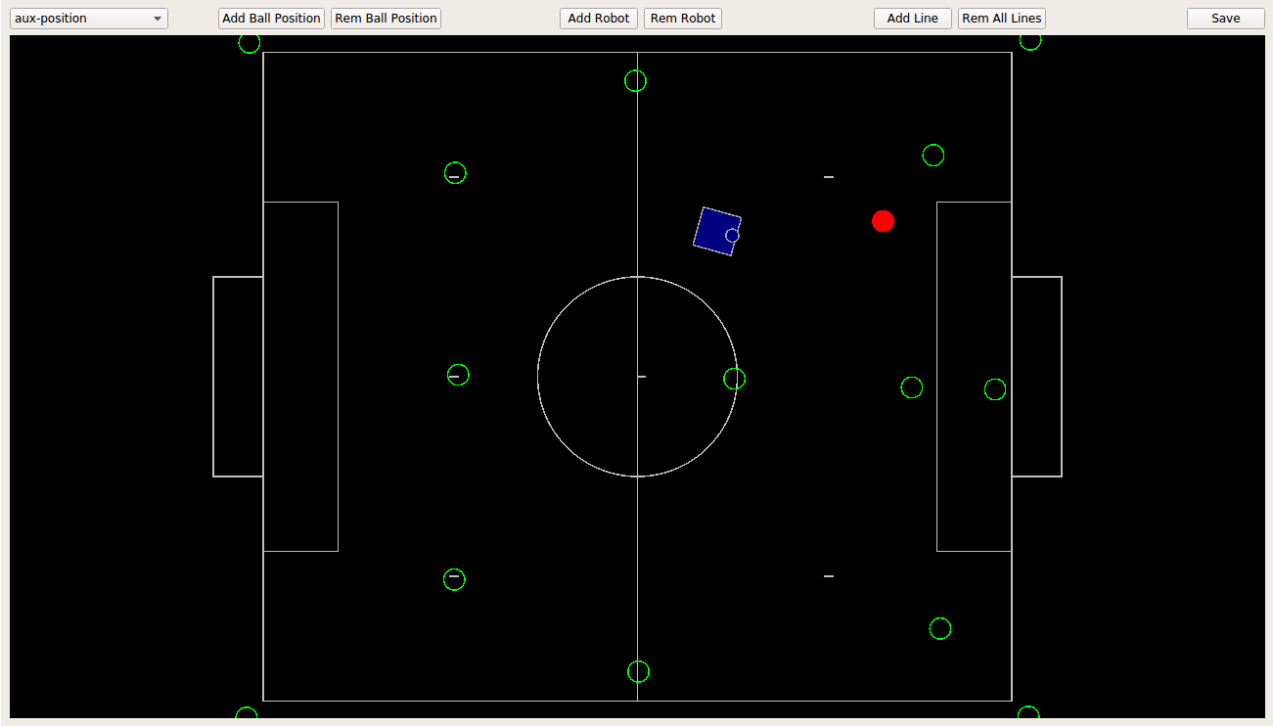

Figura 9: Interface para posicionamento com triangulação de Delaunay.

3. Os valores dessas posições são guardadas em um arquivos de texto para posterior uso no código da estratégia. Um exemplo desse arquivo de configuração se encontrar na figura 10, em que temos 2 jogadores em três posições, como escrito na linha 1 do arquivo; seguido pela localização da bola, com apenas as coordenadas x e y, e as posições dos jogadores, com as coordenadas x,y e ângulo. Essas localizações são repetidas três vezes conforme escolhido (linha 1).

```
1  2 3
2  -0.777467 0.669486
3  -0.378918 -0.0117796 0.0128183
4  -0.41622 0.555614 -0.234253
5  0.787284 0.675376
6  0.367137 0.587027 -0.281477
7  0 0 0
8  0.783357 -0.681266
9  0.270935 -0.514385 0.154319
10 0.00588987 2.60591e-07 0
```

Figura 10: Arquivo de configuração para posicionamento dcom triangulação de Delaunay.

4. A estratégia do auxiliar utiliza a triangulação de Delaunay para a posição atual da bola, em conjunto com o arquivo de configuração feito, para interpolar as posições discretas escolhidas e fazer que o algoritmo retorna uma posição ótima independente da localização da bola.

5. Com a posição obtida pela triangulação, é usado o algoritmo de planejamento de trajetórias RRT, em conjunto com o seu controlador, para chegar nessa posição interpolada.

3.5 Táticas de Jogo

Táticas de Jogo, representado pela classe 'Play', é a parte mais alto nível da estratégia. Essa classe é responsável por escolher qual técnico usar em um determinado momento da partida. O Play escolhe qual o melhor modo de se jogar em uma situação, como, por exemplo, ter uma postura mais agressiva ou mais defensiva envolvendo os três jogadores. Só pode existir um Play no jogo, que pode ser tratado como a raiz de uma Behavior Tree que tem os técnicos como filhos, tendo, cada um, três árvores relativas a cada jogador.

O seu funcionamento é similar ao Selector Behavior, no sentido em que é preciso escolher apenas um técnico a cada momento. Para avaliar quando é necessário fazer uma troca, usa-se o mesmo modelo de retorno: TRUE, FALSE e RUNNING para os técnicos.

Dado essa troca, pode ser criados técnicos temporários para jogadas específicas de jogo, como o goleiro sair do gol e voltar outro jogador, que é uma ação de transição. Isso pode significar que um técnico principal pode chamar esse outro, voltando depois para o mesmo técnico novamente. Então, podemos ter um técnico principal e vários temporários para jogadas ensaidas e específicas que envolvam mais de um jogador.

4 Resultados Obtidos

Nessa seção, são apresentados os resultados obtidos na resolução do problema da tomada de decisão para robôs jogadores de futebol IEEE Very Small Size Soccer. 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 4.

4.1 Papéis

Aqui, será apresentado os papéis criados para a competição e as árvores de comportamento criadas para cada um deles.

4.1.1 Goleiro

A BT criada para o goleiro está representada na figura 11.

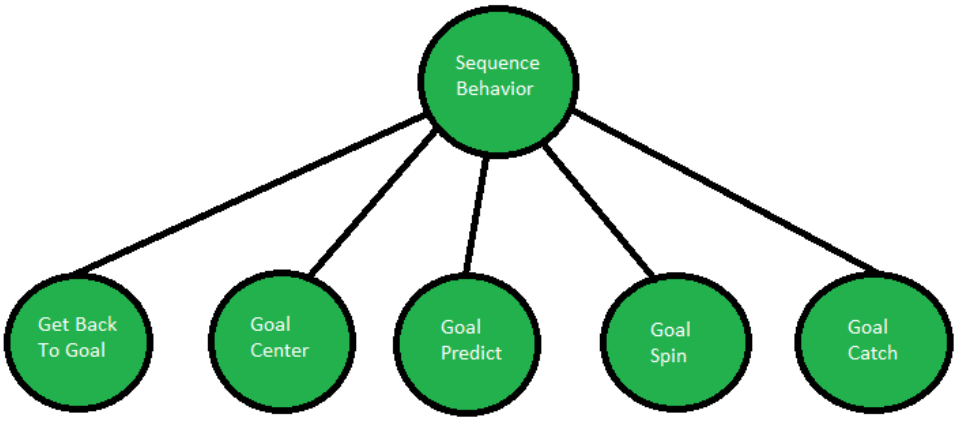

Figura 11: Behavior Tree para o goleiro.

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.

4.1.2 Principal

Já para o principal, a árvore 12 foi desenvolvida, sendo a árvore mais complexa dentre os papéis.

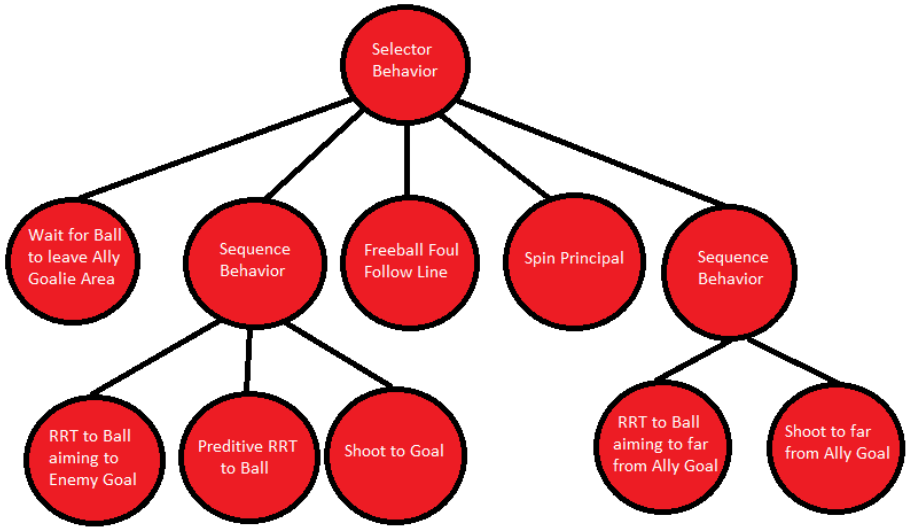

Figura 12: Behavior Tree para o principal.

Conforma a figura 12, 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 9. Esse comportamento deve evitar que o jogador entre na área do goleiro para não sofre penalty, conforme descrito na figura 8. O jogador se pocionará de acordo com posições escolhidas pelo usuário, calibradas com o arquivo de configuração do mesmo modelo da figura 10.

- **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 7. 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.

4.1.3 Auxiliar

O auxiliar, por sua vez, tem uma Behavior Tree muito simples, representada por apenas um nó que tem como função se posicionar em lugar ótimo com Triangulação de Delaunay[14]. Sua árvore, é representada simplesmente pela figura 13.

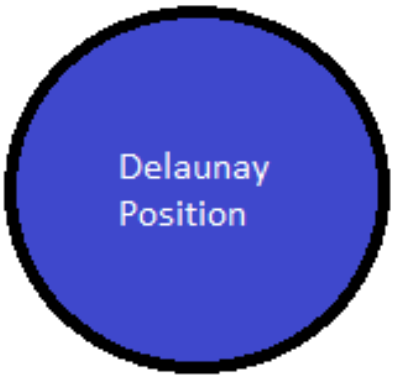

Figura 13: Behavior Tree para o auxiliar.

4.2 Técnicos

Finalmente, será mostrado quais técnicos foram usados na estratégia. A princípio, três técnicos foram desenvolvidos. Primeiramente, será descrito o comportamento de cada um deles e quando é esperado que cada um seja usado, seguido das condições de troca entre eles.

4.2.1 Dois principais

Esse técnicos usa uma formação de jogadores com 1 goleiro e dois jogadores principais. Isso significa que os dois principais irão para à bola ao mesmo tempo, o que pode ser caótico para uma situação de ataque, em que precisão para acertar a bola é necessária. Porém, para situaçõe defensivas, esse técnico demonstra muita eficácia, porque os dois jogadores se tornam efetivamente zagueiros, indo para a bola e dificultando o avanço do atacante adversário.

4.2.2 Um principal e um auxiliar

Esse técnico, diferente do anterior, é usado para situações de ataque. Ao utilizar apenas um principal, essa formação é capaz de fornecer ataques muito mais precisos, em que os jogadores aliados não se colidem durante a movimentação. O auxiliar, nessa formação, não fica muito longe da bola, ficando esperando a bola escapar do principal, por uma defesa do oponente por exemplo. No momento que o auxiliar estiver em uma posição mais próxima da bola que o principal, ele se torna o principal, enquanto que o principal

se torna o auxiliar, de modo que o ataque seja contínuo. Essa formação se mostrou muito eficaz na CBR, fazendo que a ITAndroids tivesse um dos ataques, em termos de estratégia, mais consistentes da competição.

4.2.3 Troca de Goleiro

Esse é um tipo de técnico temporário para realizar uma jogada específica de jogo, sendo feito para uma ação de transição. Esse técnico será chamado em situações em que o goleiro deverá sair do gol e se tornar um principal, enquanto que o outro jogador mais próximos do gol se torna o novo goleiro. A situação em que essa troca deve ocorrer é quando a bola está perto do goleiro e todos os jogadores inimigos estão bem longe, condição necessária por se tratar de uma jogada arriscada.

Tecnicamente, esse técnico é usado em situações em que temos 2 principais e um goleiro, já que ele só ocorrer em situações defensivas. Ele funciona fazendo que o goleiro se torne o principal sem o Behavior de esperar a bola sair do gol, como mostrado na figura 14. Em seguida, o jogador principal mais próximo do gol se torna um goleiro e, como ele está fora do gol, voltará para tal por meio do Behavior Get Back to Goal. Quando o novo goleiro chegar em sua posição, esse técnico é terminado e volta para o técnico que estava anteriormente.

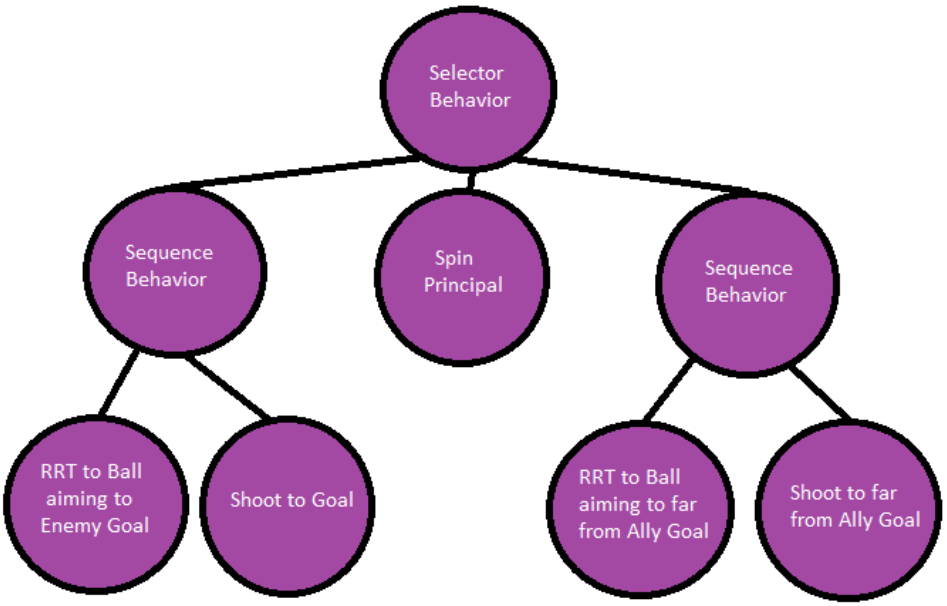

Figura 14: Behavior Tree usada para o goleiro quando este sai do gol.

5 Conclusões

O projeto de Iniciação Científica apresentou bons resultados, especialmente quanto à vitória da equipe do ITA na competição CBR 2018, rendendo o primeiro lugar à equipe, na primeira vez da história, dentre mais de 25 equipes participantes de toda América Latina.

Do ponto de vista técnico, o uso do algoritmo Covariance Matrix Adaptation Evolution Strategy se mostrou factível e funcional para otimização de parâmetros para o planejamento de trajetórias, com resultados demonstrados em partidas oficiais.

Com a estrutura de otimização implementada para a equipe, próximos passos incluem utilizar esse algoritmo para outras áreas do código, como otimização na tomada de decisão do time de robôs.

6 Agradecimentos

- Ao CNPq, pelo apoio financeiro e motivacional.
- 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.
- Ao professor doutor Celso Massaki Hirata, 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.

7 Bibliografia

Referências

[1] M. J. Powell, "The NEWUOA software for unconstrained optimization without derivatives", em *Large-scale nonlinear optimization*, Springer, 2006, pp. 255–297.

[2] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[3] D. Urieli, P. MacAlpine, S. Kalyanakrishnan, Y. Bentor e P. Stone, "On optimizing interdependent skills: A case study in simulated 3D humanoid robot soccer", em *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, International Foundation for Autonomous Agents e Multiagent Systems, 2011, pp. 769–776.

[4] M. Grant, S. Boyd e Y. Ye, *CVX: Matlab software for disciplined convex programming*, 2008.

[5] http://www.cbrobotica. org/wp-content/uploads/2014/03/VerySmall2008_en.pdf.

[6] S. Zickler, T. Laue, O. Birbach, M. Wongphati e M. Veloso, "SSL-vision: The shared vision system for the RoboCup Small Size League", em *Robot Soccer World Cup*, Springer, 2009, pp. 425–436.

[7] Y. Lim, S.-H. Choi, J.-H. Kim e D.-H. Kim, "Evolutionary univector field-based navigation with collision avoidance for mobile Robot", em *Proc. 17th World Congress The International Federation of Automatic Control, Seoul, Korea (July 2008)*, 2008.

[8] Y. Koren e J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation", em *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, IEEE, 1991, pp. 1398–1404.

[9] N. Hansen e A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies", *Evolutionary computation*, vol. 9, n.º 2, pp. 159–195, 2001.

[10] https://developers.google.com/protocol-buffers/docs/cpptutorial.

[11] K. Ahnert e M. Mulansky, "Odeint–solving ordinary differential equations in C++", em *AIP Conference Proceedings*, AIP, vol. 1389, 2011, pp. 1586–1589.

[12] F. Kerger, *OGRE 3D 1.7 Beginner's Guide*. Packt Publishing Ltd, 2010.

[13] U. Egly, G. Novak e D. Weber, *Decision making for MiroSOT soccer playing robots*. na, 2005.

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

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

[16] H. Akiyama e I. Noda, "Multi-agent positioning mechanism in the dynamic environment", em *Robot Soccer World Cup*, Springer, 2007, pp. 377–384.

[17] https://www.qt.io/.