



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



**PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO
CIENTÍFICA - PIBIC**

**Otimização de Movimentos de Robô
Humanoide**

Francisco Muniz de Paula Neto

RELATÓRIO FINAL DE ATIVIDADES

Orientador: Carlos Henrique Costa Ribeiro

08 / 2016



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



**PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO
CIENTÍFICA - PIBIC**

Relatório Final de Atividades

Otimização de Movimentos de Robô

Humanoide

São José dos Campos, __ / __ / ____

Nome do aluno	Francisco Muniz de Paula Neto
Assinatura do aluno	<i>Francisco Muniz de Paula Neto</i>

Nome do orientador	Carlos Henrique Costa Ribeiro
Assinatura do orientador	

**INSTITUTO TECNOLÓGICO DE AERONÁUTICA
PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA -
PIBIC**

Formulário de Aprovação de Relatório pelo Orientador

Relatório: Relatório Parcial Relatório Final

1- CONSIDERO O RELATÓRIO APROVADO COM BASE NOS SEGUINTE ASPECTOS

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

Local e data:

Assinatura do Orientador:

CONTEÚDO

1	Resumo do Plano Inicial	5
2	Resumo das Atividades Realizadas	5
3	Descrição do Problema	5
3.1	Problema de Otimização	5
3.2	Representação dos Movimentos Keyframe	5
3.3	Movimentos Keyframe	6
3.4	Framework de Otimização	7
3.5	Cluster	9
4	Resultados Obtidos	9
4.1	Otimizações feitas	9
5	Conclusões	13
6	Agradecimentos	13

LISTA DE FIGURAS

Figura 1	Movimento keyframe utilizando interpolação por spline cúbica. Cada cor representa a trajetória de uma junta durante o movimento.	7
Figura 2	RoboCup 3D Soccer Simulation com imagens retiradas no Monitor Roboviz.	7
Figura 3	Diagrama do framework de otimização.	8
Figura 4	Foto do cluster utilizado.	9
Figura 5	Evolução dos valores melhores e medianos de cada iteração do algoritmo de otimização para o movimento StandUpFromBack.	11
Figura 6	Segunda Evolução dos valores melhores e medianos de cada iteração do algoritmo de otimização para o movimento StandUpFromBack.	11
Figura 7	Representação do movimento StandUpFromBack em quadros.	12
Figura 8	Diagrama de referência para o campo.	12
Figura 9	Evolução dos valores melhores e medianos de cada iteração do algoritmo de otimização para o movimento KickRightLeg.	12
Figura 10	Segunda evolução dos valores melhores e medianos de cada iteração do algoritmo de otimização para o movimento KickRightLeg	13

LISTA DE TABELAS

Tabela 1	Comparação entre os movimentos otimizado e não otimizado.	10
----------	---	----

RESUMO DO PLANO INICIAL

O plano inicial era focado em fazer um estudo sobre os movimentos a serem otimizados e, utilizando-se do trabalho anteriormente desenvolvido, para paralelizar os algoritmos e fazer a otimização rodar mais rápido, rodar os algoritmos de otimização sobre tais movimentos e melhorar os resultados do time de Soccer3D do time da ITAndroids. Deveria-se utilizar uma estratégia para reduzir a quantidade de variáveis a serem otimizadas, para acelerar a otimização.

RESUMO DAS ATIVIDADES REALIZADAS

Selecionou-se o uso de movimentos keyframe para otimização, dando maior disponibilidade para o robô

Foram realizadas várias otimizações, para fazer com que fosse possível do time ter movimentos robustos, que não falhassem numa situação de jogo. Não foi possível fazer comparações numéricas sobre o time utilizado pois o time não estava em condições apropriadas para rodar partidas contra outros times, devido a defeitos recentemente solucionados.

DESCRIÇÃO DO PROBLEMA

Problema de Otimização

Um problema de otimização matemática consiste em achar o vetor x que minimiza ou maximiza uma certa função objetivo $f(x)$. Será considerado que estamos interessados apenas em minimizar uma certa função, chamada função de custo, dado que maximizar uma função $f(x)$ pode ser feito minimizando $-f(x)$.

Algoritmos de Otimização Meta Heurística mostram resultados muito bons em resolver problemas de otimização na prática, como pode ser visto em [?] e [?]. Tais algoritmos normalmente necessitam de pouco conhecimento sobre a função que estão tentando otimizar, sendo necessário computar $f(x)$ dado x . Um problema de tais algoritmos é a falta de garantia de convergência para o mínimo global, e uma convergência para um mínimo local costuma ser vista em tais algoritmos.

Representação dos Movimentos Keyframe

Keyframe

Um keyframe $k = (j_1, j_2, \dots, j_n) \in K \subseteq \mathbb{R}^n$ é um conjunto ordenado de posições angulares de junta, onde K e n são, respectivamente, o espaço de juntas e o número de graus de liberdade do robô.

Passo de Keyframe

Um passo de keyframe é um par $s = (k, t) \in S = K \times \mathbb{R}$, onde t representa o tempo em que o keyframe deve ser alcançado em relação ao início do movimento.

Movemento Keyframe

Um movimento keyframe, ou simplesmente um movimento, é definido como $\mathbf{m} = (\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_\gamma, r) \in M = S^\gamma \times \mathbb{R}$, onde γ é o número de passos associados ao movimento e r representa a taxa de velocidade do movimento. Além disso, sempre se considera um movimento começando no tempo 0, i.e. $t_1 = 0$. Por isso, cada tempo t_i representa o tempo em relação ao início do movimento.

Baseia-se a representação de tais keyframes pelas dadas em [?] e [?].

Movimentos Keyframe

Movimentos keyframe rodam em um sistema de controle aberto, onde as posições das juntas são obtidas pela interpolação de passos de keyframe, baseando-se no tempo atual da simulação. As juntas do NAO simulado do simulador Simspark são juntas de velocidade, portanto são usados simples controladores proporcionais para cada junta, para acompanhar as posições de juntas, que são necessárias para rodar o movimento keyframe.

Para obter posições de juntas que variam suavemente, interpolam-se os passos de keyframe utilizando splines cubicas como vistas em [?], que garantem derivadas de primeira e segunda ordem contínuas. Fig. 1 mostra as posições de juntas obtidas a partir da interpolação por splines cubicas; passos de keyframe são mostrados em preto.

Note que achar a interpolação por spline cúbica requer resolver um problema de sistema de equações lineares tridiagonais que pode ser resolvido em $O(n)$. Entretanto, dado que passos de keyframe são fixos para cada movimento, a interpolação pode ser computada offline. Com isso, achar a interpolação atual pode ser achada em $O(1)$ dado que os passos de keyframe são utilizados sequencialmente durante o movimento: Mantém-se um iterador para a interpolação atual e atualiza-se esse iterador quando o passo atual chega ao fim.

Para considerar a taxa de velocidade r do movimento, faz-se uma escala de tempo, multiplicando o tempo atual por r . Portanto, se ocorreu a passagem de um tempo τ em relação ao começo do movimento, o tempo utilizado na interpolação é $r\tau$.

Por fim, quando um movimento keyframe é requisitado, as juntas inicias do robô costumam estar bem distantes das posições iniciais do movimento. Isso pode gerar alta aceleração nas juntas, que, muitas vezes, faz o robô perder equilíbrio. Portanto, um suavizador de movimento foi implementado para mover de maneira suave as juntas iniciais do robô para as juntas do primeiro passo do keyframe.

Para se conseguir reduzir a probabilidade de uma queda, o suavizador de movimentos sincroniza as juntas para faze-las mover a velocidades proporcionalmente às distâncias angulares enquanto se evita violar a velocidade máxima das juntas. Quando a máxima distância angular de junta é menor que um certo valor, o suavizador para de gerar posições de juntas e o movimento keyframe começa o primeiro passo.

Otimização de Movimentos Keyframe

No caso particular de otimização de movimentos keyframe, o vetor \mathbf{x} é dado pelos parâmetros associados ao movimento keyframe \mathbf{m} . Além disso, precisa-se definir uma função de custo $f(\mathbf{x})$ que codifica a saída desejada do

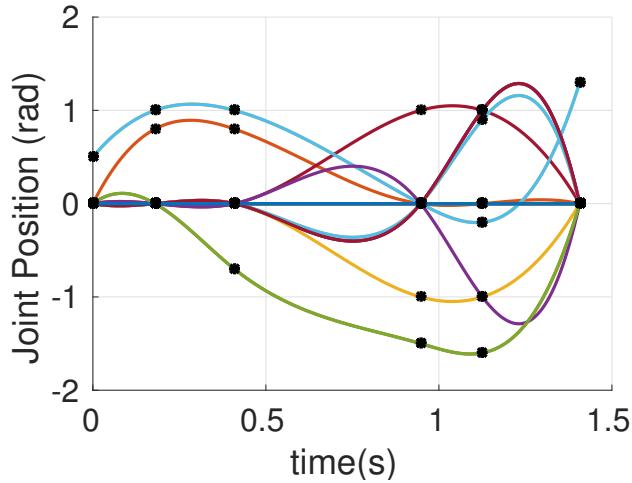


Figura 1: Movimento keyframe utilizando interpolação por spline cúbica. Cada cor representa a trajetória de uma junta durante o movimento.

movimento keyframe (e.g. Chutar a bola o mais longe possível). dado que a saída do movimento keyframe depende de uma simulação, não é possível escrever uma expressão analítica para a função $f(x)$. Portanto, resolver tal problema de otimização é particularmente complexo, exigindo uma boa avaliação da simulação do movimento.

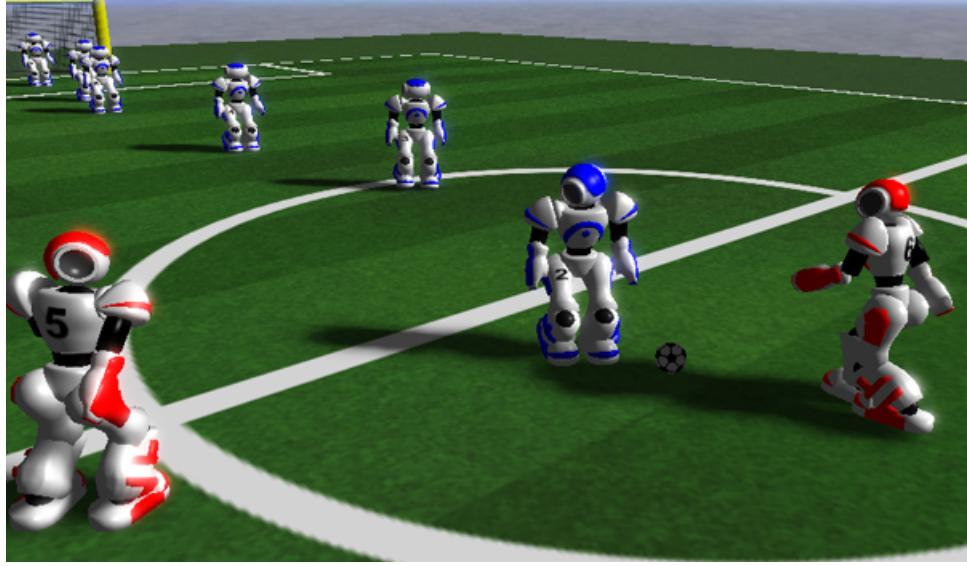


Figura 2: RoboCup 3D Soccer Simulation com imagens retiradas no Monitor Roboviz.

Framework de Otimização

O framework contém os seguintes algoritmos de otimização implementados em MATLAB: Hill Climbing (HC), Particle Swarm Optimization (PSO), e Covariance Matrix Adaptation Evolution Strategy (CMA-ES). O algoritmo HC foi implementado como visto em [?] e o PSO tal como em [?]. A implemen-

tação do CMA-ES é baseada na dada em [?]. Uma descrição do algoritmo de CMA-ES pode ser encontrada em [?].

Pode-se notar que na otimização de movimentos keyframe, cada avaliação da função de custo requer uma simulação, portanto tal avaliação é o gargalo em tempo da otimização, e implementar tais algoritmos em uma linguagem mais rápida, tal como C++, não iria reduzir significativamente o tempo de otimização.

O servidor de otimização acelera a otimização por paralelização. A arquitetura do servidor é apresentada na Fig. 3.

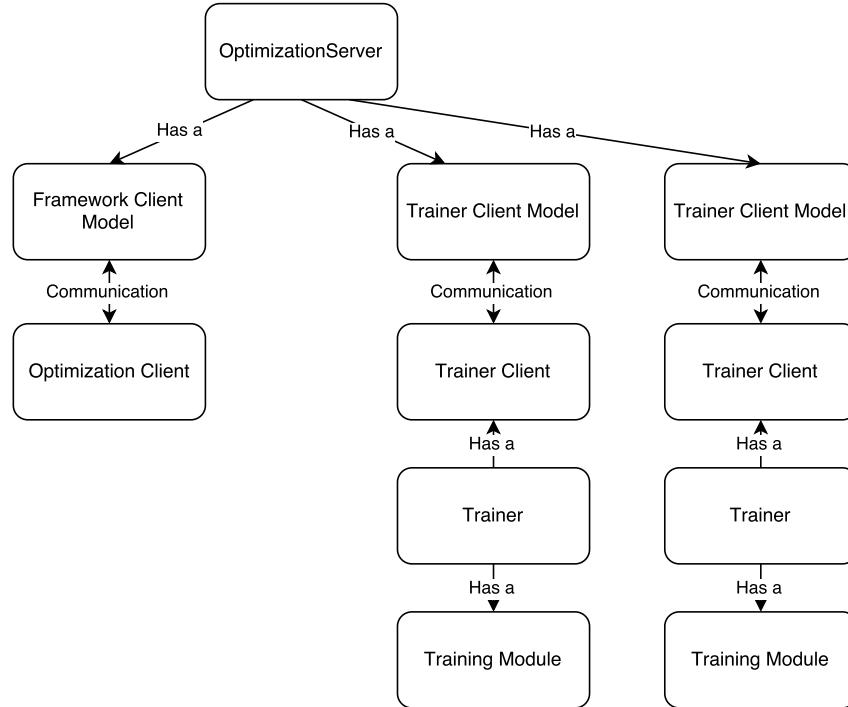


Figura 3: Diagrama do framework de otimização.

O **OptimizationServer** recebe uma matriz $\mathbf{X} \in \mathbb{R}^{k \times p}$ do **OptimizationClient**, onde o algoritmo de otimização está rodando, e quebra tal matriz em p vetores $\mathbf{x}_i \in \mathbb{R}^k$, $i = 1, \dots, p$, de tal maneira que cada vetor \mathbf{x}_i representa um ponto no espaço de otimização (ou também pode ser chamado de partícula) que contém informação suficiente para avaliar o custo da função $f(\mathbf{x}_i)$.

O servidor funciona em um ambiente utilizando multiplas threads, onde cada vetor \mathbf{x}_i a ser avaliado é enviado para uma thread T_i . Cada Thread do servidor comunica-se com um **Trainer** usando comunicação por socket de Linux. Um **Trainer** recebe um vetor \mathbf{x}_i e usa um **TrainingModule** para computar o conjunto de saídas \mathbf{o}_i , que serão utilizadas pelo **OptimizationClient** para computar o valor de custo. O **TrainingModule** funciona como uma Caixa-preta para o respectivo **Trainer**. Portanto, um modulo de treinamento pode computar o conjunto de saídas de maneira arbitrária, incluindo situações em que é necessária uma simulação para calcular tais saídas.

As saídas são reunidas em uma matriz $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_p]$ pelo servidor, e retornadas para o **OptimizationClient**. Com isso, o **OptimizationClient** usa a matriz \mathbf{O} para computar o vetor de custos $\mathbf{c} = [c_1, c_2, \dots, c_p] \in \mathbb{R}^p$, que é finalmente usado para atualizar os algoritmos de otimização e iniciar uma nova iteração dos algoritmos.

Dada tal arquitetura, para cada problema de otimização, será preciso apenas a criação de um novo módulo de treinamento, e um avaliador da função de custo. Isso pode ser facilmente realizado no framework, utilizando boas práticas de programação, tais como técnicas de Programação Orientada à Objetos.

Cluster

Pode-se observar o cluster utilizado na Fig. 4. O cluster utilizado é heterogêneo, tendo 4 computadores IPC2 i7, e 3 NUCs i5, sendo tais computadores, juntamente com o framework de otimização, utilizados para acelerar as otimizações.

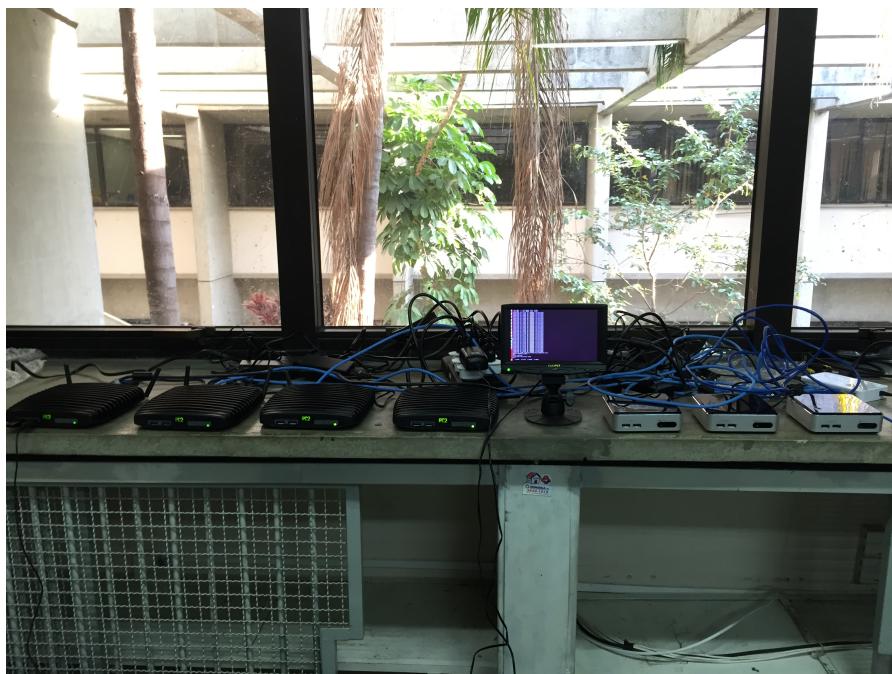


Figura 4: Foto do cluster utilizado.

RESULTADOS OBTIDOS

Otimizações feitas

Movimento de levantar de costas

O movimento $\mathbf{m}_{\text{standback}} \in M$ foi inicialmente com $\gamma = 6$. O movimento inicial era suficiente para fazer o robô levantar, mas não tinha estabilidade. Portanto, decidiu-se manter as posições de juntas e otimizar apenas os espaços de tempo. Com isso, os parâmetros selecionados para a otimização foram os valores $\mathbf{x} = [r, t_2, \dots, t_7]^T \in \mathbb{R}^7$, onde \mathbf{x} é um ponto no espaço de otimização.

Sendo a simulação do Simpark intrinsecamente ruidosa, decidiu-se avaliar cada posição do espaço 10 vezes e tirar uma média dos valores de custo para diminuir o ruído. A função de custo é computada usando (1).

Tabela 1: Comparação entre os movimentos otimizado e não otimizado.

Movimento	Fração de sucesso	tempo(s)
Não Otimizado	0.80100	1.7
Otimizado	0.93400	1.7

$$f(x) = -w_{fall}b_{fall} + w_{time}\Delta t - w_{\bar{z}_{torso}}\bar{z}_{torso} \quad (1)$$

onde b_{fall} é um inteiro associado à informação do robô ter caído ou não, sendo -1 se o robô cair, e 1 se não cair, Δt é o tempo que o robô gasta para tentar levantar, e \bar{z}_{torso} é a altura média do torso do robô, sendo um parâmetro importante para descartar movimentos que ficam muito tempo parado no chão. Além disso, os pesos $w_{fall} = 50$, $w_{time} = 10$, e $w_{\bar{z}_{torso}} = 1$ estão associados com o indicador de queda b_{fall} , o tempo decorrido Δt e a altura média \bar{z}_{torso} , respectivamente. Dada essa função de custo, o conjunto de saídas para esse problema de otimização é dado por $\theta = [b_{fall}, \Delta t, \bar{z}_{torso}]^T$. Deseja-se deixar $b_{fall} = 1$, diminuir o tempo de levantar, e aumentar o valor de \bar{z}_{torso} para garantir que o robô ficou pouco tempo parado no chão.

O módulo de treinamento executou o procedimento descrito a seguir para computar a função de custo:

1. O robô é inicializado em uma posição fixa no campo.
2. Um movimento keyframe para fazer o robô cair com as costas para trás é executado.
3. O StandUpFromBack movement é executado para fazer o robô tentar levantar.
4. As saídas necessárias para calcular a função de custo são computadas.

A Fig. 5 mostra a evolução dos valores melhores e medianos do conjunto de partículas avaliadas a cada iteração da otimização do movimento StandUpFromBack. O valor inicial encontrado é bom, entretanto o algoritmo CMA-ES evita uma convergência prematura para encontrar uma solução melhor, explorando melhor o espaço, e, percebe-se que o algoritmo consegue achar uma solução mais estável. A tabela 1 mostra dados comparando o movimento inicial e o otimizado, onde cada movimento é executado 1000 vezes. Nota-se que o movimento é mais estável que o inicial.

Em uma segunda otimização, conseguiu-se atingir um movimento um pouco mais estável, como pode ser visto na Fig. 6, usando a função de custo (2), onde $w_{fall} = 20$ e $w_{time} = 10$.

$$f(x) = -w_{fall}b_{fall} + w_{time}\Delta t \quad (2)$$

O movimento otimizado é descrito na Fig. 7.

Chute

O movimento KickRightLeg, para chutar a bola com a perna esquerda, $m_{kick} \in M$ tem $\gamma = 10$. A otimização foi feita considerando os passos de tempo e a taxa de velocidade como variáveis, portanto uma posição no espaço de otimização é dada por $x_i = [r, t_2, \dots, t_{\gamma=10}]^T$. As juntas não são consideradas na otimização por que a dimensionalidade do espaço de otimização seria muito grande. De maneira similar à otimização do movimento

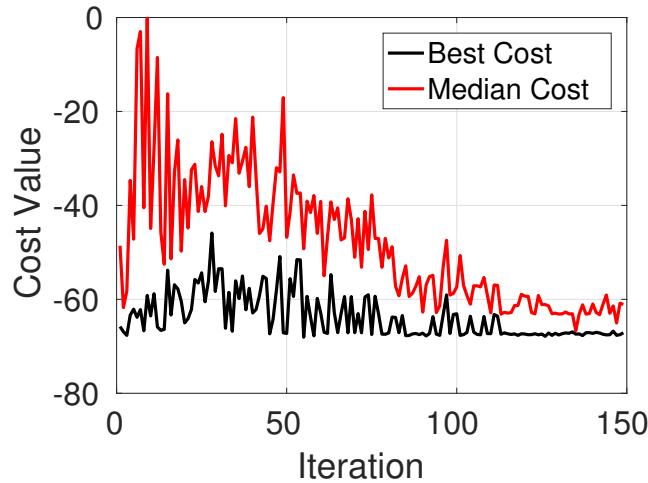


Figura 5: Evolução dos valores melhores e medianos de cada iteração do algoritmo de otimização para o movimento StandUpFromBack.

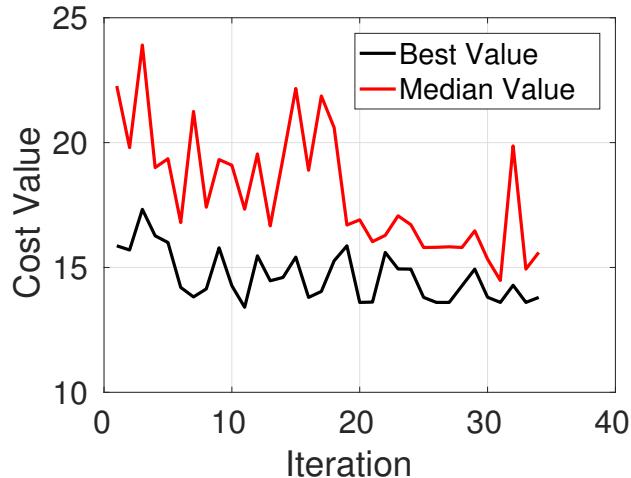


Figura 6: Segunda Evolução dos valores melhores e medianos de cada iteração do algoritmo de otimização para o movimento StandUpFromBack.

StandUpFromBack, cada posição foi avaliada 10 vezes utilizando a função de custo dada em (3).

$$f(\mathbf{x}) = -w_{\text{fall}} * b_{\text{fall}} - w_x x + w_y |y| - w_{\bar{z}_{\text{ball}}} \bar{z}_{\text{ball}} \quad (3)$$

Onde \bar{z}_{ball} é a altura média da bola, e é considerada pois se deseja um chute alto para não atingir adversários, e x e y são as coordenadas finais da posição da bola, respectivamente. O sistema de coordenadas é o mesmo utilizado pelo simulador do Simspakr, que é apresentado na Fig. 8. Além disso, os pesos $w_x = 10$, $w_y = 10$, e $w_{\bar{z}_{\text{ball}}} = 5$ estão associados aos dados de x , y , e \bar{z}_{ball} , respectivamente. Nesse caso, o modulo de treinamento rodou a seguinte simulação 10 vezes para cada partícula:

1. O robô foi inicializado na posição $(-0.2, 0.1)$.
2. A bola foi colocada na posição $(0, 0)$.
3. Então o robô tentou executar o movimento KickRightLeg.

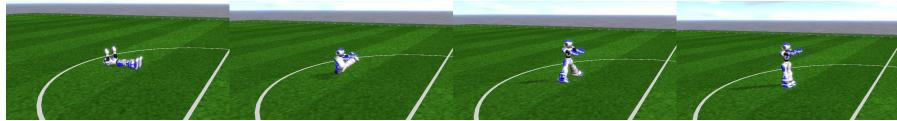


Figura 7: Representação do movimento StandUpFromBack em quadros.

4. Os dados necessários para calcular o custo da função são armazenados.

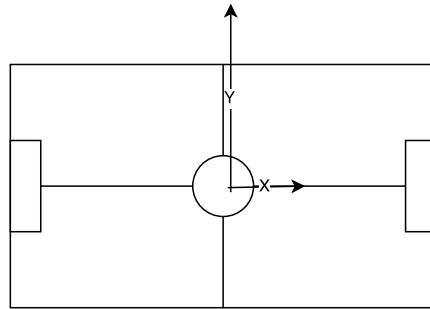


Figura 8: Diagrama de referência para o campo.

A Fig. 9 mostra a evolução dos valores melhores e medianos do conjunto de posições avaliadas a cada iteração da otimização para o movimento KickRightLeg.

Além desse chute inicial adquirido, conseguiu-se outro chute, que apresentava melhores resultados, e otimizou-se sobre tal chute, tendo os valores apresentados na Fig. 10, com a função de custo dada em (4), onde $x_g = 10, y_g = 0, w_{exp} = 100, w_x = 10$ e $w_y = 5$ são a posição objetivo em x, em y, o peso associado à exponencial, à diferença em x e a diferença em y.

$$f(x) = -w_{exp}e^{-| (w_x(x-x_g), w_y(y-y_g)) |} \quad (4)$$

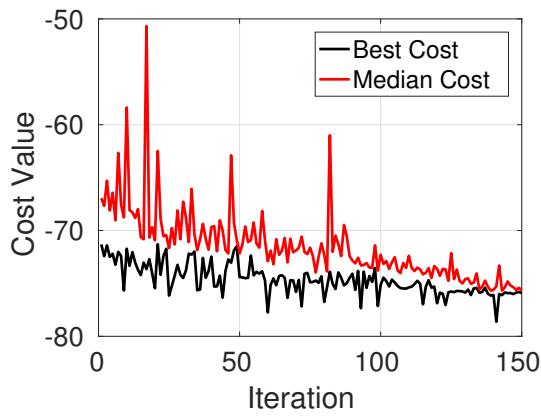


Figura 9: Evolução dos valores melhores e medianos de cada iteração do algoritmo de otimização para o movimento KickRightLeg.

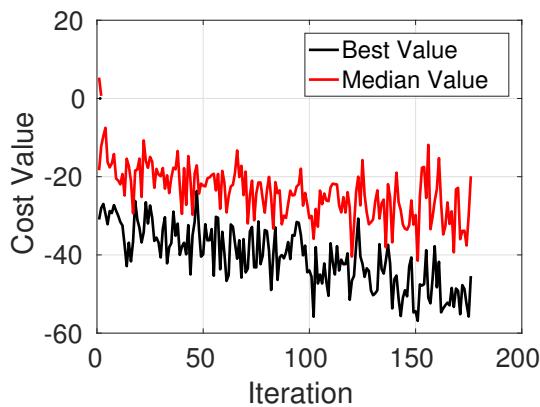


Figura 10: Segunda evolução dos valores melhores e medianos de cada iteração do algoritmo de otimização para o movimento KickRightLeg .

CONCLUSÕES

Percebe-se que, com o framework desenvolvido, conseguiu-se diminuir o tempo necessário para a otimização dos movimentos de robô humanoide, resolvendo de maneira simples e de fácil implementação.

Além disso, pode-se ver a melhoria do Time da ITAndroids, com tais otimizações feitas, ajudando o time de Soccer3D a ficar mais competitivo.

AGRADECIMENTOS

Gostaria de agradecer ao CNPQ, pela oportunidade oferecida e o auxílio dado à pesquisa realizada. Agradece-se também ao professor Carlos Henrique Costa Ribeiro, professor orientador da iniciação científica, por todo o apoio dado à pesquisa, e ajuda no desenvolvimento do projeto.

Agradecimentos a equipe de robótica ITAndroids, especialmente o time de Soccer3D juntamente com o Professor Marcos Ricardo Omena de Albuquerque Maximo, que ajudaram no desenvolvimento das ferramentas necessárias para a pesquisa. Agradecimentos ao professor Paulo André Lima de Castro por oferecer o uso do Frank Cluster, um cluster de computadores da Divisão de Computação do ITA, que ajudou no início do projeto.

Agradecimentos a Divisão de Computação do ITA, e especialmente ao LAB-SCA, que forneceu o equipamento para o cluster para as otimizações finais e ao técnico Carlos que ajudou na montagem do cluster.