

# Otimização de KeyFrames de chutes no Futebol de Robôs: CMA-ES e a liga Soccer 3D

## Exame de Inteligência Artificial Aplicada à Robótica Móvel (CT-213)

Bruno Benjamim Bertucci  
Instituto Tecnológico de Aeronáutica  
São José dos Campos, Brasil  
Email: bertucci@ita.br

Lucas José Velôso de Souza  
Instituto Tecnológico de Aeronáutica  
São José dos Campos, Brasil  
Email: lucasjvs@ita.com

Guilherme Nascimento de Oliveira  
Instituto Tecnológico de Aeronáutica  
São José dos Campos, Brasil  
Email: guilherme.oliveira@ga.ita.br

## 1. Introdução e Motivação

Em 1997, diversos professores universitários propuseram um desafio que buscassem promover tanto a robótica como os estudos em inteligência artificial, de modo elegante e atraente aos jovens da época. Assim, deu-se o surgimento da *Robot Soccer World Cup*, com objetivo principal de possibilitar que, em meados do século 21, um time de robôs humanoides possa vencer o time humano campeão da Copa do Mundo da Federação Internacional de Futebol e Associações (FIFA).

Uma das categorias da *Robot Soccer World Cup* (RobotCup) é a *3D Soccer Simulation*, uma categoria de robôs simulados. Nesse contexto, emula-se a realidade das leis físicas no computador: gravidade, atrito, torque, massa... . Dessa maneira, conseguir fazer os robôs andarem, chutarem, se comunicarem nessa realidade virtual torna-se uma problemática cuja resolução é tremamente difícil, pois envolve tarefas e otimizações tais quais o mundo real. Dentre os muitos times que participam da competição, se encontra a *ITAndroids*, uma iniciativa de robótica do *Instituto Tecnológico de Aeronáutica* (ITA).

No último ano, foram introduzidas na competição, diversos mecanismos que exploram e incentivam tecnologias que auxiliem passes de bola entre robôs. Nesse sentido, para promover essa interação, bem como esses passes, decidiu-se pensar em estratégias que os utilizem. Uma situação possível é um chute de um dos agentes, já dentro do campo inimigo, para a área do goleiro adversário (um chute de média distância). Paralelamente, apoiados com o modo *PasseModeOn* imposto pela *Robot Soccer World Cup*, no qual o robô fica livre de oponentes para chutes certeiros, mas que não podem fazer gols, pensou-se na possibilidade de um lançamento que não efetivamente realizasse gol, mas que deixasse a bola muito próxima dele, de maneira que outro agente pudesse fazê-lo.

Decidiu-se, com isso, a criação de um novo KeyFrame de chute, que atendesse a todas essas demandas. Por essa perspectiva, optou-se por já se utilizar um modelo inicial de chute (mais próximo do que era desejado) e otimizá-lo para as condições que foram apresentadas. Para tanto, utilizou-

se um conhecido algoritmo meta-heurístico de otimização chamado *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES).

## 2. Revisão Teórica

O Algoritmo de Estratégia Evolutiva de Adaptação de Matriz de Covariância, ou, em inglês, *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES), é um algoritmo meta-heurístico de otimização, embasado em população e distribuição de probabilidade. O algoritmo se fundamenta na amostragem gaussiana multivariada para explorar o espaço vetorial dos parâmetros, buscando a melhor combinação, dada uma métrica (Função de Custo). Assim, para alcançar os melhores combinações de parâmetros (otimização), o algoritmo altera, a cada iteração, as variáveis da distribuição (média, covariância e passo), por meio das melhores amostras (seleção). Entretanto, devido à complexidade do algoritmo, é necessário, revisitar alguns conceitos estatísticos antes de apresentá-lo.

### 2.1. Definições Estatísticas

**2.1.1. Matriz de covariância ( $\Sigma$ ):** é uma matriz simétrica na qual há todas as covariâncias (variâncias entre as variáveis envolvidas) dado um vetor  $\mathbf{X}$  de variáveis aleatórias. Matematicamente, pode ser expressa por meio do vetor  $\mathbf{X}$ , bem como a esperança desse vetor  $E[\mathbf{X}]$ , da seguinte forma:

$$\Sigma = E[(\mathbf{X} - E[\mathbf{X}])(\mathbf{X} - E[\mathbf{X}])^T]$$

**2.1.2. Distribuição Gaussiana Multivariada:** pode ser considerada uma generalização da distribuição normal para uma única dimensão. Assim, é um modo de gerar vetores nos quais cada dimensão segue uma distribuição gaussiana. Pode ser obtida por meio do vetor média,  $\mu$ , e da matriz de covariância da distribuição  $\Sigma$  da seguinte maneira:

$$p(\mathbf{x}) = ((2\pi)^n \det(\Sigma))^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

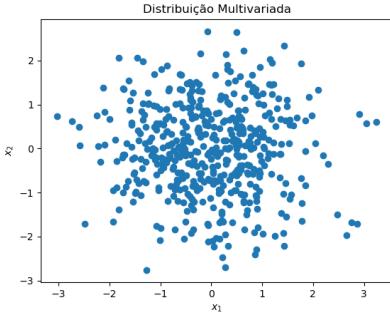


Figure 1: Exemplo de Distribuição Multivariada Bidimensional. Na imagem, foram utilizados como vetor média o vetor nulo e como matriz de covariância, a matriz identidade de segunda dimensão.

## 2.2. Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

O algoritmo trabalha com gerações a cada iteração. De modo geral, a cada amostragem, realiza-se uma distribuição multivariada utilizando-se, além da média das melhores  $N$  (hiperparâmetro) amostras obtidas na geração anterior (seleção) e da matriz de covariância, também o passo  $\sigma$ , um outro hiperparâmetro do algoritmo, que é atualizado em cada uma das iterações. As principais etapas de cada ciclo do algoritmo são:

**2.2.1. Atualização da População.** As amostras de cada geração são obtidas por:

$$\mathbf{x}_k^{(g+1)} = m^{(g)} + \sigma^{(g)} N(\mathbf{0}, \Sigma^{(g)})$$

Onde  $\mathbf{x}_k^{(g+1)}$  é o k-ésimo vetor da geração  $(g+1)$ ,  $m^{(g)}$  é a média da geração  $(g)$ ,  $\sigma^{(g)}$  o passo da geração  $(g)$ ,  $\Sigma^{(g)}$  é a matriz de covariância da geração  $(g)$  e  $N(\mathbf{0}, \Sigma^{(g)})$  é uma distribuição gaussiana multivariada, cuja média é o vetor nulo e a matriz de covariância é a da geração anterior.

**2.2.2. Atualização do Vetor Média.** O vetor média é atualizado a cada geração segundo a equação:

$$m^{(g+1)} = \sum_{i=1}^N w_i \mathbf{x}_i^{(g)}$$

Onde cada  $w_i$  são pesos, cujos valores são definidos segundo as heurísticas em [1].

**2.2.3. Atualização da Matriz de Covariância.** Para a atualização da matriz de covariância, cria-se um vetor auxiliar  $p_c$ , que também é atualizado a cada iteração, conforme:

$$p_c^{(g+1)} = (1 - c_c)p_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{eff}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}$$

Onde  $c_c$  e  $\mu_{eff}$  são hiperparâmetros estimados segundo diversas heurísticas [1].

Assim, a atualização da matriz de covariância se dá por:

$$\Sigma^{(g+1)} = (1 - c_\mu - c_1)\Sigma^{(g)} + c_1 p_c^{(g)} (p_c^{(g)})^T + c_\mu \sum_{i=1}^N w_i y_{i:\lambda} y_{i:\lambda}^T$$

Onde  $y_{i:\lambda}$  é dado por:

$$y_{i:\lambda} = \frac{\mathbf{x}_i^{(g)} - m^{(g)}}{\sigma^{(g)}}$$

Além disso, tanto  $c_\mu$  como  $c_1$  também são estimados usando heurísticas conforme [1].

**2.2.4. Atualização do Passo.** Para a atualização do passo, cria-se um vetor auxiliar  $p_\sigma$ , que também é atualizado a cada iteração, conforme:

$$p_\sigma^{(g+1)} = (1 - c_\sigma)p_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}} \Sigma^{-\frac{1}{2}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}$$

Onde  $c_\sigma$  é um hiperparâmetro estimado segundo diversas heurísticas [1].

Assim, a atualização do passo se dá por:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp \left( \frac{c_\sigma}{d_\sigma} \left( \frac{\|p_\sigma^{(g)}\|}{\|N(\mathbf{0}, I)\|} - 1 \right) \right)$$

Onde  $d_\sigma$  é um outro hiperparâmetro estimado segundo heurísticas [1].

## 3. Implementação

A implementação do algoritmo CMA-ES foi feita por meio da biblioteca *pycma* [2], escrita em *Python*.

Em cada episódio, o vetor de parâmetros é convertido por meio de um script especializado para um Keyframe que pode ser lido pelo robô. Cada episódio é composto por 6 testes executados em sequência executados no lado direito do campo, com 2 testes em cada uma de 3 posições diferentes: na linha perpendicular ao centro do gol e 45° acima e abaixo, todas a uma distância de 8 unidades do centro do gol. Em todos os casos, o agente foi inicializado 5 unidades à direita do centro do campo e deve andar até a posição ideal para chutar a bola, conforme figura 2. A pontuação final é a soma das pontuações obtidas nos 6 testes.

Caso o robô cumpra o objetivo de chutar a bola para dentro da área, foi atribuída uma pontuação de 50. Para priorizar o chute a gol em relação às demais situações que não são o objetivo da otimização, foi atribuída uma pontuação de 25 em caso de gol.

Há a chance de o keyframe gerado ser disfuncional. Nesse caso, o mais provável que aconteça é que o robô caia e a bola se move nada ou muito pouco. A pontuação atribuída nessa situação é -1.

Em todos os casos, são somados ou subtraídos alguns fatores contínuos para estabelecer um critério de desempate: somada metade da altura máxima alcançada pela bola, subtraída a distância até o gol e subtraído metade do desvio em

relação à linha reta entre a posição inicial da bola e o centro do gol. O tempo não foi considerado pois depende de muitos fatores, a maioria deles aleatórios ou não relacionados ao *keyframe*.

Por padrão, o pacote *pycma* considera uma função custo, ou seja, prioriza os menores valores. Portanto, a recompensa calculada é multiplicada por -1 antes de ser enviada ao *pycma*.

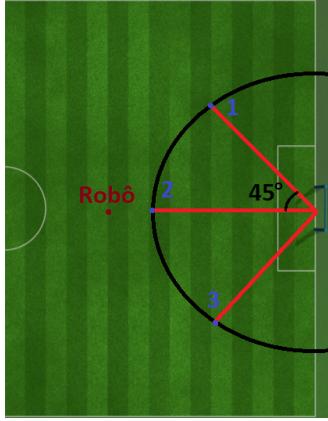


Figure 2: Diagrama do treinamento.

### 3.1. Execução do otimizador

Para que se execute o otimizador, é necessário que algumas dependências estejam instaladas devidamente no computador: *simspark* e *rcssserver3d*, que lidam com o ambiente simulado da categoria *Soccer 3D*, e o *RoboViz*, que permite a visualização do ambiente e dos agentes.

Uma vez instaladas, a otimização pode ser iniciada através do arquivo *main.py*. Após o término da otimização, mesmo que por encerramento forçado do processo, o arquivo *replace\_keyframe.py* deve ser executado para substituir os arquivos *.json* contendo os parâmetros do *keyframe* pelos parâmetros com melhor desempenho encontrados durante toda a otimização.

Para avaliar o desempenho do movimento otimizado, o arquivo executável *KickKeyframeEvaluation\_AgentTest* pode ser usado para tirar medidas como a distância percorrida ao longo de 20 tentativas de chute, gerando um arquivo de texto com esses resultados.

## 4. Resultados e discussão

O programa em *Python* para realizar a otimização do *keyframe* de chute foi executado. Os valores iniciais dos parâmetros foram extraídos de um movimento de chute de longa distância que estava otimizado para uma outra variação do agente NAO, sem o dedão no pé. Por conta dessa diferença no corpo do agente, esse mesmo movimento inicialmente não funcionava de modo efetivo com o NAO com um dedão, como pode ser observado na Figura 3. O

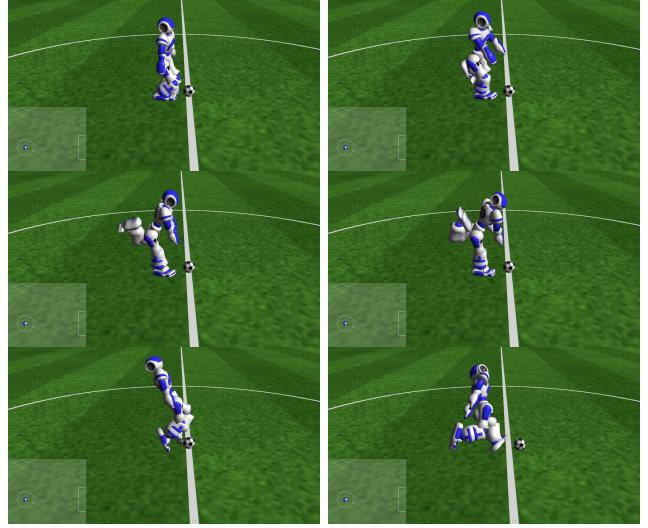


Figure 3: Movimento de chute inicial.

alvo dessa otimização foi escolhido como sendo um chute de média distância adequado para esse NAO com o dedão.

A evolução do treinamento, que ocorreu de maneira acelerada devido à retirada do limite da velocidade do servidor *simspark*, foi observada. Um gráfico da função de custo, dada como a função recompensa com sinal invertido, pela quantidade de variações geradas, onde 20 variações compreendem uma geração do CMA-ES, está ilustrado na Figura 4.

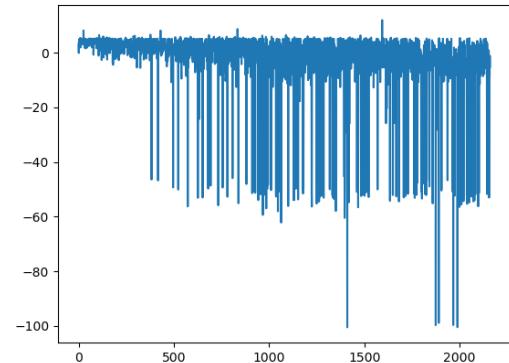


Figure 4: Gráfico da função de custo.

Pela observação do treinamento e do gráfico acima, foi possível perceber que, inicialmente, os parâmetros tinham uma variação significativa, mesmo com o baixo valor do parâmetro  $\sigma^{(0)}$ . Isso levava o agente a se desequilibrar durante a execução do movimento com bastante frequência.

Dessa forma, nos primeiros episódios de treinamento, foram obtidas poucas variações do chute nas quais o agente era capaz de movimentar a bola. Com a passagem dos episódios, essas variações deram lugar a outras que ao menos conseguiam atingir a bola, mesmo que a distância

percorrida por ela fosse baixa, ou com um desvio significativo com relação à direção do gol.

Foram necessárias por volta de 400 iterações, ou 20 gerações, para que o agente produzisse um chute capaz de levar a bola até a área do gol. Em seguida, esses chutes foram se tornando mais comuns. O melhor movimento encontrado ocorreu após cerca de 1400 iterações e foi o movimento final escolhido pelo programa de otimização, o qual possibilitou o agente a levar a bola até a área do gol duas vezes do total de seis vezes que ele foi testado.

Uma ilustração desse movimento otimizado encontra-se na Figura 5. Visuamente, percebe-se que os movimentos são bastante semelhantes até os últimos instantes, porém, o desempenho dessa chute, agora adaptado ao corpo no NAO com dedão, melhorou significativamente, como mostra as Tabelas 1 e 2, sendo que o eixo  $x$  representa a direção da reta que liga a posição original da bola ao gol, e o eixo  $y$  indica a direção perpendicular a aquela.

Chute	Eixo	Distância média	Desvio padrão
Original	$x$	1.37530955	1.37415330
Otimizado	$x$	8.19832600	2.90522037
Original	$y$	0.33583000	0.49866544
Otimizado	$y$	-1.22503340	2.55828937

TABLE 1: Comparação da distância percorrida pelos chutes original e otimizado.

Chute	Altura média	Desvio padrão
Original	0.05707245	0.11816331
Otimizado	0.13790945	0.31548648

TABLE 2: Comparação da altura máxima chutes original e otimizado.

A distância média do chute no eixo  $x$  aumentou significativamente, apesar de seu desvio padrão maior apontar que seu desempenho não é consistente. A distância média no eixo  $y$  é negativa, portanto, indica que o chute otimizado tenta a estar desviado para a direita. Percebeu-se também que o chute otimizado não foi capaz de levar a bola a uma altura significativa. Levando esses aspectos em conta, e comparando ao chute original, os resultados do chute otimizado são satisfatórios, porém, foram notados aspectos que poderiam ter levado a uma otimização melhor.

No aspecto do programa de otimização, uma complicação encontrada foi que o critério de parada utilizado era um valor suficientemente baixo da maior diferença entre os valores da recompensa entre duas gerações. Esse critério dificilmente seria atingido, pois mesmo após 2000 iterações, ainda havia uma variação muito grande na recompensa entre as amostras. Isso ocorreu porque ainda haviam várias amostras que levavam o agente a se desequilibrar e, ao mesmo tempo, a diferença na recompensa entre uma amostra que atingia a área do gol e uma que não atingia era muito grande.

Tendo isso em vista, um critério de parada melhor poderia ter sido um limite máximo para a quantidade de

gerações que se passaram sem que houvesse uma alteração da melhor amostra encontrada ao longo do treinamento.

Em termos do cálculo da recompensa, algumas alterações poderiam ter resultado em um chute melhor, como um peso maior para a altura máxima atingida pela bola.

Finalmente, foi feita a escolha de que o agente deveria caminhar até a bola e chutar de três posições equidistantes do gol na otimização. O intuito dessa escolha era adaptar o movimento de chute a desvios na chegada do agente à posição desejada para realizá-lo, o que, por sua vez, está relacionado a erros na localização do agente em diferentes pontos do campo. É possível, porém, que a presença desses erros, e sua natureza estocástica, retardaram significativamente a convergência do CMA-ES. Por exemplo, por conta desses erros, uma mesma variação do movimento poderia, em uma instância, obter um desempenho ótimo, e em outra, por conta de erros de localização, ter um desempenho insatisfatório, o que dificultaria ao CMA-ES no processo de encontrar os parâmetros para o melhor chute possível, ainda que tentasse reproduzir uma situação mais próxima daquilo que o agente se encontra durante uma partida de futebol.

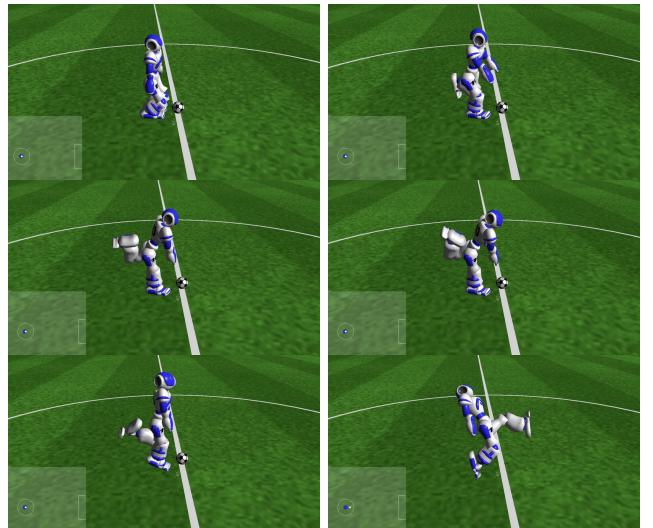


Figure 5: Movimento de chute otimizado.

## 5. Conclusão

Neste artigo pôde-se perceber a eficácia do algoritmo de otimização CMA-ES para resolver um problema livre de modelo com variáveis interdependentes.

Embora tenha sido passado um chute funcional como ponto de partida, ainda foram necessárias centenas de gerações para atingir o objetivo pela primeira vez. Isso demonstra a importância de ter um bom chute inicial para os problemas de otimização, especialmente para problemas de alta complexidade.

Por mais que o chute treinado seja bem específico, a adição recente do PasseModeOn fez com que este tipo de situação seja bem mais comum e tenha um grande potencial de conversão em gol.

O *keyframe* final tem uma taxa de sucesso entre 1/6 e 2/6. A taxa de sucesso depende também de fatores que não são o *keyframe*, alguns deles estocásticos. Considerando que este é um chute de alta dificuldade com muitas variáveis aleatórias envolvidas, pode-se considerar o resultado da otimização um sucesso.

## References

- [1] Nikolaus Hansen *The CMA Evolution Strategy: A Tutorial*. Cedex, France: University Paris-Sud, 2006
- [2] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.