# Bahia3D 2010 - Um Time para Liga de Simulação 3D da RoboCup

Adailton Cerqueira Jr., Adriano Veiga, Marco Simões, Diego Frias e Josemar Souza
Núcleo de Arquitetura de Computadores e Sistemas Operacionais
Universidade do Estado da Bahia (UNEB), Salvador, Bahia, Brasil
Emails: {adailton.junior, profpardal.88}@gmail.com, msimoes@uneb.br, diegofriass@gmail.com, josemar@uneb.br

***Resumo*—Um dos maiores desafios no desenvolvimento de um time de futebol para liga de simulação 3D, para um time inciante, é a criação de movimentos. Este artigo, não somente introduz a nova arquitetura do time Bahia3D, como também foca no nosso modelo de geração de movimentos. Adicionalmente, iremos descrever o trabalho que ainda está em desenvolvimento.**

## I. Introdução

O Bahia3D é desenvolvido pelo grupo Bahia Robotics Team (BRT), que faz parte do Núcleo de Arquitetura de Computadores e Sistemas Operacionais (ACSO), com a finalidade de investigar a aplicação de métodos da inteligência artificial em robôs autônomos. O BRT vem participando da sub-liga 2d e Mixed Reality desde 2007, onde vem tendo um bom desempenho e conquistou o 3º lugar na Mixed Reality na Robocup 2009. A equipe participa da sub-liga 3d desde 2009, onde ficou em 16º na classificação geral da liga.

No primeiro ano de participação do Bahia3D na Robocup resolvemos utilizar um time de base e melhorar suas habilidades. Utilizamos como time base o Little Green Bats [1]. Adotamos esta abordagem, pois o grupo BRT ainda não possuía experiência para os desafios básicos de movimentação da sub-liga 3D. Entretanto, no mesmo ano, por conta da mudança do sistema de visão do servidor da liga, a equipe decidiu iniciar um novo time desde a base, já que a base do Little Green Bats não contemplava o novo modelo de visão.

Na seção II é explicado a nova arquitetura dos agentes. Na seção III será descrita as duas abordagens principais utilizadas para a criação dos movimentos: a abordagem baseada no ambiente do Microsoft Robotics Studio [2], que será explicado na seção III-A e a abordagem de retroalimentação, explicado na seção III-B. A Seção IV descreve o modelo matemático para a criação dos movimentos. Logo após, na seção V, apontamos nossas maiores dificuldades encontradas durante o desenvolvimento do agente. Por fim, na seção VI mostramos os resultados obtidos e os nossos trabalhos futuros.

## II. Arquitetura

A decisão de abandonar o time base do Little Green Bats, além do fator de não contemplar o novo modelo de visão do servidor, se deu por falta de compreensão da arquitetura do time base. Por este motivo, uma nova arquitetura modular foi definida. Esta nova arquitetura prevê um baixo acoplamento entre os módulos. A figura 1 ilustra esta nova arquitetura. O agente se divide em quatro módulos. O módulo Sensor, o módulo Atuador (Actuator), o módulo Agente (Agent) e o módulo Cérebro (Brain). Cada um destes módulos tem uma funcionalidade especifica para o funcionamento do agente.

O módulo Sensor é responsável pela captura da mensagem do servidor e pela conversão da mensagem em dados compreensíveis para o agente. Ao receber a mensagem do servidor, o Sensor extrai as informações úteis e as armazena na base de dados do agente.

O módulo Atuador é responsável pela construção e envio da mensagem para o servidor. O agente informa o comando, com seu respectivo valor, e o Atuador, no fim do ciclo, monta a mensagem que será enviada para o servidor.

O módulo Agente trabalha como uma espécie de módulo intermediário de controle. Recebendo as ações abstratas do Brain e transformando essa ação em um comando especifico. Por exemplo, o Brain envia a ação “chutar”e o Agent converte essa ação para um comando especifico, informando quais juntas serão envolvidas nesta ação e qual a velocidade para que isto ocorra.

O módulo Cérebro é onde se encontra todo o raciocínio do agente. Internamente, este módulo é dividido em três camadas inter-relacionadas, como pode ser visto na figura 2.

O Nível Reativo (Reactive) é o nível mais básico do Cérebro, sendo responsável pela execução das ações mais simples e com pouco raciocínio, além de executar as ações resultantes das decisões dos níveis superiores. O Nível Planejador (Planner) é o responsável pela análise das informações recebidas do servidor e pela tomada de decisão com base nessa análise. Por ultimo, temos o Nível Cognitivo (Cognitive) que planeja e gere as metas que o time deve seguir, com base na informações adquiridas durante a partida.

Vale ressaltar que as duas ultimas camadas (Planejador e Cognitivo) ainda encontram-se por desenvolver e que com sua implementação esperamos prover a estrutura para que o time possa trabalhar com uma inteligência coletiva.

## III. Movimento

Um dos maiores desafios na elaboração do agente simulado em três dimensões é a criação de movimentos complexos a partir do movimento de cada junta do robô. Para vencer este desafio, criamos um modelo de planejamento de movimentos baseado na abordagem do Microsoft Robotics Studio. O movimento é controlado através de um método de feedback, que procura manter o movimento do robô dentro do planejado.

### *A. Abordagem Baseada no Microsoft Robotics Studio*

O ambiente de simulação Microsoft Robotics Studio (MSRS) também possui um modelo simulado do robô Nao.

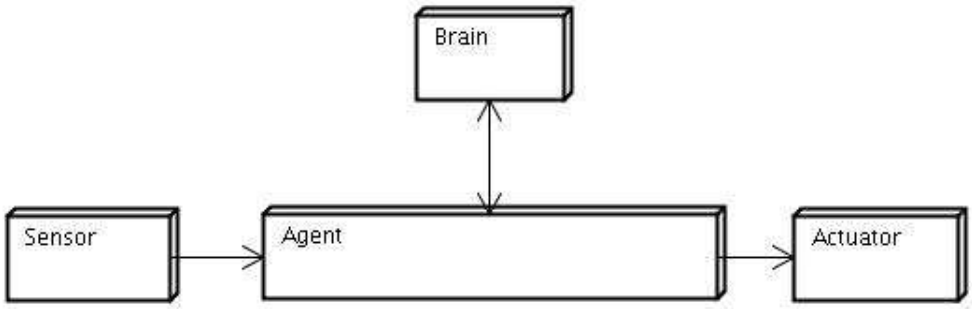


Figura 1. Arquitetura do Agente

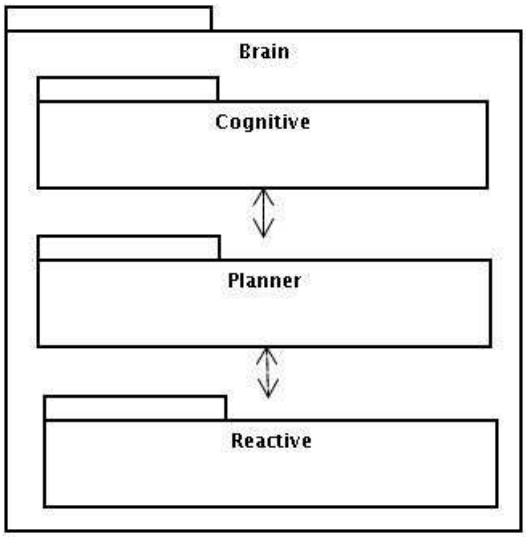


Figura 2. Divisão em Camadas do Cérebro

Em sua disponibilização gratuita do simulador, existem alguns movimentos pré-definidos e um código fonte base, para o auxilio ao rápido desenvolvimento de novas equipes.

Alguns meses antes do início da participação do Bahia3D na sub-liga de simulação 3D da Robocup, a equipe havia experimentado o ambiente do MSRS e adquiriu alguma experiência. Então, para resolver o problema da elaboração dos movimentos básicos do robô na simulação do Simspark, a equipe optou por adotar uma abordagem baseada na estrutura dos movimentos pré-definidos do MSRS.

A estrutura do movimento no MSRS consiste em uma sequência de poses fixas, separadas por um certo intervalo de tempo entre elas. Cada pose indica o ângulo de cada junta relativo ao seu próprio eixo. Entretanto, as simulações do Nao no MSRS e no Simspark diferem em algumas características, como por exemplo, os ângulos máximos e mínimos, o eixo ou a orientação das juntas.

Então, criamos um modelo de planejamento de movimentos baseados na abordagem da sequencia de poses do MSRS e adaptamos alguns movimentos básicos para a simulação do Simspark. Mas ainda não satisfeitos com o resultado, resolvemos melhorar os movimentos, suavizando a transição entre as poses.

### *B. Abordagem de FeedBack*

Assim como o ambiente real, o ambiente simulado também possui ruídos e nem sempre uma ação será executada de forma precisa. Então, para garantir que o planejamento do movimento descrito anteriormente seja executado de forma satisfatória, utilizamos um método de feedback.

Através deste método, é feito um controle do movimento, que vai readaptando dinamicamente o planejamento do movimento de acordo com a situação atual do robô. Esse método recebe como entrada os ângulos e velocidades desejadas (ou seja, os valores contidos no planejamento) e os ângulos atuais (os valores recebidos pelo servidor) de cada uma das juntas. Assim, a saída do método é a velocidade com a qual cada junta deve se mover para que o movimento completo do robô seja o mais próximo possível do planejado.

## IV. Modelo Matemático

A velocidade do movimento do robô quando ele se move é o resultado da variação contínua das velocidades angulares das juntas. A velocidade angular de cada junta deve ser especificada em intervalos de tempo equidistantes $t_i = t_{i-1} + \Delta t$, $i = 1, 2, ..., n$. Aqui $\tau = n\Delta t$ é a duração do movimento. A cada intervalo de tempo, a posição ($\alpha$) de cada junta $j$ muda em $w^j \Delta t$, que é $\alpha_i = \alpha_{i-1} + w_i^j \Delta t$, onde $w_i^j = w^j(t_i)$ é a velocidade angular da junta no intervalo de tempo $t_i$. Como condição inicial, nós consideramos que no tempo $t_0$ a junta $j$ estava em uma posição conhecida $\alpha_0^j$ e tinha a velocidade conhecida $w_0^t$.

### *A. Restrições físicas*

A posição, a velocidade e a aceleração das juntas devem ser limitadas.

*1) Posição:* Para cada junta $j$ nós temos $\alpha^j \in [\alpha_{min}^j, \alpha_{max}^j]$. Em um domínio discreto, a posição da junta no intervalo de tempo $i$ deve satisfazer $\alpha_{min}^j, \leqslant \alpha_{i-1}^j + w_i^j \Delta t \leqslant \alpha_{max}^j$. Isto é, assumindo que a posição no intervalo de tempo $i-1$ satisfaz a restrição, nós temos que

$$\frac{\alpha_{min}^j - \alpha_{i-1}^j}{\Delta t} \leqslant w_i^j \leqslant \frac{\alpha_{max}^j - \alpha_{i-1}^j}{\Delta t} \qquad (1)$$

*2) Velocidade:* Será assumido que a velocidade $w^j = \frac{d\alpha^j}{dt}$ varia no mesmo intervalo para todas as juntas, que é $w^j \in [w_{min}, w_{max}]$, $\forall j$, onde $w_{min} \leq 0$ e $w_{max} \geq 0$, nunca ambos sendo nulos. Em um domínio discreto, a velocidade instantânea no intervalo de tempo $i$ deve satisfazer a restrição

$$w_{min} \leqslant w_i^j \leqslant w_{max} \tag{2}$$

As velocidades máxima e mínima podem ser estimadas a partir dos dados de engenharia reversa, procurando por valores de velocidade extrema em uma família de movimentos personalizados.

*3) Aceleração:* A aceleração das juntas $\lambda^j = \frac{dw^j}{dt}$ também devem ser restritas, a fim de obter movimentos suaves e naturais. Nós vamos assumir o mesmo limite de aceleração para todas as juntas, isto é, $\lambda^j \in [\lambda_{min}, \lambda_{max}], \forall j$. Em um domínio discreto, a aceleração instantânea no intervalo de tempo $i$ é dada por $\lambda_i^j = \frac{w_i^j - w_{i-1}^j}{\Delta t}$, $\forall i$. Então, assumindo que a velocidade no intervalo de tempo $i-1$ satisfaz a restrição, nós temos que

$$w_{i-1}^j + \Delta t \lambda_{min} \leqslant w_i^j \leqslant w_{i-1}^j + \Delta t \lambda_{max} \tag{3}$$

As acelerações máxima e mínima podem ser estimadas a partir dos dados de engenharia reversa, calculando a segunda derivada do ângulo (ou a primeira derivada da velocidade) e procurando por valores extremos em uma família de movimentos personalizados.

*4) Combinando restrições:* Resumindo, nós temos

$$\mathbb{W}_i^{min} \leqslant w_i^j \leqslant \mathbb{W}_i^{max}, \forall i \tag{4}$$

onde

$$\mathbb{W}_i^{min} = max(w_{min}, \frac{\alpha_{min}^j - \alpha_{i-1}^j}{\Delta t}, w_{i-1}^j + \Delta t \lambda_{min}) \tag{5}$$

e

$$\mathbb{W}_i^{max} = min(w_{max}, \frac{\alpha_{max}^j - \alpha_{i-1}^j}{\Delta t}, w_{i-1}^j + \Delta t \lambda_{max}) \tag{6}$$

### B. Dinâmica

Considere um movimento elementar $k$ (por exemplo "andar para frente") iniciando no tempo $t_0$. Assuma a função de velocidade angular correspondente para a junta $j$ dada por $\omega_i^{k,j\star}$, $i = 1, 2, ..., n^k$ para um intervalo de tempo de referência $\Delta t^\star$, com condição inicial $\alpha^{k,j}(t_0) = \alpha_0^{k,j}$ e $\omega^{k,j}(t_0) = \omega_0^{k,j}$.

Considerando que todo o movimento é dado pela integral de $\omega(t)$, nós temos

$$\Omega^{k,j\star} = 0.5\Delta t^\star \sum_{i=1}^{n^k} (\omega_i^{k,j\star} + \omega_{i-1}^{k,j\star}) = 0.5\Delta t \sum_{i=1}^{n^k} (\omega_i^{k,j} + \omega_{i-1}^{k,j}) \tag{7}$$

$$\Omega^{k,j\star} = \Delta t \left( \frac{\omega_0^{k,j} + \omega_{n^k}^{k,j}}{2} + \sum_{i=1}^{n^k - 1} \omega_i^{k,j} \right) \tag{8}$$

Deixe $\tau^{k\star} = \eta^k \Delta t^\star$ ser a duração de referência do movimento $k$. Então, alterando $\Delta t \lessgtr \Delta t^\star$ nós podemos mudar a velocidade do movimento, mas satisfazendo a restrição 7. Pode-se notar que isso é alcançado sempre que $\Delta t \omega_i^{k,j} \equiv \Delta t^\star \omega_i^{k,j\star}$, $\forall i$, que é configurado

$$\omega_i^{k,j} \equiv (\Delta t^\star / \Delta t) \omega_i^{k,j\star}, \forall i \tag{9}$$

sujeito às restrições 4.

### C. Sincronização

A maioria dos sistemas de controle tem um período de atualização que define o menor intervalo de tempo $\Delta t_{min}$. Por simplicidade e robustez, é desejável que os intervalos de tempo $\Delta t$ sejam múltiplos de tal intervalo mínimo, isto é, $\Delta t = m \Delta t_{min}$, for $m \geqslant 1$. A máxima velocidade alcançável é obtida por $m = 1$.

## V. Dificuldades Encontradas

Nossa atual dificuldade no desenvolvimento do agente encontra-se no recebimento da mensagem HEAR, onde consta as mensagens ditas pelos agentes através do SAY. O problema é quando estas mensagens possuem caracteres que não encontram-se na tabela ASCII. Este é um problema de robustez, poís, teoricamente, as mensagens só deveriam possuir os caracteres da tabela ASCII da faixa de [0x20, 0x7E], exceto espaços em branco e parênteses.

## VI. Resultados e Trabalhos Futuros

Obtivemos bons resultados com o nosso modelo de movimentos, já que nosso agente agora é capaz de realizar seus movimentos mais básicos, como andar, levantar e chutar, de forma suave e com uma certa estabilidade. Nosso próximo desafio em relação aos movimentos, é torná-los mais flexíveis. Por exemplo, tornar o movimento de andar tão flexível que possamos escolher o tamanho do passo e a velocidade do movimento durante o jogo, para se adaptar às mais diversas situações.

Com os movimentos básicos implementados também podemos passar para a implementação das decisões de alto nível, já pensando na estratégia de jogo. E devido à arquitetura dos nossos agentes, incrementar aos poucos os módulos de raciocínio de alto nível se dará de forma natural, já que a mesma foi projetada para isso.


## Agradecimentos

Este projeto é apoiado pela UNEB, PICIN/UNEB, PROFORTE/UNEB, IC/FAPESB, PIBIC/CNPq e Fácil Computadores.


## Referências


[1] S. van Dijk, M. Klomp, B. Neijt, M. Platje, and M. van de Sanden, "Little green bats humanoid 3d simulation team description paper," in *Robocup Proceedings*. Robocup, 2008.

[2] Microsoft, "Microsoft robotics studio," http://www.microsoft.com/Robotics/.