Estudo e Desenvolvimento de um Time Base para o Robocup Soccer Simulator 2D

Matheus de Andrade Flausino[1], **Anielly Palhão**[1]

[1]Curso de Ciência da Computação – Universidade Federal de Alfenas (UNIFAL-MG)
Campus Santa Clara – Alfenas – MG – Brazil

{a12032,a12041}@bcc.unifal-mg.edu.br

Abstract. *This paper describes a team for the Robocup Soccer Simulator 2D which implements basic actions necessary to perform a soccer match in the proposed environment, seeking to integrate the perceptions, actions and behavior of each player of the team in order to achieve a goal maximum: the victory of the team in the game.*

Resumo. *Este artigo descreve uma equipe para o Robocup Soccer Simulator 2D que implementa ações básicas necessárias para realizar uma partida de futebol no ambiente proposto, buscando integrar as percepções, as ações e o comportamento de cada jogador do time a fim de alcançar um objetivo máximo: a vitória da equipe na partida.*

1. Introdução

O Robocup Soccer Simulator 2D é uma simulação de futebol virtual bidimensional com duas equipes de 11 agentes representado por um servidor central, chamado SoccerServer. Um servidor o qual fornece informações sobre o ambiente e o atualiza conforme os eventos recebidos. Cada atualização é denominada CICLO, cada segundo equivale a 10 ciclos. A partida é definida durante 6000 ciclos.

O presente trabalho tem como objetivo um time de base para o Robocup 2D, faz-se necessário a definição de uma estratégia para definir a atuação do agente no ambiente, de modo a alterá-lo ao seu favor (fazer gols e vencer a partida). Logo cada time deve ter uma formação correta no jogo: goleiros, atacantes, zagueiros entre outros.

Uma partida de futebol apresenta determinadas características, descritas na subseção 1.1, que tornam a sua modelagem em inteligência artificial um tanto quanto atrativa para o desenvolvimento de novas técnicas e novos estudos na área. "O Robocup torna-se um forte recurso para pesquisas na área de Inteligência Artificial, uma vez que ele fornece um problema padrão onde uma vasta gama de algoritmos e tecnologias podem ser integradas e/ou implementadas"[Kitano et al. 1997].

1.1. Características do ambiente fornecido pelo Robocup Soccer Simulator 2D

Parcialmente observável: cada agente possui um ângulo de visão que pode variar durante o jogo, porém é impossível ter um visão completa do ambiente, tornando-o um ambiente parcialmente observável.

Estocástico: o estado atual do ambiente somado a ação escolhida pelo agente não define por completo o próximo estado do ambiente. Esta afirmação é denotada, especialmente, pela presença de agentes adversários e companheiros que também tomam decisões que

alteram o ambiente.
Multi-agente: uma partida de futebol no Robocup Soccer Simulator 2D, assim como na vida real, conta com 11 jogadores de cada time, e todos eles são agentes autômatos independentes que interagem entre si, buscando seus objetivos.
Dinâmico: enquanto o agente pensa/atua, o ambiente pode sofrer mudanças.
Discreto: cada agente possui uma percepção do ambiente e pode realizar apenas uma ação a cada ciclo. Sendo uma partida constituída de 6000 ciclos, cada agente possui 6000 ações para serem realizadas durante o jogo.[Kitano et al. 1997]

2. Extraindo a coordenada cartesiana para localização do agente e a direção para onde o agente está olhando

No ambiente de simulação Robocup Soccer 2D (moldes competitivos), os agentes não possuem nenhum tipo de informação sobre seu posicionamento global e nem sobre a direção a qual estão olhando, apenas possuem informações sobre os objetos que estão no seu campo de visão. Para orientar a sua localização, existem as flags distribuídas no campo, como pode ser visto na Figura 1. Contudo, informações sobre a localização e direção do agente são de extrema necessidade para o bom desempenho do mesmo e consequentemente do time. Com base nessa necessidade, desenvolvemos um algoritmo que utiliza as flags para inferir uma coordenada cartesiana que representa a posição do agente e um ângulo que representa a direção global que ele está olhando. Inicialmente, o algoritmo extrai a direção para onde o agente está olhando, os passos desta parte do algoritmo são descritos a seguir:

1. Percorrer todas as flags externas do campo, na seguinte direção: Flags Externas Superiores, Flags Externas da lateral direita, Flags Externas Inferiores e Flags Externas da lateral esquerda. Assim que encontrar duas flags no campo de visão do agente, este laço deve ser interrompido e ir para o próximo passo, caso nenhuma flag esteja no campo de visão do agente o algoritmo de extração das coordenadas cartesianas é abortado.
2. Após encontrar duas flags no campo, são criados pontos bidimensionais relativos para cada uma das duas flags, utilizando as informações das variáveis Distance e Direction, ambas variáveis fornecidas pelo servidor.
3. Neste momento é possível calcular o ângulo relativo formado entre as duas flags capturadas no passo 1, usando a equação da tangente $(\tan\theta = \frac{\Delta y}{\Delta x})$ nos pontos bidimensionais calculados no passo 2.
4. Para finalizar a extração do ângulo que representa a direção, é necessário verificar a qual lado do campo as flags utilizadas para o cálculo pertencem. Isto é necessário pelo fato de que até então calculamos apenas o ângulo entre as duas flags, e agora precisamos adicionar um valor ao ângulo calculado a fim de torná-lo o ângulo global. Se as flags pertencerem ao lado esquerdo do campo será adicionado 90°ao ângulo, senão, se as flags pertencerem ao lado direito do campo será adicionado 270°, ou então, se o agente estiver olhando mais para a esquerda então será adicionado 180°.

Em seguida, o algoritmo começa a calcular os pontos bidimensionais da posição global do agente no campo. E para isso ele segue os seguintes passos:

1. Pegar uma das flags usadas para calcular a direção que o agente está olhando e criar um novo ponto bidimensional para representá-la. Esse ponto é calculado usando as variáveis Distance e Direction, como no passo 2 do algoritmo anterior , porém agora deve-se somar à variável Direction, o valor encontrado para o ângulo global que o agente está olhando, valor este que fora calculado anteriormente.
2. Para conseguir as coordenadas do ponto, que representará a posição do agente no campo, basta apenas fazer a subtração entre o ponto real da flag utilizada no passo 1 deste algoritmo e o ponto relativo calculado no passo anterior. O resultado será o ponto onde o agente está localizado.

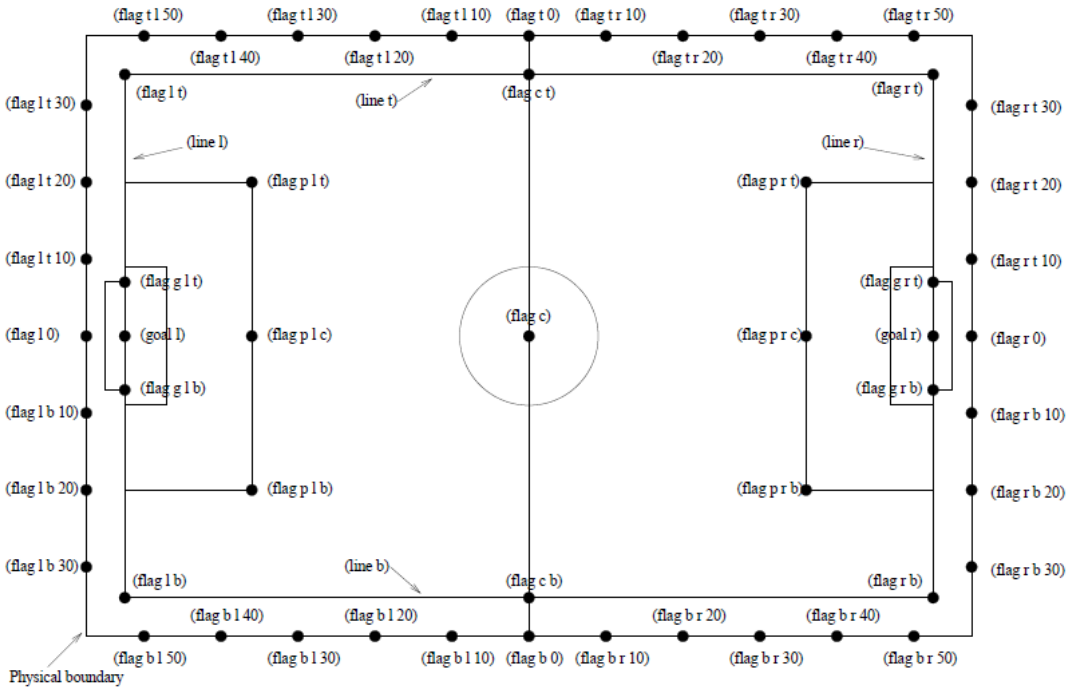

Figura 1. Flags dispostas pelo campo. Fonte:[Chen et al. 2002]

3. Inferindo a velocidade dos objetos móveis

Calculado o ângulo que representa a direção global onde o agente está olhando, é possível agora, então, de maneira simples, inferir a sua velocidade, que é representado por um vetor de 2 coordenadas, utilizando as variáveis Amount of Speed e DirectionOfSpeed, enviadas para o agente pelo servidor do Robocup Soccer Simulator 2D.
Em seguida, com o vetor de velocidade do agente calculado, podemos extrair o vetor de velocidade de todos os objetos móveis que estão no campo de visão do agente. Para tal, precisamos resolver um sistema linear de 2 equações e 2 variáveis, utilizando informações dispostas no manual do Robocup Soccer Simulator 2D. As informações necessárias são as seguintes:

$$
\begin{aligned}
p_{rx} &= p_{xt} - p_{xo} \\
p_{ry} &= p_{yt} - p_{yo} \\
v_{rx} &= v_{xt} - v_{xo} \\
v_{ry} &= v_{yt} - v_{yo} \\
Distance &= \sqrt{p_{rx}^2 + p_{ry}^2} \\
Direction &= \arctan\left(p_{ry}/p_{rx}\right) - a_o \\
e_{rx} &= p_{rx}/Distance \\
e_{ry} &= p_{ry}/Distance \\
DistChng &= (v_{rx} * e_{rx}) + (v_{ry} * e_{ry}) \\
DirChng &= [(-(v_{rx} * e_{ry}) + (v_{ry} * e_{rx}))/Distance] * (180/\pi) \\
BodyDir &= PlayerBodyDir - AgentBodyDir - AgentHeadDir \\
HeadDir &= PlayerHeadDir - AgentBodyDir - AgentHeadDir
\end{aligned}
$$

Figura 2. Variáveis do See Info. Fonte:[Chen et al. 2002]

A partir destas informações é possível definir as equações que formarão o sistema linear. Efetuado todos os cálculos necessários, as equações finais ficam da forma:

$$
vely = \frac{(\frac{e_{rx} \times Distance \times DirChng \times \pi}{180}) + (e_{rx}^2 \times v_{yo}) + (e_{ry}^2 \times v_{yo}) + (e_{ry} \times DistChng)}{e_{ry}^2 + e_{rx}^2}
$$

$$
velx = \frac{DistChng + (e_{rx} \times v_{xo} + e_{ry} \times v_{yo}) - e_{ry} \times vely}{e_{rx}}
$$

Onde *velx* e *vely* tornam-se as componentes do vetor velocidade do objeto em questão e *vxo* e *vyo* são as componentes do vetor velocidade do agente, calculado anteriormente no início desta seção. As demais variáveis são obtidas pelo servidor do Robocup Soccer Simulator 2D e são detalhadas no manual do mesmo.

4. Tomada de Decisões

O time possui estratégias diferentes para o goleiro, para os atacantes e para os demais jogadores. Contudo, os agentes possuem um nível de inteligência baixo, sendo apenas reativos aos estímulos gerados pelo ambiente, seguindo uma estrutura de decisão conhecida como árvore de decisões. A estratégia para o goleiro é bastante trivial, consiste apenas em ficar entre a linha da bola e a flag do gol o qual ele defende e capturar a bola sempre que possível. Esta estratégia é ilustrada no fluxograma a seguir:

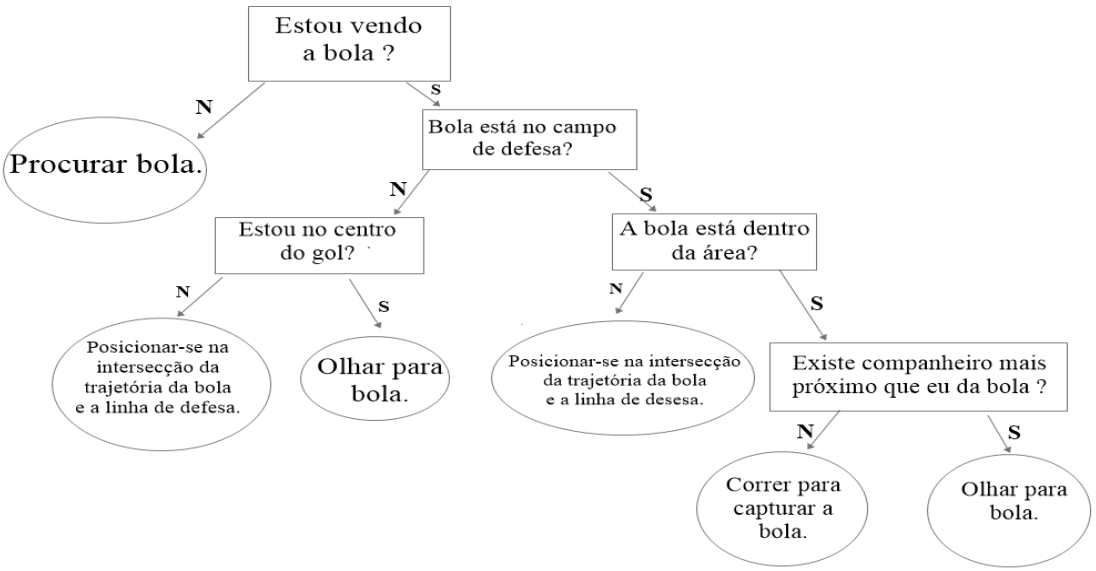

Figura 3. Fluxograma que ilustra a tomada de decisões do goleiro

A estratégia para os atacantes apenas difere da estratégia dos demais jogadores quanto a prioridade em realizar o passe, observado que, o atacante se preocupa menos em efetuar um passe de bola quando comparado aos demais jogadores. O fluxograma que ilustra a estratégia dos jogadores de linha, pode ser visto a seguir:

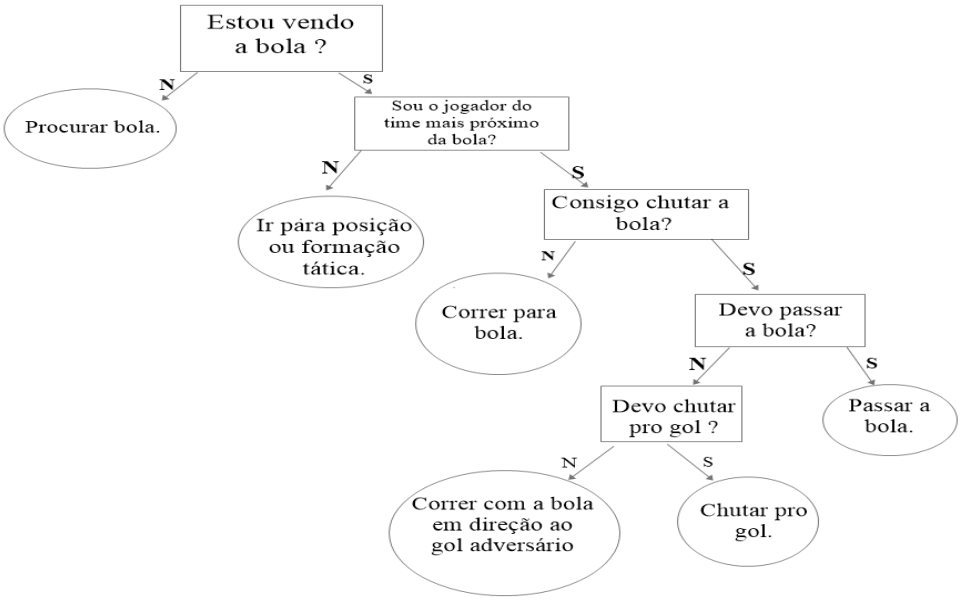

Figura 4. Fluxograma que ilustra a tomada de decisões dos jogadores

5. Posicionamento no Campo de Jogo

Entre os muitos fatores que interferem no resultado final de uma partida de futebol, alguns aspectos estão relacionados com os sistemas de jogos, também conhecido como esquemas táticos, que são adotados pelas equipes durante a partida [Favarin et al. 2007].
O time base desenvolvido implementa funcionalidades que permitem a criação de formações táticas e atribuições de função para cada agente (lateral, zagueiro, atacante, entre outros). Além disso, também possui um algoritmo para decidir sobre o ponto bidimensional que o agente deve ocupar sendo este o ponto mais adequado para a situação atual da partida, tendo em consideração a função do agente no campo e a formação tática vigente. Este ponto é calculado da seguinte maneira: Toma-se p como sendo o ponto original do agente na formação tática e então adiciona-se os valores de *pBola* que representa o ponto atual da posição da bola, multiplicado por *ancoraX* e *ancoraY* que representam o quanto o agente deve se aproximar da bola. O ponto resultante dessa operação se torna o ponto de posicionamento atual do agente, porém esse ponto deve respeitar a extensão do campo e os limites impostos pela formação tática, como por exemplo, um defensor não deve se posicionar muito próximo do campo de ataque e nem um atacante deve voltar muito para o campo de defesa, etc.

Figura 5. Time no seu posicionamento utilizando estratégia 4-3-3

6. O time base desenvolvido

Desenvolvido na linguagem de programação JAVA e utilizando a versão do simulador Robocup Soccer 2D para Windows, o time base desenvolvido implementa as funcionalidades básicas necessárias para a execução de uma partida, tornando-o um framework para o ambiente. Dentre as principais funcionalidades contidas no cliente desenvolvido, podemos enumerar as seguintes:

- **Conectividade cliente-servidor:** Esta é a parte medular do projeto. O cliente possui uma conexão bem estabelecida com o servidor tanto para enviar mensagens,

quanto para recebê-las, e portanto, a comunicação ocorre de maneira satisfatória. Para "escutar" as mensagens do servidor (mensagens do tipo "see", "sense body", "hear", "init") , já existem métodos que recortam e armazenam essas informações.

- **Sistema de inferência de posição cartesiana, direção e velocidade dos objetos:** Como já apresentado em seções anteriores, o cliente possui métodos que inferem a posição e direção do agente e também métodos que inferem o vetor velocidade dos objetos móveis presentes no campo de visão do agente.
- **Predições sobre posições futuras dos objetos:** Foram desenvolvidos métodos capazes de tentar antever a posição global dos objetos móveis em ciclos futuros, isso é possível pelo fato de possuirmos as informações da velocidade e posição dos objetos. Salientando que, com isso, o cliente também possui métodos para determinar pontos de intersecção entre a trajetória do movimento do agente e a trajetória dos objetos.
- **Ações do agente:** Diversos métodos que performam ações do agente, tais como (chute para um ponto alvo, virar o corpo para um ponto, correr para um ponto, virar o pescoço, etc), foram desenvolvidos com o objetivo de simplificar posteriores desenvolvimentos/mudanças.
- **Percepções do agente:** O cliente possui uma certa modelagem do ambiente, que permite ao agente tomar conhecimento sobre as coisas à sua volta, como por exemplo, se ele está sendo marcado, se possui um companheiro próximo para efetuar um passe de bola, se está fora de posição dentro da formação tática adotada, se a bola está no campo de ataque ou não, etc.

7. Resultados

Como todo o esforço do nosso trabalho esteve em desenvolver um código base para o Robocup que forneça informações indispensáveis para uma execução consistente e satisfatória de uma partida simulada de futebol, e implementar recursos básicos para as possíveis ações que um agente pode executar, optamos então por analisar a qualidade das nossas soluções propostas ao invés de simplesmente executar uma partida de futebol contra algum outro time com performance já reconhecida no cenário.

Durante a análise dos resultados, percebemos que a aptidão das nossas soluções poderia ser melhorada caso os dados de entrada dos algoritmos desenvolvidos fossem previamente tratados. Esse tratamento dos dados ficará melhor exposto no fim desta seção, onde a interpretação final dos resultados já terá sido descrita.

De antemão, vale ressaltar que os valores reais sobre a localização, direção e velocidade de qualquer objeto móvel no campo (jogadores e a bola) em um determinado ciclo, foram obtidos através do logplayer do Robocup, que é uma ferramenta capaz de executar os arquivos de log gerados pelo servidor do Robocup após uma partida, permitindo que a mesma possa ser reprisada posteriormente com as informações reais disponíveis na tela. Este recurso é mais adequadamente descrito no manual do Robocup.[Chen et al. 2002].

Inicialmente, analisamos a eficiência do algoritmo que infere o ponto de localização do agente no campo. Para esta análise utilizamos 100 tentativas aleatórias que o algoritmo inferiu sobre o posicionamento do agente em determinados ciclos da partida equacionados com o posicionamento real do agente nestes mesmos ciclos.

A equação utilizada para verificar a diferença entre o ponto que foi inferido pelo algoritmo, e o ponto real, é a fórmula de distância entre pontos:

$\sqrt{(x_e - x_r)^2 + (y_e - y_r)^2}$, onde (x_e, y_e) é o ponto estimado pelo algoritmo, e (x_r, y_r) é o ponto real. O gráfico da Figura 6 ilustra essa diferença entre o posicionamento real e o posicionamento estimado pelo algoritmo.

Em seguida, analisamos a eficiência do algoritmo que infere a localização de um determinado objeto visível por um agente, calculando a diferença entre o que o algoritmo estimou para como sendo a localização deste objeto no campo e sua localização real. A metodologia abordada aqui é a mesma utilizada para calcular a diferença do posicionamento do agente, utilizamos a mesma equação para determinar a diferença e também a mesma quantidade de dados aleatórios. O gráfico da Figura 7 expõe as diferenças obtidas nesta amostragem.

E o último algoritmo analisado foi o algoritmo de inferir velocidade dos objetos visíveis. Esta análise também adota as metodologias utilizadas anteriormente. O gráfico da Figura 8 explicita as diferenças obtidas nesta amostragem.

Interpretando os dados obtidos, é possível notar que os gráficos que representam as diferenças de posicionamento, tanto o gráfico do agente, quanto o gráfico dos objetos visíveis, sofrem intensas oscilações dentro de uma certa margem de erro. Consideramos aqui esta margem de erro como sendo uma margem aceitável, visto que o campo possui dimensões muito maiores do que as dimensões das diferenças obtidas, e portanto adotar as estimativas, não desloca muito o objeto do seu real posicionamento no campo.

Expressando com mais exatidão, o campo possui uma área de $(105 \times 68) = 7140$ unidades de medida, enquanto a área da circunferência do erro, considerando a pior média dos resultados mostrados nos gráficos do fim desta seção como sendo o raio da circunferência, é $(\pi \times 1.34046^2) = 5.64492$ unidades de medida, que representa $\approx 0.080\%$ da área do campo.

Contudo, esse comportamento ruidoso apresentado nos gráficos, era previsto. Sabendo que o servidor do Robocup possui um módulo de adição de ruído visual que tem como objetivo incorporar uma quantização nos valores de distância que o agente percebe dos objetos em seu campo de visão, impedindo que ele saiba a exata posição de um objeto a medida que o mesmo esteja mais distante, e como, teoricamente, as flags externas do campo são as mais distantes dos jogadores dentro do campo, sempre capturaremos alguma taxa de ruído visual que impedirá que saibamos a real distância entre estas flags e o agente, dificultando assim, um cálculo exato sobre o posicionamneto dos objetos e do agente. Portanto, mesmo que o agente em uma determinada transição de ciclos permaneça na mesma posição, porém gire sua cabeça em alguma angulação θ suficiente para colocar novas flags com distâncias diferentes das anteriores no seu campo de visão, o algoritmo calculará a sua localização com a propagação do novo ruído, gerando uma oscilação mesmo estando parado. Logo, fica evidente que, as performances dos algoritmos desenvolvidos neste trabalho seriam superiores caso houvesse um método de amortização da taxa de ruídos com base na função de quantização das distâncias dos objetos feitos pelo servidor Robocup.

Por fim, observamos que as soluções propostas neste trabalho fornecem resultados com um alto grau de confiança e com uma margem de erro satisfatória, possibilitando que os futuros desenvolvimentos possam concentrar seus esforços em outros problemas propostos pelo paradigma Robocup.

Figura 6. Gráfico da diferença entre o posicionamento real e o posicionamento estimado nas 100 tentativas. Média de 1.34046 e desvio padrão de 1.80397

.

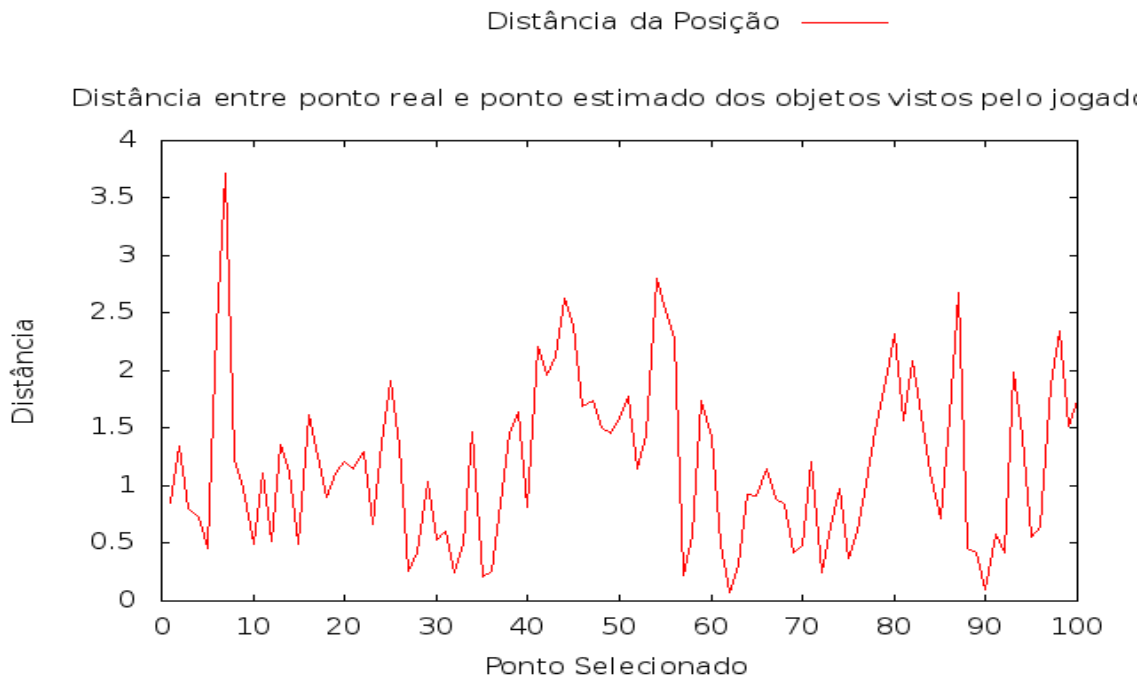

Figura 7. Gráfico da diferença entre o posicionamento real e o posicionamento estimado dos objetos nas 100 tentativas. Média de 1.19754 e desvio padrão de 0.71706

.

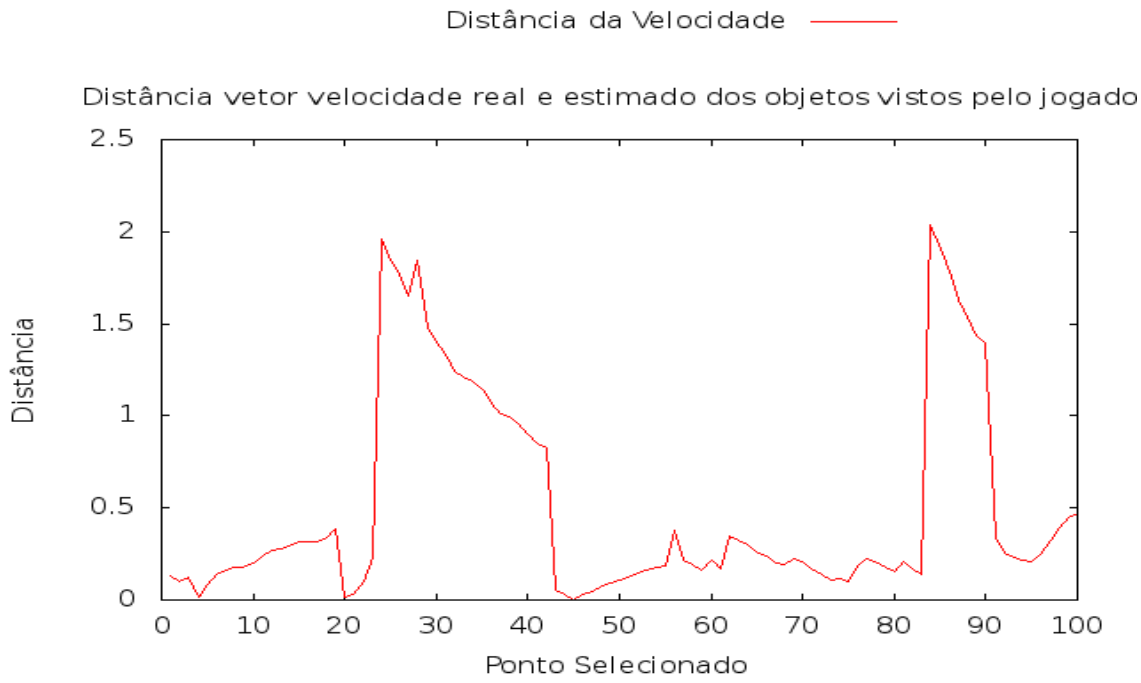

Figura 8. Gráfico da diferença entre o vetor real e o vetor estimado da velocidade dos objetos nas 100 tentativas. Média de 0.50940, com desvio padrão de 0.56950.

.

Referências

[Chen et al. 2002] Chen, M., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., and Yin, X. (2002). Manual the robocup soccer simulator.

[Favarin et al. 2007] Favarin, M. A., Moioli, A., Santos, T. C. D., and Fachim, R. D. (2007). Esquemas tÁticos das equipes na copa do mundo de futebol 2006:inovações ou adaptações? 5.

[Kitano et al. 1997] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., and Matsubara, H. (1997). Robocup: A challenge problem for ai. 18.