COMPUTING 4nd Edition Volume 1

Simulador computacional para auxiliar no desenvolvimento de estratégias de comportamento para futebol de robôs

Fabrício J. C. Montenegro[1], **Rodrigo da S. Guerra**[1]

[1]Universidade Federal de Santa Maria (UFSM)
97105-900 – Santa Maria – RS – Brasil

fabriciojcmontenegro@gmail.com, rodrigo.guerra@ufsm.br

Abstract. *The work here presented talks about the development of a computational simulator applied to robotics with focus on the development of high level behavior strategies for robot soccer. The development of behavior strategies for robot soccer can be greatly improved when developed and tested on a simulator. The available simulators, though, provide a low level of abstraction so they can maintain a hardware compatibility or a high level of abstraction that gives up on this compatibility. The simulator developed on this work provides a high level of abstraction without letting go of hardware compatibility. It allows the creation of behavior strategies that can communicate with the simulation as well as directly with a real robot. The simulator was developed and is utilized by a humanoid robot soccer team.*

Resumo. *O trabalho aqui apresentado trata sobre o desenvolvimento de um simulador computacional aplicado à robótica com foco no desenvolvimento de estratégias de comportamento de alto nível para futebol de robôs. O desenvolvimento de estratégias de comportamento para futebol de robôs pode ser grandemente facilitado quando desenvolvido e testado junto a um simulador. Os simuladores encontrados, entretanto, oferecem um baixo nível de abstração para manter uma compatibilidade com o hardware ou um alto nível de abstração que abre mão dessa compatibilidade. O simulador desenvolvido neste trabalho oferece um alto nível de abstração sem abrir mão da compatibilidade com o hardware. Ele permite a criação de estratégias de comportamento que podem comunicar-se tanto com o simulador quanto diretamente com um robô real. O simulador foi desenvolvido e é utilizado por uma equipe de futebol de robôs humanoides.*

1. Introdução

A RoboCup é um evento internacional anual que tem como objetivo fomentar o desenvolvimento e a pesquisa nas áreas de robótica e inteligência artificial [Kitano et al. 1998]. A liga de futebol de robôs humanoides oferece um ambiente dinâmico para pesquisadores e estudantes do mundo inteiro aprimorarem suas pesquisas. Consta dentre as regras da prova de futebol, que os robôs devem ser autônomos, i.e., não devem ser controlados remotamente.

O desenvolvimento de um robô humanoide agrega conhecimentos de diversas áreas da tecnologia, entre elas, engenharia elétrica, engenharia mecânica, mecatrônica e computação. Uma vez que o projeto envolve tantas áreas, a equipe de desenvolvimento

acaba subdividindo-se em equipes menores. Durante o desenvolvimento do projeto, muitas das subequipes precisam fazer uso constante do robô para atingir os resultados desejados.

O desenvolvimento da caminhada é um bom exemplo. É possível alterar parâmetros do algoritmo de caminhada e teorizar sobre qual será o resultado na prática, mas esse resultado só será conhecido quando for feito um teste em uma situação real. O teste é imprescindível para uma boa calibragem dos parâmetros visto que as condições externas ao robô afetam diretamente sua performance. Este teste, muitas vezes, vem com um alto custo pois as partes eletrônicas e motores dos robôs são frágeis e de alto valor financeiro. Quanto maior o volume de testes, maior o risco de que partes do robô venham a desenvolver algum defeito.

Além disso, enquanto a equipe responsável pelo desenvolvimento da caminhada utiliza o robô para testes, equipes responsáveis por outras partes do projeto não podem usá-lo, burocratizando o avanço do projeto.

Em face dessas condições fica claro que, ao evitar testes com o robô real, é possível diminuir o custo e aumentar a velocidade de desenvolvimento do projeto. Para isso, podemos utilizar um simulador computacional. Com esse objetivo, podemos encontrar trabalhos relacionados que serão apresentados mais adiante neste texto.

Os simuladores encontrados encaixam-se em duas categorias distintas. A primeira abrange simuladores de robótica com alta compatibilidade com o hardware, mas que não oferecem um alto nível de abstração. A segunda categoria diz respeito aos simuladores com um alto nível de abstração mas que não oferecem o compromisso de manter uma compatibilidade com o hardware.

O objetivo do sistema desenvolvido nesse trabalho é abordar os problemas citados anteriormente, criando um simulador computacional aplicado à robótica que ofereça ao programador um nível alto de abstração mas que mantenha uma compatibilidade com o hardware. Para isso foi criado um simulador 2D com uma interface simples para oferecer ao programador um alto nível de abstração. O simulador ainda possui uma API que descreve a comunicação que pode acontecer com um robô real, oferecendo a compatibilidade com o hardware.

A seguir será feita uma revisão bibliográfica que apresentará mais detalhes sobre a RoboCup, robótica e inteligência artificial, contextualizando o leitor sobre estes tópicos. O restante do texto está disposto em mais quatro seções. A seção de metodologia descreve a abordagem utilizada para solução dos problemas discutidos e apresenta detalhes sobre o desenvolvimento do projeto. A seção de discussão apresenta um debate sobre a abordagem utilizada em comparação com outras abordagens. A seção de trabalhos relacionados apresenta trabalhos de relevância para o projeto e faz uma comparação destes trabalhos com o trabalho aqui apresentado. E a seção de conclusão discute os pontos de sucesso da abordagem bem como problemas encontrados, além de propor melhorias para o sistema e trabalhos futuros.

2. Revisão Bibliográfica

O desenvolvimento do projeto aqui apresentado se dá no contexto de robótica e inteligência artificial, com foco na prova de futebol de robôs humanoides da RoboCup. Estes

tópicos são apresentados em mais detalhes a seguir.

2.1. *RoboCup*

Em 1997, um evento colocava frente à frente pela segunda vez Garry Kasparov, considerado o melhor jogador de xadrez de todos os tempos, e *Deep Blue*, um supercomputador desenvolvido pela *IBM* para jogar xadrez [Campbell et al. 2002]. A máquina venceu o confronto e, ao transpor esse desafio, a pesquisa em inteligência artificial procurava um novo objetivo que fosse desafiador o suficiente para expandir o estado da arte. Foi então que a *RoboCup* surgiu propondo o objetivo de que à metade do século XXI uma equipe de robôs humanoides seja capaz de vencer uma partida de futebol contra a equipe (de humanos) campeã mundial do mesmo ano seguindo as regras oficiais da FIFA [Kitano et al. 1998]. Atualmente este objetivo não parece tão próximo de ser alcançado, mas a cada ano a comissão organizadora altera parte das regras da competição afim de aumentar o desafio, fazendo com que as equipes tenham que transpor barreiras cada vez mais altas.

2.2. Robótica e Inteligência Artificial

Robótica e inteligência artificial são dois assuntos ligados intimamente. A teoria de *embodiment*, aqui traduzido livremente como *personificação* no sentido de possuir um corpo, sugere que agentes autônomos do mundo natural são em sua maioria seres biológicos e que sua inteligência está diretamente ligada a seus corpos e a forma com a qual eles interagem com o ambiente. Dessa forma, para criar agentes com inteligência artificial, precisamos dar um corpo artificial, ou robótico, a esses agentes. A personificação implica que o agente está continuamente sujeito a quaisquer influências do ambiente e podemos estudar, por exemplo, como um organismo adquire experiência: conhecimento sobre o ambiente obtido ao interagir com ele [Pfeifer et al. 2001].

Um agente é qualquer coisa que pode ser vista como aquela que percebe o ambiente através de sensores e age nesse ambiente através de atuadores. Um agente humano tem olhos, ouvidos e outros órgãos funcionando como sensores; e mãos, pernas, boca e outras partes do corpo como atuadores. Um agente robótico usa câmeras como sensores e motores variados como atuadores. Um *software* agente tem palavras codificadas em bits como suas percepções e suas ações [Russell et al. 1995].

Na Figura 1 estão descritos ambiente e agente. O agente age no ambiente através de seus atuadores e percebe o ambiente através de seus sensores.

Para que seja criado um simulador computacional com alto nível de abstração focado somente na definição do comportamento, é preciso simular o ambiente, os sensores e os atuadores, deixando o usuário responsável apenas pela criação da mente do agente. Nessa abordagem os sensores e atuadores fazem parte do ambiente.

3. Metodologia

Foi um requisito do sistema que ele possuísse uma interface gráfica 2D simulando o campo de futebol. Durante a maior parte do tempo a bola se encontra no chão e a altura dos objetos pode ser ignorada, colaborando com o alto nível de abstração desejado para o sistema. Outro requisito foi que o sistema fosse capaz de representar quatro tipos básicos de objetos: robôs, postes, bolas e um tipo de objeto não identificado. Por fim,

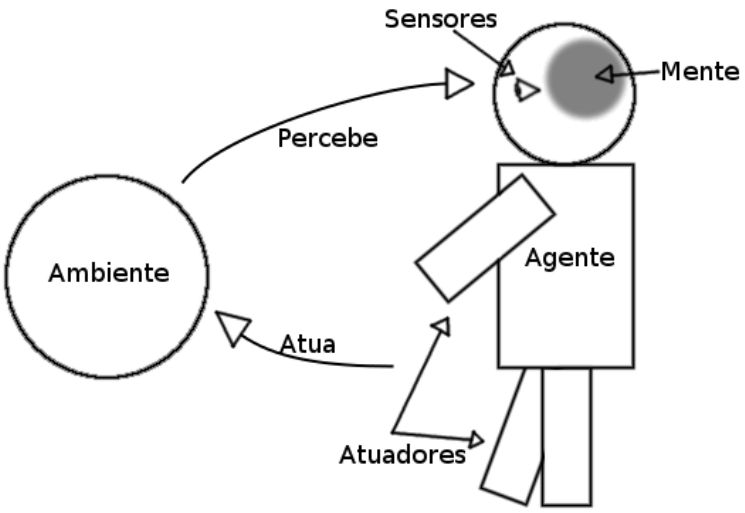

Figura 1. Um agente interage com o ambiente através de sensores e atuadores.

era necessário que um processo que representa uma mente fosse capaz de conectar-se à simulação via rede para controlar um robô simulado. Entretanto, várias mentes poderiam conectar-se ao simulador simultaneamente, dando a ele uma característica de servidor.

Seguindo os requisitos descritos acima, o sistema projetado funciona em duas partes: simulação, parte responsável por criar um ambiente onde o usuário possa criar um cenário de jogo com robôs, bolas e postes; e mente, parte responsável pela tomada de decisão do robô simulado em resposta ao ambiente. A simulação comunica à mente informações percebidas pelos sensores. A mente recebe essas informações, toma uma decisão e envia como resposta à simulação um comando para o robô simulado. A simulação recebe então o comando e executa através dos atuadores do robô simulado, atualizando a simulação para a próxima iteração. A Figura 2 descreve a arquitetura do sistema.

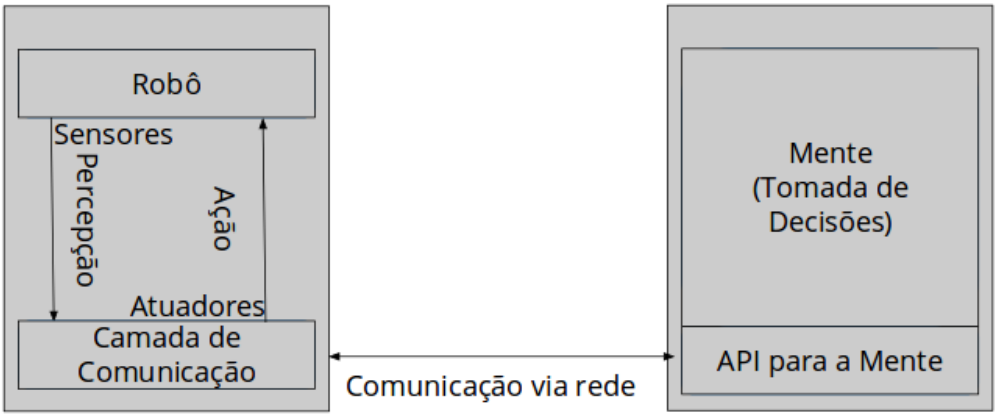

Figura 2. O sistema funciona em duas partes que comunicam-se. A parte a ser desenvolvida pelo criador do comportamento do robô é a mente, que se comunica com a simulação através de uma camada de abstração.

Para atingir o objetivo do sistema, é preciso saber como é o funcionamento de um

robô humanoide que joga futebol para que seja possível simular sensores e atuadores de maneira transparente.

Quanto aos sensores, foi levado em consideração apenas a câmera, que é o sensor essencial dentre os permitidos pelas regras da competição para que o robô perceba o ambiente. Para que a simulação gere uma abstração para o usuário, é necessário simular todo o processo feito desde a captura de uma imagem até a detecção de objetos nela. Para o programador do comportamento não é necessário saber como esses objetos são detectados no ambiente, ele apenas precisa receber uma saída que informa o tipo e a posição de um objeto percebido no ambiente.

Tratando-se dos atuadores, o robô é capaz de mover braços, pernas e pescoço, mas apenas alguns desses movimentos são realmente importantes para o desenvolvedor do comportamento. A posição em que a cabeça se encontra é uma informação valiosa pois alterando essa posição podemos buscar objetos no ambiente. O funcionamento dos braços são irrelevantes para o comportamento pois durante a prova eles influenciam apenas no equilíbrio e, caso o robô sofra uma queda, os braços o ajudam a erguer-se. A caminhada possui um complexo sistema, mantido separado do comportamento, que é responsável também por fazer o robô levantar-se, tornando o controle dos braços irrelevante para o comportamento. O controle sobre as pernas do robô ao qual o usuário tem acesso fica então limitado à direção e a velocidade da caminhada e a um controle de chute, uma vez que a principal interação entre o jogador e o ambiente em uma partida de futebol é o chute na bola. Conforme dito anteriormente, o sistema funciona em duas partes: simulação e mente. A seguir, as duas partes são explicadas em mais detalhes.

3.1. Simulação

A simulação é a parte do sistema que fornece ao usuário uma tela, demonstrada na Figura 3, onde ele pode criar um cenário para testar o comportamento desenvolvido para o robô. Na simulação, é possível criar os objetos básicos presentes em uma prova de futebol de robôs humanoides: robô, bola e poste; mas também um objeto não identificado, para um teste genérico.

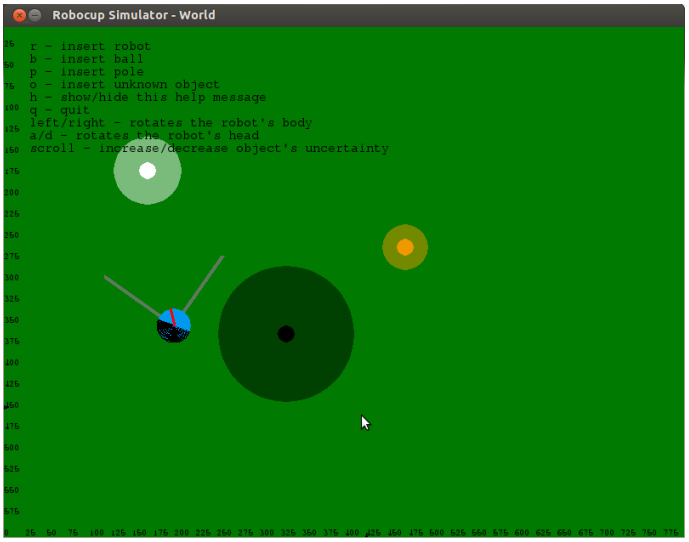

Figura 3. Demonstração da tela do programa.

O usuário pode inserir objetos no cenário através de teclas pré-definidas. Os objetos, cujas propriedades são exibidas na Figura 4, são adicionados em uma lista. Os objetos possuem uma *string* que descreve seu tipo e sua posição no ambiente. A posição é descrita em coordenadas retangulares considerando a origem do espaço 2D como o ponto do topo esquerdo da tela.

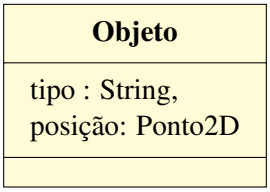

Figura 4. Um objeto possui um tipo e uma posição.

Uma vez criada a cena, o sistema simula a percepção do robô, analisando seu posicionamento em relação aos objetos e o posicionamento da cabeça em relação ao próprio corpo, para então decidir quais objetos o robô vê. A posição é convertida para um espaço 2D que possui como origem a posição do robô. Este é um ponto de vista egocêntrico onde robô é sempre a origem do espaço. A posição é convertida para coordenadas polares. O uso de coordenadas polares permite tomar a primeira coordenada, o raio, como a distância do objeto em relação ao robô. A segunda coordenada, o ângulo, informa de maneira simples o lado no qual o objeto está em relação ao robô. Como consideramos o ângulo zero como sendo a frente do robô, ângulos positivos representam objetos à esquerda do robô, e ângulos negativos representam objetos à direita do robô. Os objetos percebidos pelo robô são adicionados a uma lista, que é enviada para a mente do robô.

Esses dados são enviados a uma mente – cujo funcionamento será descrito mais adiante – conectada à simulação. Cada robô presente na simulação fica imóvel até que uma mente conecte-se ao sistema enviando comandos àquele robô. Cada robô pode ser controlado por apenas uma mente, mas inúmeros robôs podem estar presentes na simulação sendo controlados simultaneamente.

Caso a mente envie comandos ao robô simulado, a simulação faz com que ele execute esses comandos, isto é, caso o comando seja para o robô andar, por exemplo, a simulação irá atualizar a posição do robô no mundo. A simulação não permitirá que o robô se mova caso outro objeto esteja ocupando a posição para onde a mente o comandou a ir, tratando colisões e não permitindo que um objeto se mova através de outros. No caso do chute, uma física simples usando colisões inelásticas foi implementada para melhorar a usabilidade do simulador. O objetivo não é simular com perfeição a física do mundo real, até mesmo porque as irregularidades da superfície – que na prova de futebol de robôs humanoides é um gramado sintético – são muito grandes e aumentam enormemente a complexidade do cálculo da trajetória da bola após o chute. Ao invés disso, a bola traça uma trajetória em linha reta em um ângulo adjacente à frente do robô simulado.

3.2. Mente

A mente é a parte do sistema que é desenvolvida pelo usuário. Para que esta tarefa seja feita, existe uma API em Python que fornece ao usuário funções, representadas na Figura 5, para que ele acesse as informações percebidas pelo robô e para que ele envie comandos a este robô.

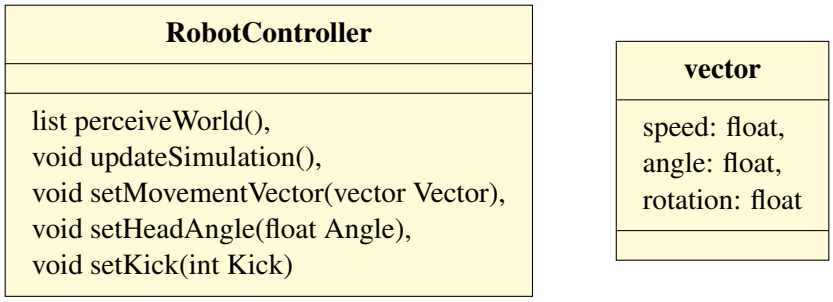

Figura 5. A API fornece funções para que o usuário comunique-se com o robô.

A primeira coisa a ser feita pelo usuário deve ser atualizar a simulação. A função *updateSimulation* comunica à simulação possíveis comandos destinados ao robô e recebe uma lista de objetos percebidos por ele. Com a certeza de que a percepção do robô foi atualizada, o usuário pode acessar a lista de objetos percebidos pelo robô através da função *perceiveWorld*. Com esses dados em mãos, o usuário pode programar sua lógica de comportamento e utilizar as demais funções para controlar o robô.

Em geral, robôs humanoides podem caminhar omnidirecionalmente, i.e., em todas as direções, incluindo caminhada para trás, caminhada lateral e etc. Esses robôs podem também girar em seu próprio eixo, rotacionando sua face para outra direção. Além disso, eles podem andar em alguma direção e girar simultaneamente. Por causa dessa capacidade, o vetor de movimento precisa indicar a direção e a velocidade da caminhada, mas também um ângulo de rotação.

A função *setMovementVector* recebe um vetor de movimento que irá controlar a caminhada do robô. Esse vetor possui três coordenadas. A primeira, *speed*, representa a magnitude do vetor e é um valor entre 0 e 1 que descreve a velocidade proporcional na qual o robô deverá andar. A segunda coordenada, *angle*, informa a direção do vetor, considerando 0 como a frente do robô. A terceira e última coordenada do vetor de movimento, *rotation*, refere-se à rotação que o robô deve fazer enquanto caminha na direção do vetor.

O usuário ainda pode controlar a posição da cabeça do robô através da função *setHeadAngle*, que recebe um ângulo. Por fim, o usuário pode enviar um comando de chute. No caso do sistema desenvolvido, o chute não é parametrizado, i.e., o robô não pode chutar a bola em diferentes direções. Os tipos de chute são limitados ao chute com o pé esquerdo ou chute com o pé direito. Sendo assim, o valor que a função de chute recebe deve ser -1, para um chute com o pé esquerdo, 1, para um chute com o pé direito, ou 0, para que o robô não chute ou pare de chutar.

3.3. Desenvolvimento de uma mente

Ao desenvolver uma mente que controla um robô, o usuário pode utilizar a abordagem que preferir. Uma abordagem muito utilizada é o uso de árvores de decisão. As árvores de decisão geram um código baseado em condições que resultam em novas condições ou ações. Um exemplo simples de mente que utiliza essa abordagem seria um código que analisa a distância da bola. Caso a bola esteja longe demais para que o robô a chute, a mente comanda o robô para que ele ande até a bola. Caso contrário (se o robô está perto o suficiente para chutar a bola), a mente analisa a posição da bola e decide se o chute deve

ser feito com o pé direito ou esquerdo.

A partir deste simples exemplo fica claro que o usuário deve ficar atento ao fato de que o sistema o oferece uma percepção atualizada do ambiente. Sendo assim, ele é responsável por armazenar informações sobre os objetos, dando uma espécie de memória ao robô. Seguindo neste exemplo, caso o robô não veja mais a bola, o usuário tem o papel de armazenar a informação sobre em qual dos lados a bola foi vista pela última vez.

Uma vez que as informações e comandos trocados entre simulação e mente são dados pertinentes no sistema real do robô, é possível substituir a simulação por um robô real de maneira transparente fazendo a mente comunicar-se diretamente com ele. Ou seja, ao executar o código da mente no robô real, este funcionará da mesma maneira que funcionou na simulação sem a necessidade de transcrever o código. Para o programador do comportamento, é indiferente se as informações sobre o ambiente vêm de uma simulação ou de um robô real e é igualmente irrelevante se ele envia comandos ao robô simulado ou ao robô real, pois, uma vez que o padrão de comunicação seja mantido, o código desenvolvido e testado no ambiente de simulação irá manter a compatibilidade com o hardware.

O sistema desenvolvido teve como linguagem de programação escolhida linguagem Python por ser uma linguagem poderosa com uma curva de aprendizagem suave. Para a interface gráfica foi utilizada a biblioteca gráfica PyGame que funciona como uma camada sobre a biblioteca gráfica SDL. A comunicação é feita pela rede utilizando sockets e os dados são enviados no formato JSON por ser um formato conciso e de fácil entendimento.

4. Discussão

Um fator de debate presente durante o desenvolvimento e o uso do sistema descrito neste trabalho foi o limite de abrangência do que é descrito como sendo a mente do robô. Até que ponto a parte descrita aqui como mente deve ser responsável pelo controle das ações do robô? Tanto a simulação quanto a API oferecida para o desenvolvedor do comportamento do robô dependem da decisão de até que ponto o comportamento é responsável pelo controle das funções do robô. Um exemplo disso é a caminhada, que se dá de maneira separada. O comportamento não controla o equilíbrio do robô enquanto ele caminha e, caso o robô venha a cair, a parte do sistema responsável por fazê-lo levantar-se é o módulo de caminhada. Considerando isso podemos abstrair outras partes do processo.

Usando como exemplo a experiência de uma equipe que utilizou o sistema aqui descrito, foi decidido que o controle da cabeça do robô seria feito pelo comportamento, mas esse controle também pode ser abstraído. A simulação poderia controlar a cabeça do robô fazendo uma busca por objetos e parar essa busca uma vez que o objeto fosse encontrado. Ela poderia informar ao comportamento somente a posição dos objetos no mundo, ignorando totalmente a posição real da cabeça. Seguindo esta abordagem alternativa, um sistema de controle da cabeça do robô funcionaria de maneira separada ao comportamento, tornando desnecessária a função de controle da cabeça presente na API.

Considerando, então, que o comportamento não tem controle algum sobre a cabeça, alguns testes foram feitos. Durante os testes foi decidido utilizar uma abordagem para a detecção de lados do campo, auxiliando na localização do robô. A abordagem escolhida utilizava imagens do horizonte além do campo de futebol para fazer o cálculo

da direção. Para tomar tais imagens, o robô deve posicionar sua cabeça de maneira que ele veja o horizonte. O problema é que a localização do robô cabe ao comportamento, então a decisão de quando olhar para o horizonte também cabia ao comportamento. Mas, neste caso, o comportamento não tinha controle sobre os movimentos da cabeça, travando o desenvolvimento até que uma outra abordagem fosse desenvolvida.

Este é um pequeno exemplo de que a arquitetura utilizada para este sistema de simulação pode ser diferente. Outro exemplo é que poderíamos armazenar na API informações sobre os objetos detectados durante um longo período de tempo, criando uma camada que simula uma memória sobre a posição dos objetos. Oferecendo esta funcionalidade, retiramos esta responsabilidade do usuário.

Estes exemplos mostram que a abrangência do controle e da responsabilidade que o desenvolvedor do comportamento tem é arbitrária. Podemos dar mais ou menos controle ao comportamento dependendo das necessidades da equipe que utilizará o sistema. O ponto central deste trabalho, entretanto, é que um sistema de simulação com foco no comportamento é de grande ajuda no desenvolvimento de robôs humanoides que jogam futebol, independentemente da abrangência do comportamento.

5. Trabalhos Relacionados

O uso de simuladores é de grande ajuda em projetos de robótica e isso prova-se verdade observando-se o uso extensivo de simuladores em tais projetos [Burkhard 2015] [Baltes et al. 2016]. A maioria deles encaixa-se em uma categoria de simuladores focados na física e dinâmica do mundo real. Por outro lado, existe um número muito pequeno de simuladores aplicados a futebol de robô com foco no desenvolvimento de estratégias de comportamento e de tomadas de decisão. Essas duas categorias de simuladores são exemplificadas pelos dois trabalhos relacionados descritos a seguir.

5.1. Gazebo

O Gazebo [Koenig and Howard 2013] é um simulador 3D amplamente utilizado em aplicações de robótica por ser extremamente poderoso, capaz de simular dinâmica e física do mundo real. Entretanto, a criação de estratégias de comportamento para futebol de robôs pode ser simplificada uma vez que o módulo de comportamento do robô pode abstrair outras partes do funcionamento do robô. Nesse cenário, o Gazebo acaba tornando a tarefa muito complexa e seu uso inviável, pois ele simula todos os aspectos do robô com um nível de abstração baixo.

5.2. RoboCup Soccer Simulator Server

Ao buscar um simulador com um alto nível de abstração, com o foco em desenvolvimento de estratégias de comportamento para futebol de robôs humanoides, é possível encontrar o RoboCup Soccer Simulator Server (RCSSS) [Noda et al. 2016]. O RCSSS é o simulador oficial usado pela RoboCup Federation como base para a liga de simulação de futebol 2D. O problema de usar o RCSSS para a situação descrita neste trabalho é que ele não é um simulador de robótica, mas sim um simulador de futebol. O RCSSS não possui nenhum compromisso de compatibilidade com o hardware, ignorando muitas características de um robô real. Sendo assim, um código desenvolvido e testado usando o RCSSS não necessariamente funcionará em um robô real.

5.3. Comparação com Trabalhos Relacionados

Conforme descrito, o Gazebo é um simulador com baixo nível de abstração com foco em aspectos da física e do robô do mundo real. Nesse sentido, o trabalho aqui apresentado oferece vantagens em relação ao Gazebo pois o usuário pode abstrair partes complexas do projeto, focando especificamente na tarefa de criar estratégias de comportamento. Essa característica simplifica a tarefa, aumentando a eficiência do desenvolvimento.

Por outro lado, o RCSSS é um simulador de futebol com alto nível de abstração mas que se propõe a ser aplicado em um outro nicho. Este simulador foca na simulação como um fim em si. Entretanto, o sistema aqui apresentado oferece uma compatibilidade com o hardware que não está presente no RCSSS, através de uma API para que o usuário comunique-se com um robô real. Dessa maneira, o sistema oferece um alto nível de abstração mas, ao mesmo tempo, mantendo um compromisso com a compatibilidade com o hardware. Assim, o sistema aborda um ponto onde haviam lacunas deixadas pelos trabalhos aqui relacionados.

6. Conclusão

O desenvolvimento de um robô para a prova de futebol de robôs humanoides da RoboCup envolve muitos desafios. Com o foco no desenvolvimento de estratégias de comportamento, o uso de um simulador computacional pode aumentar a eficiência e diminuir o custo do projeto. Assim, um simulador computacional foi criado com o objetivo de prover um alto nível de abstração para o usuário, mas sem abrir mão da compatibilidade com o hardware.

Nos anos de 2015 e 2016, uma equipe de futebol de robôs humanoides utilizou o simulador extensivamente durante suas participações em competições com grande sucesso. O sistema deixou de ser um simulador para ser um ambiente de simulação e criação de estratégias de comportamento para futebol de robôs, uma vez que ele oferece uma API para a criação de tais estratégias.

O processo de criação e teste de estratégias de comportamento passou a ser bem mais eficiente pois problemas mecânicos e de hardware durante os dias de competição não limitaram o desenvolvimento da mente do robô, como acontecia anteriormente. Além disso os testes da estratégia de comportamento no robô real se tornaram bem mais raros diminuindo a frequência de defeitos mecânicos que ocorrem por desgaste.

Após observar-se a equipe utilizando o sistema proposto, percebe-se que a utilização de um sistema de simulação para o desenvolvimento e testes das estratégias de comportamento para futebol de robôs humanoides é de grande ajuda. Existe um aumento claro na eficiência durante a criação do comportamento uma vez que o desenvolvimento das estratégias se dá de maneira independente do funcionamento do robô real. Além disso, o custo do projeto diminui, pois os testes com o robô real são mais raros, diminuindo o número de defeitos em motores e partes mecânicas que precisariam ser substituídas por partes novas.

O que fica claro é que a utilização de um ambiente de simulação para a criação de estratégias de comportamento para futebol de robôs aumenta a eficiência e diminui o custo do projeto, mas também que, conforme novas técnicas e abordagens são criadas, um simulador com esse propósito deve ser constantemente adaptado para as necessidades

da equipe que o utiliza, conforme discutido na seção 4.

Como melhoria para o sistema, a equipe pretende adaptar o ambiente de simulação de maneira que use o framework ROS (Robot Operating System) que oferece uma estrutura de módulos que comunicam-se através de tópicos. O framework é amplamente utilizado em aplicações de robótica, sendo utilizado inclusive por algumas equipes de futebol de robôs humanoides. Fazendo com que o ambiente de simulação seja um módulo do ROS, é possível conectá-lo facilmente a qualquer sistema que use o framework, potencializando a participação e o uso pela comunidade.

Além disso, utilizando-se da capacidade do sistema aqui desenvolvido de simular mais do que apenas um robô, pode-se usar este sistema de simulação para o teste de sistemas que utilizem a comunicação entre robôs para a solução de tarefas. Pode-se, por exemplo, criar um sistema que utilize a comunicação entre robôs para uma melhor localização dos robôs no campo ou para estratégias de jogadas ensaiadas, e utilizar o sistema de simulação aqui apresentado para a execução de testes.

Referências

Baltes, J., Bagot, J., Sadeghnejad, S., Anderson, J., and Hsu, C.-H. (2016). Full-body motion planning for humanoid robots using rapidly exploring random trees. *KI-Künstliche Intelligenz*, pages 1–11.

Burkhard, H.-D. (2015). Simulated humanoid robots for e-learning. In *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 1–1. IEEE.

Campbell, M., Hoane, A. J., and Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1):57–83.

Kitano, H., Kitano, H., Asada, M., Asada, M., Kuniyoshi, Y., Kuniyoshi, Y., Noda, I., Noda, I., Osawai, E., Osawai, E., Matsubara, H., and Matsubara, H. (1998). RoboCup: A challenge problem for AI and robotics. *RoboCup-97: Robot Soccer World Cup I*, 18(1):1–19.

Koenig, N. and Howard, A. (2013). Gazebo-3d multiple robot simulator with dynamics (2003). *URL: http://gazebosim. org*, 3.

Noda, I., Boedecker, J., Akiyama, H., Vatankhah, H., Stone, P., Obst, O., and Dorer, K. (2016). The robocup soccer simulator. *URL: https://sourceforge.net/projects/sserver/files/rcssserver/*.

Pfeifer, R., Scheier, C., and Illustrator-Follath, I. (2001). *Understanding intelligence*. MIT press.

Russell, S., Norvig, P., and Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25.