UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E DA TECNOLOGIA
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

GUILHERME FABRIS DE SOUZA

SISTEMA MULTIAGENTES DE SIMULAÇÃO DE FUTEBOL DE ROBÔS: EXPLORANDO MECANISMOS DE APRENDIZAGEM E MOTIVAÇÃO

CAXIAS DO SUL – RS
2016

GUILHERME FABRIS DE SOUZA

SISTEMA MULTIAGENTES DE SIMULAÇÃO DE FUTEBOL DE ROBÔS: EXPLORANDO MECANISMOS DE APRENDIZAGEM E MOTIVAÇÃO

Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Engenharia de Controle e Automação pela Universidade de Caxias do Sul, orientado pela Profa. Carine Geltrudes Webber.

Caxias do Sul – RS

2016

RESUMO

Um sistema multiagentes é formado por vários agentes que trabalham em grupo para realizar um determinado conjunto de tarefas. Um agente pode ser definido como uma unidade autônoma capaz de exercer alguma ação em um ambiente. Em um sistema multiagentes os agentes encapsulam funcionalidades de um sistema e compartilham um ambiente onde estão situados. Os agentes possuem diferentes habilidades de interação e organização, que permitem que resolvam problemas de forma coletiva, seja cooperando ou colaborando entre si. Dadas as características de um sistema multiagentes é notável o seu potencial: a possibilidade da decomposição de um problema em problemas menores, cada um resolvido por um agente. Por ser mais facilmente implementado, cada agente demanda menor capacidade computacional do que um único módulo ou software. Almejando mais veracidade na tomada de decisões feita por sistemas computacionais, a computação afetiva (um subcampo da Inteligência Artificial) realiza pesquisas relacionadas às emoções modeladas em computadores. Inserir modelos de emoção em sistemas computacionais é considerado útil pela comunidade acadêmica. Uma das razões pelas quais se defende esta abordagem é que na natureza as emoções ajudam espécies de animais a sobreviverem em ambientes complexos dinâmicos, incertos e com recursos limitados. Ela pode auxiliar ainda aos sistemas a responderem adequadamente a situações desconhecidas. Neste contexto este trabalho tem como proposta utilizar mecanismos de aprendizagem e motivação em um agente de software aplicado ao futebol de robôs. A partir de mecanismos de aprendizagem automática deseja-se ensinar os agentes (jogadores) a desempenhar melhor sua função em campo. Para isso será abordado o conceito de emoção como mecanismo motivacional e decisório do agente. Como forma de avaliação implementou-se agentes no contexto de um jogo de futebol. Foi possível realizar testes e comparações em termos de comportamento com agentes que não aplicam estes conceitos. Conclui-se que apesar do satisfatório desempenho do ambiente de simulação de futebol de robôs, do método de aprendizagem por reforço ter sido aplicado com sucesso, inserir o conceito de emoção nos agentes para obter um diferencial na tomada de decisão em uma partida de futebol, necessita que o agente possua uma percepção bastante detalhada e certa compreensão do que está

ocorrendo ao seu redor para que o resultado destes eventos sirva de suporte do sistema de emoções, assim o conceito de emoção possa ser aplicado como um diferencial.

Palavras-chave: Sistema multiagentes, Agentes, Futebol de robôs, Aprendizagem por reforço, Motivação.

ABSTRACT

A multi-agent system is formed by several agents that work in group to perform a certain set of tasks. An agent can be defined as an autonomous unit capable of exerting some action in an environment. In a multi-agent system, agents encapsulate a system's functionality and share an environment where it is located. The agents have different interaction and organization skills, which allow them to solve problems collectively, either by cooperating or collaborating with each other. Given the characteristics of a multi-agent system is remarkable its potential: the possibility of decomposing a problem into smaller problems, each solved by an agent. Because it is more easily implemented, each agent requires less computational capacity than a single module or software. Aiming for more veracity in computer-based decision making, affective computing (a subfield of Artificial Intelligence) conducts research related to computer-modeled emotions. Inserting emotion models into computer systems is considered useful by the academic community. One of the reasons for this approach is that in nature, emotions help animal species survive in complex, uncertain, resource-constrained environments. It can also help systems respond appropriately to unknown situations. In this context, the purpose of this work is to use learning and motivation mechanisms in a software agent applied to robot soccer. From automatic learning mechanisms, it is desired to teach the agents (players) to better perform their function in the field. For this, the concept of emotion will be approached as the agent's motivational and deciding mechanism. As a form of evaluation agents were implemented in the context of a football game. It was possible to perform tests and comparisons in terms of behavior with agents that do not apply these concepts. It is concluded that despite the satisfactory performance of the robot soccer simulation environment, the reinforcement learning method has been applied successfully, to insert the concept of emotion in the agents to obtain a differential in the decision making in a soccer game, Requires the agent to have a very detailed perception and a certain understanding of what is happening around him so that the result of these events supports the system of emotions, so the concept of emotion can be applied as a differential.

Keywords: Multi-agent system, Agents, Robot soccer, Reinforcement learning, Motivation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um sistema multiagentes é formado por vários agentes que trabalham em grupo para realizar um determinado conjunto de tarefas (WOOLDRIDGE, 2002). Um agente pode ser definido como uma unidade autônoma capaz de exercer alguma ação em um ambiente. Em um sistema multiagentes os agentes encapsulam funcionalidades de um sistema e compartilham um ambiente onde estão situados. Os agentes possuem diferentes habilidades de interação e organização, que permitem que resolvam problemas de forma coletiva, seja cooperando ou colaborando entre si.

Diversas metodologias já foram desenvolvidas para a concepção de sistemas multiagentes. As metodologias têm em comum a perspectiva de que um sistema multiagentes deve ser composto por, pelo menos quatro elementos: agentes, ambiente, interação e organização (RUSSELL e NORVIG, 1997). Os agentes constituem os componentes principais do sistema, sendo responsáveis pela modelagem do conhecimento do sistema, bem como pela tomada de decisão que em geral deve emergir de ações coordenadas, através da cooperação entre agentes e formação de equipes. O ambiente compreende o contexto ou o escopo onde os agentes se situam. As interações permitem que os agentes se comuniquem e desta maneira resolvam problemas coletivamente. A organização é um fator relevante, pois estabelece um modelo de funcionamento do sistema, em geral com uma inspiração social ou biológica. De fato, apesar de o sistema multiagentes poder ser inserido em diversos tipos de problemas, cada aplicação requererá uma configuração multiagentes particular. Recentemente tem-se investigado questões relacionadas a inclusão de aspectos emocionais em agentes (PICARD, 2002). O desenvolvimento de agentes que modelem emoções tem o intuito de tornar o comportamento dos agentes mais eficiente, uma vez que se observa que na natureza as emoções ajudam os animais a sobreviverem em um ambiente complexo, dinâmico, incerto e com recursos limitados. De forma similar, a simulação das emoções poderia auxiliar os agentes a responderem a situações que necessitem de uma ação rápida de maneira adequada (STEHOUWER, 2004). Por esta razão, o mecanismo de emoção pode ser útil computacionalmente.

O tema de pesquisa proposto envolve, portanto, a investigação da validade da inserção de um modelo de emoções em agentes que tem por objetivo disputar uma partida de futebol.

1.1 OBJETIVOS

Esta seção descreve o objetivo geral e os objetivos específicos propostos para o presente trabalho.

1.1.1 Objetivo Geral

Utilizar um método de aprendizagem por reforço e a modelagem de emoções na concepção da simulação de futebol de robôs.

1.1.2 Objetivos Específicos

a) Pesquisar sobre sistemas multiagentes.
b) Pesquisar sobre métodos de aprendizagem por reforço.
c) Detalhar o problema de gerenciamento de um time de futebol de robôs.
d) Definir um modelo de arquitetura de futebol simulado de robôs.
e) Definir a arquitetura para o sistema multiagentes.
f) Especificar os agentes.
g) Criar um ambiente de simulação de agentes específico para um jogo de futebol.
h) Codificar os comportamentos dos agentes em linguagem de programação compatível com o ambiente de simulação.
i) Promover testes e avaliações do desempenho dos agentes em um campeonato simulado de futebol de robôs.

1.2 ORGANIZAÇÃO DO DOCUMENTO

Esta monografia está organizada em capítulos. A estrutura deste trabalho será detalhada a seguir.

Capítulo 2 discute a fundamentação teórica necessária para o entendimento do trabalho. São detalhados os conceitos de agentes e suas arquiteturas principais, sistemas multiagentes, são apresentados alguns conceitos de computação afetiva e aprendizagem por reforço. Estes conceitos são necessários para a compreensão técnica abordada neste trabalho.

Capítulo 3 aborda aplicações de sistemas multiagentes, isto inclui exemplos de aplicações dá e o enfoque voltado para sistemas multiagentes no futebol de robôs, onde são apresentadas as duas principais federações assim como as categorias de competição robótica de cada federação.

Capítulo 4 detalha o desenvolvimento do ambiente de simulação de futebol de robôs, e o capítulo 5 apresenta os resultados obtidos.

Por fim no Capítulo 6 conclui-se o trabalho, apresentando as suas contribuições e indicante possíveis trabalhos futuros. Um apêndice contém as iterações realizadas para obter os resultados apresentados.

2 SISTEMAS MULTIAGENTES: CONCEITOS E MODELOS

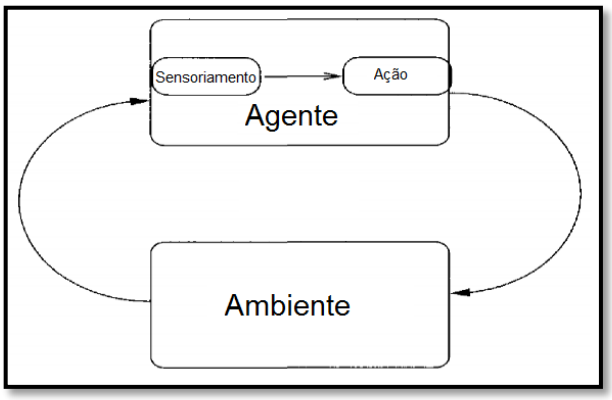
A Inteligência Artificial, segundo (LUGER, 2004), é o estudo dos mecanismos subjacentes ao comportamento inteligente por meio da construção e da avaliação de artefatos que tentam representar esses mecanismos. O campo de agentes inteligentes constitui um framework conceitual para que se possa criar tais artefatos. Este capítulo aborda os fundamentos da área de sistemas multiagentes, com ênfase em processos oriundos da computação afetiva e da aprendizagem por reforço.

2.1 AGENTES INTELIGENTES

Na literatura, a definição de agente não é única, porém podemos destacar duas definições que se encaixam com o objetivo deste trabalho. Segundo (JUCHEM e BASTOS, 2001), "*um agente é uma entidade de software, que exibe um comportamento que lhe permite executar a maior parte de suas ações sem interferência direta de agentes humanos ou de outros agentes computacionais, possuindo controle total sobre suas ações e estado interno*". Para (WOOLDRIDGE, 2002) "*Um agente é um sistema computacional situado em algum ambiente com capacidade autônoma de executar ações neste ambiente a fim de alcançar seus objetivos*".

A Figura 1 ilustra um modelo de interação entre agente e ambiente. Nela pode-se observar que um agente percebe seu ambiente por meio de sensores de entrada e age sobre ele por meio de ações. O ciclo de ação-percepção é necessário para que o agente perceba continuamente seu ambiente e aja conforme seu objetivo.

Figura 1 - Modelo de agente reativo



Fonte: (WOOLDRIDGE, 2002), adaptado para português

A fim de que um agente possa reagir ao seu ambiente em vista de seus objetivos, ele deve possuir segundo (JUCHEM e BASTOS, 2001) as seguintes propriedades: pró-atividade, capacidade de reação e habilidade social.

A pró-atividade diz respeito a capacidade de um agente apresentar um comportamento objetivo e tomar iniciativas a fim de satisfazer os seus objetivos. A capacidade de reação trata de como o agente vai perceber e reagir às alterações ocorridas em seu ambiente a fim de atingir os seus objetivos. Por último, a habilidade social envolve a capacidade de interagir com outros agentes computacionais ou humanos em prol da satisfação de seus objetivos de projeto.

Estas habilidades são requeridas uma vez que o agente, quando situado em um domínio de complexidade razoável, não possui o controle completo deste ambiente. Ainda segundo (WOOLDRIDGE, 2002), o agente pode ter na maioria dos casos apenas uma percepção e controle parciais sobre seu ambiente.

Outro aspecto que deve ser considerado segundo (WOOLDRIDGE, 2002), é que, do ponto de vista de um agente, uma mesma ação realizada mais de uma vez em circunstâncias muito similares pode resultar em um efeito diferente. Partindo deste pressuposto um agente mesmo quando situado em um ambiente simples (modelado com poucas variáveis), deve estar preparado para a possibilidade de que se uma ação não produz o resultado esperado, todo seu plano de ação pode se tornar falho, necessitando ser revisto. Caso isso não ocorra, o agente corre o risco de não alcançar seu objetivo. Baseado neste fato precisa-se assumir que, em geral os ambientes em que os agentes estão inseridos são não-determinísticos (não totalmente previsíveis). Logo todas as habilidades mencionadas são necessárias para que o agente esteja apto a lidar com um ambiente não previsível, respondendo as mudanças e garantindo a robustez do sistema como um todo.

A capacidade de um agente de modificar o ambiente o qual está situado é determinado pelo repertório de ações disponíveis. Mas deve ser observado que nem todas as ações podem ser realizadas em qualquer situação, cada ação contém uma pré-condição associada, que definem as possíveis situações em que podem ser aplicadas (WOOLDRIDGE, 2002).

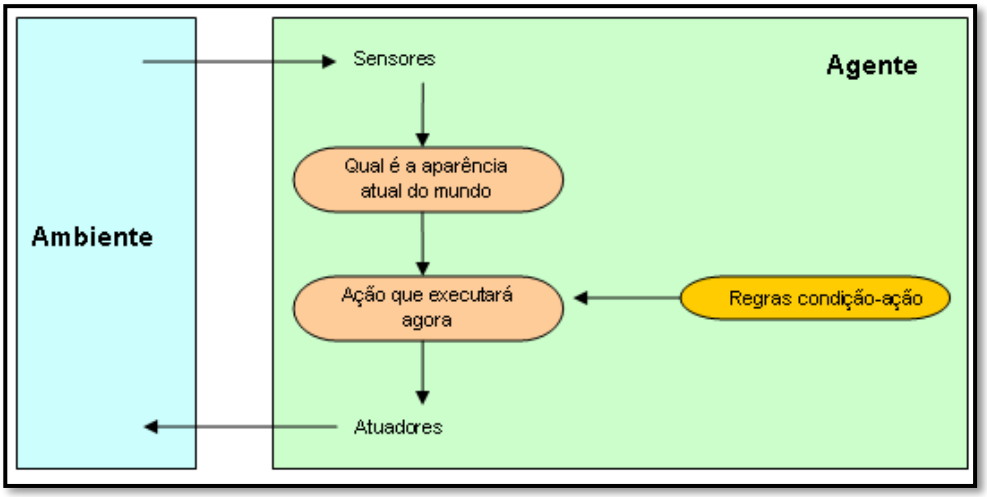
O principal problema encarado por agentes inteligentes é de decidir quais ações devem realizar a fim de que seus objetivos sejam atingidos.

2.2 ARQUITETURA DE AGENTES

Em um projeto se deve classificar o ambiente no qual o agente será situado, a fim de determinar se o mesmo é ou não determinístico. A classificação do ambiente auxilia o projetista na decisão fundamental em relação a qual arquitetura o agente será modelado (RUSSELL e NORVIG, 1997).

Na literatura, são abordadas duas arquiteturas que classificam os agentes como reativos ou deliberativos. Um agente puramente reativo é baseado em modelos simples, como o estímulo-resposta e baseado em arquiteturas de subsunção (ZAMBERLAM e GIRAFFA, 2001, p. 3). Nestes modelos, a escolha de uma ação ou resposta, está diretamente interligada com a ocorrência de um conjunto de eventos, ou estímulos, percebidos pelo agente no ou do ambiente, ou pelas mensagens enviadas por outros agentes. A Figura 2 ilustra um exemplo simples de arquitetura para um agente reativo.

Figura 2 - Exemplo de arquitetura de um agente reativo simples



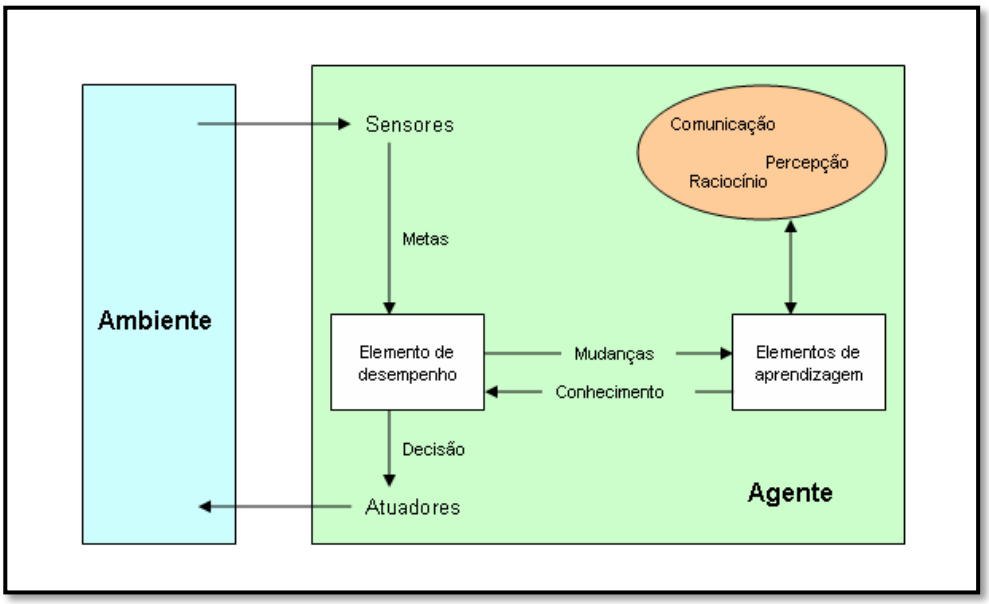
Fonte: http://www.inf.ufes.br/~liviaufmt/Disciplina/1.1.1%20agentes.htm

Agentes deliberativos, por outro lado, planejam suas ações por meio de intenções e desejos que conduzem seu comportamento e os tornam capazes de fazer escolhas entre as possíveis ações. Este tipo de agente pode ser ainda baseado em arquiteturas que conduzem seu estado mental para um contexto relacionado com crenças, desejos e vontades, e até emoções.

As arquiteturas que se destacam neste conceito é, a arquitetura BDI (*beliefs, desires and intentions*), e a arquitetura OCC (*Ortony, Clore and Collins*) ambas utilizam referências psicológicas que se baseiam no comportamento humano (JAQUES, 2005).

Na figura 3 pode ser visto um exemplo ilustrativo da arquitetura de um agente deliberativo, o qual diferente do agente reativo, não possui regras de ação-reação relacionadas a percepção atual do mundo, mas sim conceitos de decisão complexos onde o seu conhecimento sobre o ambiente, suas crenças, desejos e vontades são levados em consideração durante a escolha de uma ação/reação.

Figura 3 - Exemplo de arquitetura de um agente cognitivo



Fonte: http://www.inf.ufes.br/~liviaufmt/Disciplina/1.1.1%20agentes.htm

2.3 SISTEMAS MULTIAGENTES

Os Sistemas Multiagentes (SMA) são sistemas compostos por um conjunto de agentes que interagem entre si de forma organizada a fim de alcançar um objetivo em comum. Este tipo de sistema é normalmente aplicado em problemas para os quais um único agente (processo) não seria capaz de resolver, ou seria inviável (extremamente complexo) encontrar uma solução ótima (BITTENCOURT, 2006).

Em um sistema multiagentes o objetivo desejado, ou objetivo global, é fragmentando em objetivos locais e distribuído para cada agente que compõe o sistema. Desta forma partindo dos objetivos mais simples é possível combinar os resultados parciais, obtendo assim uma solução para o objetivo global (LUSTOSA, 2004). Contudo, para obter uma solução ótima global é necessário, segundo (WOOLDRIDGE, 2002), que o sistema multiagentes contenha os seguintes aspectos: coordenação, cooperação entre agentes e formação de equipes a fim de facilitar a resolução dos objetivos locais.

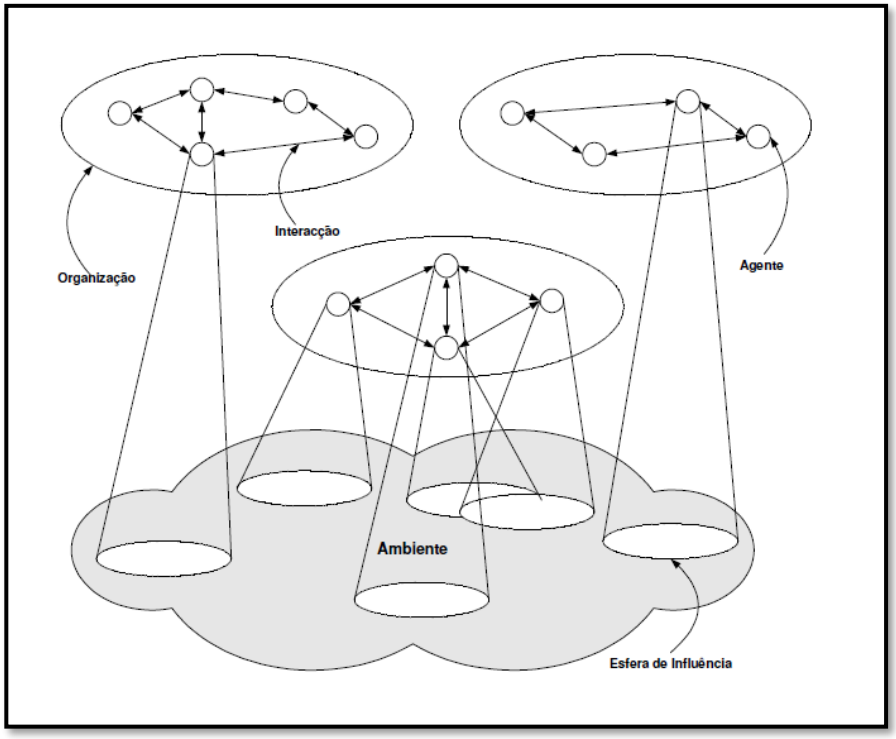
Em um sistema multiagentes bem coordenado, observa-se a coerência do resultado atingido. Pois, cada agente possui apenas uma visão local e incompleta do ambiente. Sendo assim, a coordenação é fundamental no planejamento e alinhamento de tarefas, para evitar conflitos e retrabalho, alocar recursos limitados, conciliar preferências e buscar soluções de caráter global. A coerência em um sistema multiagentes faz referência ao quão bem o sistema multiagentes se comporta em conjunto, quando avaliado em alguma dimensão. Ou seja, pode-se avaliar em termos da qualidade da solução encontrada, da eficiência do sistema, da quantidade de recursos utilizados e do decaimento do desempenho em relação a presença de incertezas ou falhas (WOOLDRIDGE, 2002).

Outra forma dos agentes interagirem é por meio de mecanismos de cooperação. A cooperação entre agentes pode ser vista como uma ferramenta de auxilio, onde agentes trocam informações relevantes a fim de facilitar o cumprimento da tarefa que lhes foi atribuída. Na literatura, são discutidos níveis de cooperação entre agentes que podem variar desde totalmente cooperativos até sistemas antagônicos. Sistemas totalmente cooperativos em geral caracterizam-se por necessitar de uma maior demanda de comunicação. Em sistemas antagônicos os agentes podem optar por não cooperar entre si, ou até tentar impedir as ações de outro agente que conflite com seus objetivos. Em contraste com sistemas cooperativos, sistemas antagônicos não necessitam de uma demanda alta de comunicação, mas por estas razões, a eficiência do sistema pode ser comprometida (BAZZAN, 2010).

Formar equipes em prol da realização de um objetivo, é uma decisão que o agente deve tomar quando em seu conjunto de ações não existir um meio de solucionar sozinho uma tarefa. A Figura 4 ilustra a visão geral de um sistema

multiagentes, onde os círculos menores representam agentes interagindo entre si. Os círculos em pontilhado indicam as elipses de relações entre agentes. Pode ser observado ainda, a formação de equipes e interação entre os agentes, os quais utilizam a característica de cooperação para que juntos construam uma visão mais ampla (completa) do ambiente, a fim de atingir seus objetivos.

Figura 4 - Visão geral de um sistema multiagentes

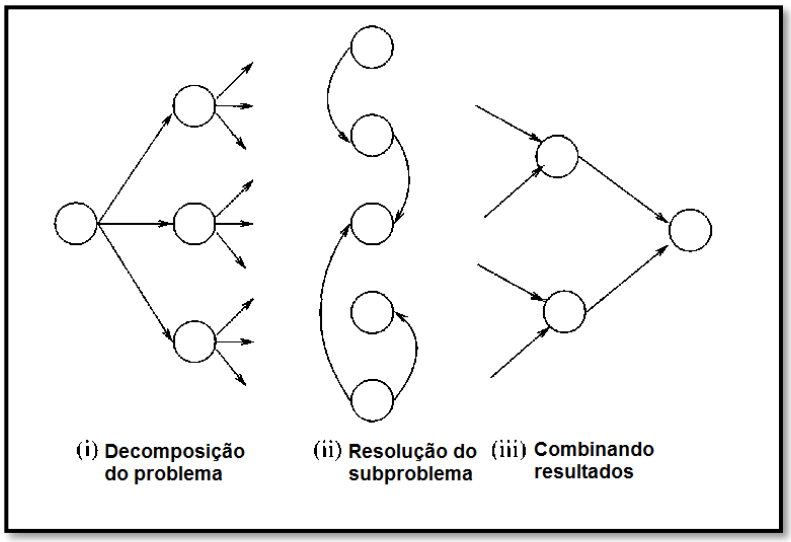


Fonte: (WOOLDRIDGE, 2002)

Assim que a solução parcial do problema é encontrada, ela dele ser compartilhada para que o sistema produza a solução global para o problema. Este processo pode ser hierárquico, onde as soluções parciais são unidas em diferentes níveis da hierarquia. A Figura 5 demonstra um modelo de decomposição de tarefas, onde os agentes subdividem as tarefas hierarquicamente, formando grupos, cooperando entre si e unindo seus resultados para construir a solução global. Na Figura 5 (i) os agentes possuem o conhecimento da subdivisão do problema, e o subdividem de forma hierárquica a fim de formar níveis diferentes de combinação de problemas locais. Figura 5 (ii) demonstra a cooperação entre agentes do mesmo nível de subdivisão, os agentes cooperam entre si para solucionar o problema local e combinar seus resultados a fim de aumentar a chance de obter uma solução

adequada para o problema local. Por fim na Figura 5 (iii) as soluções são combinadas, partindo de tarefas e passando por todos os níveis até constituir o problema global.

Figura 5 - Exemplo de subdivisão de tarefas, cooperação e compartilhamento de resultados



Fonte: (WOOLDRIDGE, 2002)

2.4 COMPUTAÇÃO AFETIVA

Almejando mais veracidade na tomada de decisões feita por sistemas computacionais, a computação afetiva, um subcampo da Inteligência Artificial, realiza pesquisas relacionadas à emoções modeladas em computadores. Este campo se desenvolve a partir de dois ramos principais. O primeiro estuda mecanismos de reconhecimento das emoções humanas na interação homem-máquina. O segundo ramo estuda a simulação de emoções em máquinas a fim de modelar sistemas computacionais inteligentes que suas tomadas de decisões pareçam mais reais (JAQUES, 2005).

Inserir modelos de emoção em sistemas computacionais é considerado útil por diversos autores (STEHOUWER, 2004; PICARD, 2002; ZAMBERLAM e GIRAFFA, 2001; JAQUES, 2005). Uma das razões pelas quais se defende esta abordagem é que na natureza as emoções ajudam espécies de animais a sobreviverem em ambientes complexos dinâmicos, incertos e com recursos limitados. Ela pode

auxiliar ainda aos sistemas a responderem adequadamente a situações desconhecidas.

Para entender a estrutura dos modelos mais conceituados que tentam transmitir às máquinas a essência do que é compreendido como emoção, e como ela se manifesta, é necessário entender como as emoções são vistas pelos pesquisadores da área.

As emoções podem ser vistas como manifestações que envolvem uma avaliação subjetiva de uma situação, cuja natureza particular é determinada pela maneira que a situação disparadora é construída. Estas manifestações estão contidas no cognitivo dos seres-vivos ditos emocionais, e os auxiliam na sobrevivência e na comunicação. Analiticamente as emoções podem caracterizadas como sendo um conjunto de indicadores rápidos de ação a ser tomada, que são resultantes de um processo de avaliação de um evento qualquer, utilizando memórias de experiências passadas, as emoções disparam a mais provavelmente correta ação.

Baseados no comportamento humano, existem dois modelos que se destacam na literatura, um deles baseia-se na psicologia cognitiva das emoções e é conhecido como modelo OCC, o outro chamado de modelo BDI leva em consideração três aspectos cognitivos para definir a ação do agente (ZAMBERLAM e GIRAFFA, 2001).

Em 1988 Ortony, Clore e Collins apresentam um modelo teórico para o surgimento das condições de disparo e a diferenciação das emoções. No livro "*The Cognitive Structure of Emotions*" é apresentado o modelo identificado na literatura como Modelo OCC. Categorizado por Scherer (2000) como um modelo léxico, ou seja, um modelo baseado na psicologia cognitiva das emoções.

Caracteriza-se o aspecto cognitivo do modelo OCC pela preocupação central em explicar o papel da cognição em avaliar uma condição de disparo sendo efetiva ou não, e na diferenciação das emoções. Seus autores desconsideraram aspectos fisiológicos e comportamentais que consideram como consequências dos estados emocionais. Este modelo define 26 classes de emoções. As emoções são divididas em três categorias que se distinguem pelo estímulo disparador: agentes, eventos e objetos. Agentes podem ser pessoas, animais e objetos inanimados. Eventos

podem ser vistos como a maneira de perceber as coisas que acontecem no ambiente, e objetos são todas as coisas vistas como inanimadas.

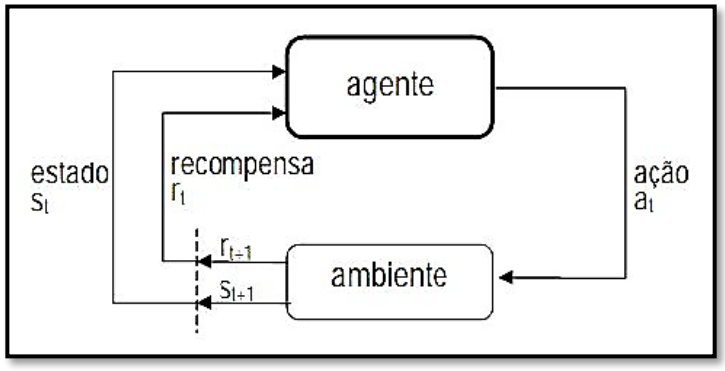
Outro modelo que tenta representar as emoções em modelos computacionais é o BDI, o qual caracteriza o agente como um sistema racional e representa a cognição emocional em três aspectos, crenças (*beliefs*), desejos (*desires*) e intenções (*intentions*) (ZAMBERLAM e GIRAFFA, 2001). Crenças podem ser vistas como um componente informativo no sistema, os quais são necessários para fornecer informações sobre o provável estado do ambiente. Os desejos definem os objetivos do agente. E por fim, as intenções fazem o papel de componente deliberativo do sistema. Servem para decidir o curso de ação que deve ser tomado pelo sistema (ZAMBERLAM e GIRAFFA, 2001).

2.5 APRENDIZAGEM POR REFORÇO

A aprendizagem por reforço é um subcampo da teoria de aprendizado de máquina a qual está contida nos campos de conhecimento da Inteligência Artificial. Este subcampo dedica-se no desenvolvimento de algoritmos e técnicas que permitem a uma máquina aumentar a probabilidade da correta tomada de decisão, e analogamente aperfeiçoar seu desempenho em alguma tarefa (CAMPONOGARA e SERRA, 2005).

Na prática a aprendizagem por reforço baseia-se no princípio da recompensa, de forma que através de um *feedback* do ambiente, o agente é recompensado positivamente quando obtém resultados considerados corretos. Por conseguinte, o agente deve tomar decisões baseando-se no passado e buscando sempre melhorar seus resultados (JÚNIOR, 2009). A Figura 6 apresenta o funcionamento do algoritmo de Aprendizagem por Reforço, onde a_t representa a ação executada no estado s_t, r_{t+1} a nova recompensa recebida em consequência do par estado-ação atual, e r_t e s_{t+1} representam a recompensa recebida e o novo estado alcançado.

Figura 6 - Aprendizagem por Reforço



Fonte: (CRUCIOL, 2012)

Através das pesquisas envolvendo Aprendizagem por Reforço, foram desenvolvidos algoritmos para aprimorar os resultados alcançados, dentre os quais, um destaca-se tanto pela facilidade de utilização quanto pelos resultados alcançados.

O algoritmo *Q-Learning* foi desenvolvido por Watking em 1989. Segundo (JÚNIOR, 2009), ele é considerado como um método de diferença temporal, de forma que a função ação-valor chamada de Q converge para valores ótimos de Q independentemente da política inicial usada. Uma política indica qual ação deve ser realizada em cada estado, constituindo assim, um mapeamento de estados para ações. O valor de um estado é definido como a soma das recompensas recebidas partindo-se do estado e seguindo-se uma determinada política até o estado final. Observa-se que o valor é, então, dependente da política.

Uma política ótima é o mapeamento de estados para ações que maximiza a soma das recompensas partindo-se de um estado inicial arbitrário e realizando-se ações até um estado final. Vale notar que pode-se querer minimizar a soma das recompensas. No entanto, nesta seção trata-se o caso mais comum (maximizar a soma), exceto quando explicitamente indicado.

A partir de uma inicialização com um valor arbitrário, definida pelo projetista, a função Q tende a convergir para valores ótimos a cada atualização da matriz dos $Q_{valores}$ do algoritmo. Desta forma, a cada ação do agente é enviada uma recompensa de grandeza relacionada ao quão assertivo este agente foi, em relação a ação esperada. Um novo estado é gerado pela atualização da matriz dos

$Q_{valores}$, e por meio do valor antigo de $Q(s_t, a_t)$ é feita uma correção com base na nova informação. O método de exploração *off-line* é dado pela seguinte equação:

$$Q(s,a) = Q(s,a) + \alpha \left[r(s_t, a_t) + \lambda \max_a Q_t(s_t, a_t) - Q(s,a) \right] \quad (1)$$

Sendo $r(s_t, a_t)$ a recompensa após a realização de uma ação a_t no estado s_t, α a taxa de aprendizagem, que determina o peso da informação nova em relação à antiga. Uma taxa de zero, fará com que o agente não aprenda nada, enquanto uma taxa de 1 faz com que o agente considere apenas a informação mais recente. Esta pode estar no intervalo de $0 < \alpha \leq 1$ podendo ser o mesmo valor para todos os pares de estados-ações. Lambda (λ) é o fator de desconto que determina a importância das recompensas futuras.

De maneira genérica o algoritmo *Q-Learning* pode ser estruturado conforme o Algoritmo 1:

Algoritmo 1 Q-Learning

Entrada: s' e r': sendo s' o estado atual e r' o sinal de recompensa
Saída: a: sendo a uma ação

1: repita
2: Inicializar s
3: repita
4: Selecionar uma ação a
5: Executar a ação a
6: Verificar a recompensa r
7: Verificar o novo estado T_{n+1}
8: $Q(s,a) = Q(s,a) + \alpha(r(s_t, a_t) + \lambda max Q_t(s_t, a_t) - Q(s,a))$
9: $s \leftarrow s_{t+1}$
10: até que s seja terminal
11: até que não for o fim dos episódios

O funcionamento do algoritmo acima é descrito pelos cinco passos a seguir (a partir de um dado instante de tempo T_n (CRUCIOL, 2012)):

1. O agente verifica o estado S_{Tn};
2. O agente seleciona uma ação A_{Tn};
3. O agente recebe uma determinada recompensa R;

4. O agente verifica a recompensa que recebeu em função do novo estado alcançado;
5. O agente atualiza o valor da ação que foi tomada.

3 APLICAÇÕES DE SISTEMAS MULTIAGENTES

Através das características de um sistema multiagentes explicadas na Seção 2.3, pode-se notar o potencial deste subcampo da Inteligência Artificial, e a flexibilidade de aplicação nas mais diversas áreas e problemas. Isso reflete como consequência a publicação de inúmeros trabalhos conceituando, formalizando protocolos, técnicas e modelos para aplicação deste tipo de abordagem.

Uma vantagem em utilizar uma arquitetura multiagentes, a decomposição da resolução de um problema em problemas menores, cada um resolvido por um agente. Por ser mais facilmente implementado, cada agente demanda menor capacidade computacional do que um único módulo ou software. E ainda se destaca que sua concepção visa reduzir a complexidade na implementação de um software e também possibilitar que abordagens biológicas e sociais sejam incorporadas em projetos de software.

Os exemplos apresentados neste capítulo demonstram alguns problemas que são tratados utilizando a abordagem do sistema multiagentes. Dentre eles destaca-se a competição de futebol de robôs, foco deste trabalho.

3.1 MONITORAMENTO DE TRÁFEGO URBANO

O congestionamento do tráfego urbano, é um problema característico das grandes cidades de todo o país. Ele acarreta diretamente, por exemplo, em poluição, acidentes de trânsito, insegurança e perda de produtividade e conforto social (SCHMITZ, 2002, p. 1). Na tentativa de resolver este problema, há sempre uma grande quantidade de recursos aplicados em serviços e orçamentos de projetos.

A utilização de semáforos para a coordenação do trânsito pode ser vista como muito mais que a sincronização da mudança de estados dos mesmos, o fluxo de tráfego, identificação de possíveis gargalos e a dinâmica do sistema. Torna-o um problema de difícil controle centralizado.

3.2 SEGURANÇA EM REDES

É sabido que a internet cresce em ritmo acelerado, este novo panorama traz consigo algumas necessidades e preocupações quanto á autenticidade, confidencialidade e integridade das informações.

De fato, tornou-se imprescindível a presença de mecanismos que visem dar apoio à segurança e agilidade no processo de tomada de decisão, no caso de ausência do administrador de segurança, em decorrência de que a maioria das intrusões ocorre fora do expediente normal das corporações.

O nível de sucesso do ataque depende obviamente da vulnerabilidade do sistema, este cenário mostra a necessidade do uso de um sistema de detecção de intrusos para auxiliar os administradores de segurança.

A vantagem de utilizar um sistema autônomo é estar em execução contínua com a mínima intervenção humana, resistir a subversão de forma a não se tornar um veículo de ataque, detecção de ataques explorando vulnerabilidades que não podem ser corrigidas, dentre outras (FERREIRA, 2002, p. 18).

3.3 MONITORAMENTO DO ESPAÇO AÉREO

O espaço aéreo vem crescendo em demanda cerca de 300% segunda a *Federal Aviation Administration (*FAA) nas próximas décadas.

Este aumento já pode ser visto, onde atrasos constantes e cancelamentos de voos tornam-se problemas cada vez mais frequentes. Entretanto acidentes aéreos são a maior motivação para aplicar um sistema autônomo de controle, uma vez que o serviço de gerenciamento executado no Brasil, por órgãos de Controle de Tráfego Aéreo, é caracterizado como uma tarefa complexa devido ao elevado número de variáveis associadas ao processo (ARRUDA, 2009, p. 1).

Mesmo com o apoio da alta tecnologia envolvidos neste processo, a experiência humana ainda é um fator determinante para a efetividade das medidas restritivas aplicadas. Ou seja, quanto maior a experiência do profissional responsável por este serviço, melhor a decisão tomada.

Com a crescente demanda do espaço aéreo, tem-se elevado consideravelmente a carga horária de trabalho destes profissionais gerando fadiga e até decisões ineficazes.

Para auxiliar os controladores de voos, um sistema multiagentes pode ser aplicado a fim de:

- Auxiliar no apoio à decisão referente à medidas restritivas aplicadas pelos centros de controle.
- Reduzir a carga de trabalho imposta aos controladores de voo.

3.4 GRUPO DE EMPILHADEIRAS EM DEPÓSITOS

O mau gerenciamento de estoque pode impactar nos lucros da empresa, dado que um estoque consome certa quantidade de capital de giro, necessitam de espaço físico, requerem manuseio e transporte e em alguns casos podem deteriorar-se.

Independentemente da posição do ciclo de operação sempre será gerado estoque uma vez que existe um desiquilíbrio no ritmo entre os fluxos de fornecimento e demanda (ORDOÑEZ, ANDRADE, *et al.*, 2009)

Um sistema multiagentes pode ser empregado a fim de distribuir os produtos de forma inteligente melhorando o fluxo de movimentação de produtos.

Na prática pode-se aplicar o sistema multiagentes a um grupo de empilhadeiras as quais serão responsáveis por realizar todas as tarefas necessárias a fim de tender a uma solução ótima na armazenagem e preparação de pedidos para distribuição.

Encontrar a solução ótima para estes problemas envolve avaliar um grande número de opções, podendo ser impossível avaliar todos os estados.

Por meio dos sistemas multiagentes busca-se desenvolver agentes dotados de uma percepção parcial do problema, capazes de avaliar seu entorno e de forma coletiva construir uma solução global.

3.5 ROBOCUP

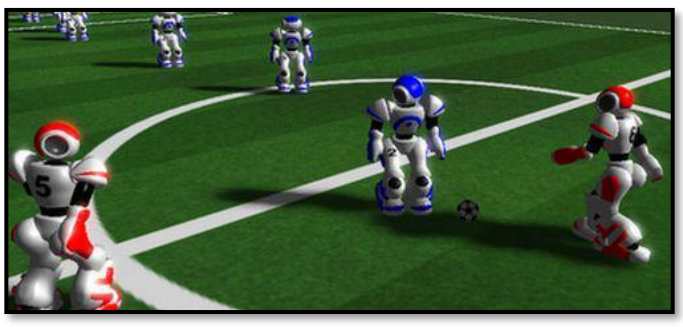
Sendo uma das mais conhecidas federações de incentivo ao desenvolvimento da robótica e inteligência artificial, a RoboCup Federation possui integrantes e equipes espalhados pelo mundo todo (RoboCup Brasil - Site Oficial, 2012). Eles participam de competições de futebol, simulações de resgate e na categoria chamada de RoboCup@Home a qual tem por objetivo geral o desenvolvimento da interação homem-máquina.

As competições de futebol em especial, necessitam de alto entrosamento, coordenação e capacidade de comunicação entre os agentes que compõem o time a fim de elaborar a melhor estratégia de jogo dentre as possíveis para cada situação. Pode ser observado que estas são algumas das características que um sistema multiagentes deve possuir, proporcionado uma ótima plataforma para o desenvolvimento deste trabalho.

Atualmente existem seis categorias definidas pela RoboCup para o futebol de robôs: *Simulation*, *Small Size*, *Middle Size*, Four-legged, *Humanoid* e a *E-league.* A categoria *Simulation* é destinada a simulação de futebol de robôs onde o ambiente de jogo é bidimensional ou tridimensional. A versão tridimensional do simulador, chamado de *SimSpark,* foi desenvolvido por iniciativa da RoboCup, como tentativa de promover a Inteligência Artificial e robótica inteligente, fornecendo um ambiente onde uma ampla gama de tecnologia pode ser integrada (SimSpark, 2012). A proposta de um simulador tridimensional veio da ideia de realizar simulações cada ver mais realistas (ROBOCUP, 2016).

A grande vantagem desta categoria é que, as equipes não precisam se preocupar com a aquisição de robôs fazendo com que os custos de participação sejam reduzidos. A Figura 7 ilustra uma cena extraída do simulador SimSpark.

Figura 7 - Ambiente de Simulação SimSpark



Fonte: http://www.cs.utexas.edu/~patmac/cs344m/

Na categoria *SmallSize* os robôs não podem ultrapassar 18 centímetros de diâmetro, as jogadas são processadas por um computador principal a partir da posição dos robôs e a posição da bola. A Figura 8 apresenta um exemplo de robô desta categoria.

Figura 8 - Robô da categoria *SmallSize.*

Fonte: http://www.cs.cmu.edu/~robosoccer/image-gallery/index.html

Já na categoria *MiddleSize* os robôs possuem até 45 centímetros de diâmetro. Diferentemente da *SmallSize*, eles são autônomos, comunicam-se entre si via *wireless* e obtém todas as informações necessárias para o jogo através de seus sensores. A Figura 9 apresenta um protótipo de robô da categoria *MiddleSize*.

Figura 9 - Robô da categoria *MiddleSize.*

Fonte: http://visionandrobotics.nl/2014/07/22/asml-net-niet-op-tijd-klaar-voor-brazilie/

As categorias *Four-Legged* e *Humanoid* utilizam modelos de robôs quadrúpedes e bípedes respectivamente. Na *Four-Legged é* utilizado como jogador

o modelo AIBO (Figura 10) desenvolvido pela Sony®. Ele é visto como uma plataforma de baixo custo utilizado nesta categoria desde 1999.

Figura 10 - Sony AIBO categoria *FourLegged.*

Fonte: http://www.jfdubeau.com/babbling-eloquently/2015/6/19/i-want-my-robot-pet

Por uma progressão natural a categoria *Four-Legged* vem sendo ofuscada pela categoria *Humanoid,* a qual utiliza robôs bípedes desenvolvidos pelos próprios participantes ou o robô padrão NAO (Figura 11) que é desenvolvido pela Aldebaran®. Este robô possui 25 graus de liberdade e proporciona uma maior gama de possibilidades de movimentação, aproximando-se dos movimentos utilizados no futebol real, o qual sempre foi um dos desejos da RoboCup (RoboCup Brasil - Site Oficial, 2012).

Figura 11 - Aldebaran NAO categoria *Humanoid.*

Fonte: http://robotics.wikia.com/wiki/NAO

3.5.1 Futebol de Robôs Tridimensional

A simulação de futebol de robôs tridimensional é realizada com base em uma adaptação das regras do futebol convencional, para atender as limitações do agente simulado e do ambiente de simulação.

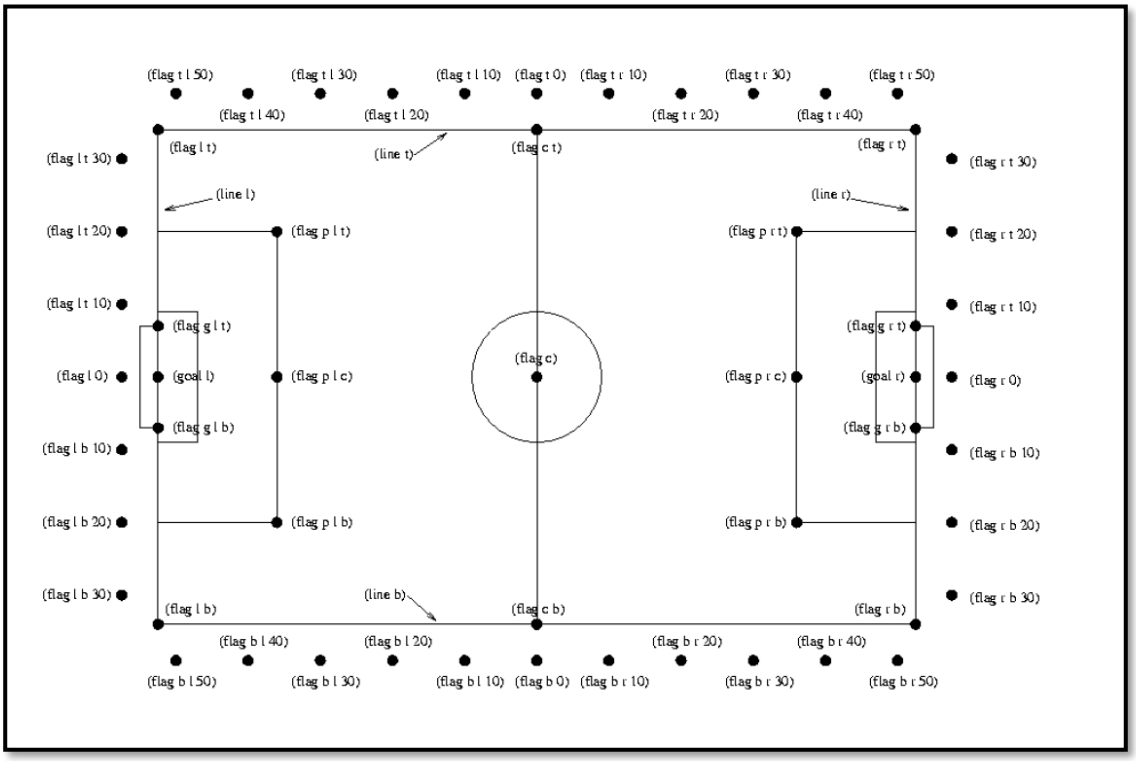
O ambiente de simulação (campo de futebol) foi redimensionado para atender as proporções necessárias. As dimensões do campo de futebol simulado são de 30 metros de comprimento por 20 metros de largura, enquanto um campo de futebol oficial é de aproximadamente 110 metros de comprimento por 70 metros de largura. Sendo assim, o campo simulado tem cerca de 35% das dimensões de um campo de tamanho oficial (ROBOCUP, 2016).

As goleiras, a grande área, o círculo central e a bola tiveram suas medidas alteradas conforme as necessidades; Cada goleira mede 2,1 metros de largura por 0,8 metros de altura, a grande área tem dimensões de 3,9 metros de largura por 1,8 metros de comprimento, o círculo central possui 2 metros de diâmetro, e a bola utilizada tem um raio de apenas 0,04m e 26 gramas de massa (ROBOCUP, 2016).

Os jogos de futebol robótico simulado possuem dois tempos de 5 minutos de duração que corresponde a 3000 ciclos de simulação cada. Para casos em que é necessário desempatar o jogo, um tempo adicional de 5 minutos (3000 ciclos de simulação) é aplicado, no qual a primeira equipe que marcar um gol é a vencedora.

A localização geográfica dos agentes no campo de futebol é auxiliada através de marcadores distribuídos pelo campo. Cada marcador possui as coordenadas conhecidas pelos agentes que, com as informações passadas para eles durante a comunicação servidor-jogador, são capazes de se orientar. Para a localização das goleiras utiliza-se marcadores de posição localizados no centro do gol e nos cantos do travessão a 0,8m acima do solo. A Figura 12 ilustra em detalhes a posição de cada marcador do campo de futebol.

Figura 12 – Localização e nomes dos marcadores de orientação no *Soccer Server*

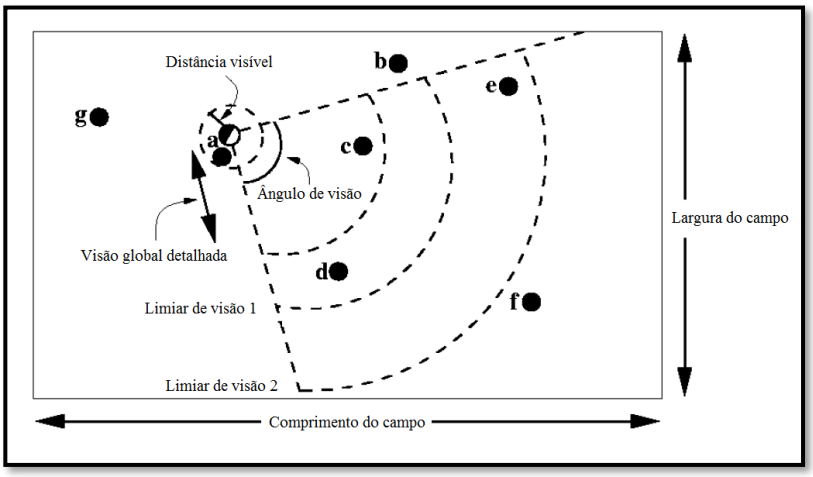


Fonte: http://simspark.sourceforge.net/wiki/index.php/Soccer_Simulation

A percepção dos agentes sobre o campo é subjetiva e baseada em três protocolos de comunicação jogador-servidor: ouvir, ver e sentir. O protocolo ouvir destina-se a comunicar o agente em a relação às mensagens transmitidas pelo árbitro, pelos colegas de time, pelo treinador e pelos adversários. As informações visuais transmitidas via protocolo desempenham um papel primordial no futebol robótico simulado. Este protocolo permite aos agentes obter informações de direção e distância dos jogadores e objetos que entraram em seu campo de visão e também perceber a bola ou outros jogadores quando os mesmos se encontram bem próximos ao jogador (Figura 13). É importante destacar que a percepção de visão não informa ao jogador sua posição global, a posição global de outros jogadores, ou da bola, mas sim uma posição relativa dos agentes e objetos que se encontram em seu campo de visão, necessitando do auxílio dos marcadores encontrados no campo e a realização de conversão de posição relativa em posição absoluta. A quantidade de informações enviadas dos objetos para o agente, depende ainda do seu cone de visão, da qualidade da imagem e da distância relativa entre o jogador e os objetos, ou jogadores. A Figura 13 ilustra o campo de visão de um jogador. Nela

pode ser visto três linhas pontilhadas onde na primeira descrita como “Visão global detalhada”, é enviado ao jogador todas as características dos jogadores contidos em seu campo de visão (equipe, número e direção do corpo e direção da cabeça). Após o primeiro limiar, informações como número do jogador, direção do corpo e da cabeça não são enviadas no protocolo. E após o segundo limiar os jogadores são identificados como anônimos sem informação de seu número ou equipe que pertence. A bola possui situação similar subtraindo a informação de direção relativa e velocidade relativa após o “primeiro limiar de visão”, mantendo apenas a distância e direção do jogador até a bola. Os círculos g e b que não estão contidos no protocolo de percepção visual deste campo de visão. Os objetos estáticos (bandeiras linhas e balizas) não entram nestas regras, pois são necessários para que o jogador consiga calcular as posições absolutas necessárias, a partir das informações recebidas pelo protocolo de percepção da visão.

Figura 13 - Limitações do campo de visão



Fonte: (Análise do Domínio de Aplicação: Futebol Robótico, 2016)

A última percepção sentir refere-se às informações físicas correspondentes ao estado do agente. Nela o agente recebe os parâmetros de *stamina* (grandeza utilizada para realizar um movimento), velocidade (valor absoluto e vetorial), ângulo da cabeça, quantidade de arrancadas (refere-se ao ato de acelerar o corpo do jogador), quantidade de giros (refere-se ao ato de rotacionar o corpo do jogador), quantidade de falas (mensagens enviadas pelo jogador), quantidade de rotações do pescoço, quantidade de movimentos, quantidade de mudanças do campo de

visão e para o goleiro quantidade de comando pegar executados na tentativa de agarrar a bola.

Os jogadores recebem atualizações das duas percepções um determinado período, que pode variar de 37,5 milissegundos (campo de visão pequeno e qualidade baixa) à 300 milissegundos (campo de visão amplo e qualidade alta), de acordo com o a qualidade de visão e o campo de visão escolhidos naquele instante.

Cada jogador deve se comunicar com o servidor através de dois protocolos (ação e percepção), enviando informações referentes as percepções e ações que deseja executar. Estes dois protocolos não preveem qualquer resposta além da mensagem de erro, no caso de algum parâmetro ser ilegal ou se for enviado um comando desconhecido (Análise do Domínio de Aplicação: Futebol Robótico, 2016).

3.5.2 O Árbitro Virtual

Para conduzir o jogo de futebol de robôs simulado, existem dois árbitros, um arbitro automático (agente de monitoramento) e um arbitro humano que acompanha o jogo através de um monitor. O árbitro automático consegue arbitrar o jogo em geral sem a necessidade de intervenção do árbitro humano. Ele limita automaticamente cada tempo de jogo, mantém ainda as informações do último jogador a tocar na bola e verifica se a bola entrou na pequena área, ou se foi chutado para a lateral. Ele também é capaz de detectar e marcar gols, saídas de bola, apitar escanteio ou um tiro de meta para a equipe correta. Durante uma cobrança de uma falta, ele monitora se a equipe oponente mantém uma distância mínima de 1,3 metros, bem como 1,0 metros no caso de um tiro de meta, evita colisões em massa de robôs em torno da bola, bem como robôs caídos ao redor do campo bloqueando o jogo. Além disso cuida para que nenhuma equipe bloqueie a própria goleira com mais de uma certa quantidade de jogadores (Análise do Domínio de Aplicação: Futebol Robótico, 2016).

No entanto, faltas do tipo anti-jogo são difíceis de serem julgadas pelo árbitro artificial, levando em consideração que o mesmo não possui uma percepção das

intenções dos jogadores. Para estas e outras situações é necessária a intervenção do árbitro humano.

3.5.3 O Árbitro Humano

O árbitro humano monitora o jogo através do *soccer monitor,* uma interface que permite ao árbitro humano intervir no jogo. Ele é responsável por apitar o início de jogo, detectar faltas e outras atitudes que necessitem de interpretação da jogada, comandar cobranças de faltas a curta distância (onde o agente pode chutar diretamente para o gol), e por resolver situações em que o jogo fica travado. Estas situações são provocadas se por exemplo nenhum jogador consegue alcançar a bola. Nesses casos o árbitro humano pode manipular a bola, colocando-a em um lugar de preferência (Análise do Domínio de Aplicação: Futebol Robótico, 2016).

3.5.4 Principais Regras Específicas do *Soccer Simulation*

1. Gols marcados diretamente após o toque inicial de jogo não são aceitos, a bola tem de sair o círculo central antes do jogador realizar um chute a gol.

2. Apenas o jogador designado à posição de goleiro pode jogar no gol.

3. Manusear a bola com a mão ou o braço através de um ato deliberado de contato ou tomada de bola de um jogador, deve envolver a decisão do árbitro em marca uma falta. O goleiro pode não ser culpado de uma falta de manuseio com as mãos quando suas mãos dentro de sua própria área.

4. Agarrar um adversário inclui o ato de impedir seu movimento. Impedir a progressão de um adversário significa se mover deliberadamente para a trajetória do adversário para obstrui-lo, retardá-lo ou forçar sua mudança de direção. Detenção ou impedir a progressão de um adversário é penalizado com um livre para a equipe adversária

5. Obstruir a bola envolve o uso de corpo, braços ou pernas de um jogador para impedir o progresso do jogo, que inclui a cobertura da bola, a bola realização, segurando a bola entre os braços ou pernas e deitado na frente do gol em um ataque situação. Deliberadamente obstruir a bola, como julgado pelo árbitro, por mais de 10 segundos é penalizado com um livre para a equipe adversária. Obstrução não intencional de a bola por mais de 10 segundos é encerrado por uma bola queda pelo árbitro.

6. O goleiro é reposicionado se o mesmo se mantiver imóvel após 30 segundos, ou se tentar e não conseguir ficar de pé em 60 segundos; Os outros jogadores serão reposicionados após 15 segundos de imobilidade, ou depois de não conseguir ficar de pé em 30 segundos.

7. Durante pênaltis, cada equipe tem apenas um único jogador em campo: o goleiro para a equipe de defesa e um atacante para a equipe atacante. A equipe atacante recebe 40 segundos para cobrar o pênalti. O goleiro da equipe defensora tem que ficar dentro da sua grande área durante este período. Jogos empatados são decididos em 5 pênaltis para cada equipe.

3.6 FIRA

A *Federation of Internacional Robot-soccer Association*, assim como a *RoboCup* é uma federação fundada através da ideia de popularizar e desenvolver a robótica. Foi idealizada pelo Professor Jong-Hwan Kim em 1995 e oficializada em 1997.

A FIRA realiza grandes eventos como a FIRA Robot World Cup, nestes a eventos são explorados o desenvolvimento de muitos problemas voltados à área de pesquisa (Sistemas Multiagentes e Robótica). Desde seu início até hoje a FIRA já visitou muitos países, incluindo Austrália, Brasil, China, França e Coréia, tornando-se reconhecida mundialmente.

Como a RoboCup a Fira explora problemas de dinâmica complexa, fazendo do futebol de robôs uma plataforma ideal. No futebol de robôs a FIRA organiza-se através das categorias *AndroSot*, *AmireSot, RoboSot, NaroSot, MiroSot, HuroCup* e *SimuroSot* esta última faz parte do futebol de robôs simulado.

A categoria *AndroSot* é disputada por andróides controlados remotamente por um computador. Os robôs desta categoria podem pesar até 600 gramas e ter no máximo 50cm de altura.

Atualmente nesta categoria cada time é composto por 5 jogadores e cada partida é disputada em um campo de 220 cm de comprimento e 180 cm de largura. Na figura 14 pode ser visto os robôs que disputam esta categoria.

Figura 14 - Ilustração do início de uma partida da categoria *AndroSot*

Fonte: http://www.fira.net/contents/sub03/sub03_5.asp

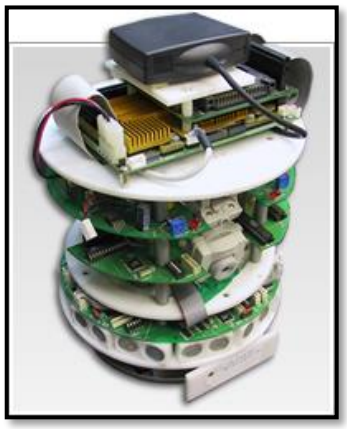
A categoria *AmireSot* é disputada por robôs totalmente autônomos, com sistema de visão acoplado. O jogo utiliza uma bola de tênis amarela e o campo tem dimensões de 130cm de comprimento por 90 cm de largura. A Figura 15 ilustra um robô utilizado nesta categoria.

Figura 15 - Robô utilizado na categoria AmireSot

Fonte: http://www.fira.net/contents/sub03/sub03_2.asp

Na categoria *RoboSot* se disputa partidas com times de três jogadores robôs parcialmente ou completamente autônomos. No caso de serem parcialmente autônomos, uma estação de computador pode ser utilizada para o processamento de informações provenientes das câmeras dos robôs. A dimensão do robô desta categoria é de base de tamanho 20 cm x 20 cm x sem limite para altura, a bola utilizada é uma bola de tênis amarela ou verde, e dimensões do campo é de 260 cm X 220 cm. Na Figura 16 pode ser visto um exemplo do robô utilizado na categoria *RoboSot.*

Figura 16 - Exemplo de robô utilizado na categoria *RoboSot*



Fonte: http://www.fira.net/contents/sub03/sub03_6.asp

A *HuroCup* é uma categoria é disputada por robôs humanoides, com tamanho máximo de 150 cm e peso máximo de 30 Kg. Esta categoria tem como regra a utilização de robôs bípedes que podem ser remotamente controlados ou autônomos.

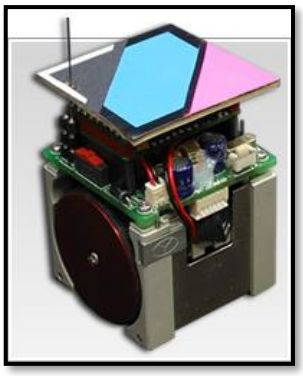
A Figura 17 ilustra um dos tipos de robô humanoide em jogo pela categoria *HuroCup*.

Figura 17 - Robôs humanoides disputando a partida pela categoria *HuroCup*

Fonte: http://www.fira.net/contents/sub03/sub03_1.asp

Na categoria *NaroSot* uma partida é disputada por dois times de 5 jogadores cada. Uma estação de computador é usada em cada time e dedica-se principalmente para o processo de visão computacional para identificar a localização dos jogadores. O tamanho dos robôs desta categoria é limitado em 4 cm x 4 cm x 5,5 cm, largura, comprimento e altura respectivamente. Esta categoria utiliza uma bola de ping-pong laranja e um campo com dimensões de 130 cm de comprimento por 90 cm de largura. Na Figura 18 é visto um exemplo de robô utilizado para disputar uma partida da *NaroSot*.

Figura 18 - Exemplo de robo da categoria *NaroSot*

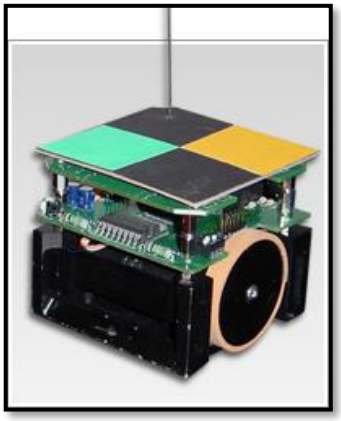


Fonte: http://www.fira.net/contents/sub03/sub03_4.asp

Com estrutura similar aos robôs da categoria *NaroSot,* a categoria *MiroSot* se diferencia nas dimensões do robô utilizado no tipo de bola e nas dimensões do campo. O tamanho de cada robô é de 7,5 cm de largura 7,5 cm de comprimento e

7,5 cm de altura. A bola utilizada é uma bola de golfe laranja e o campo tem dimensões de 150 cm de comprimento por 130 cm de largura para a subcategoria *Small League* e 220 cm de comprimento 180 cm de largura para a subcategoria *Middle League*. A Figura 19 ilustra um robô utilizado para disputar as partidas da *MiroSot*.

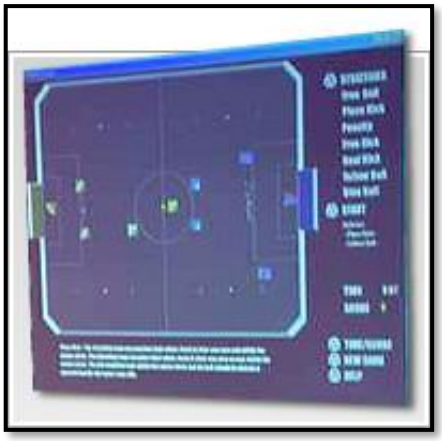
Figura 19 - Exemplo de robô da categoria *MiroSot*



Fonte: http://www.fira.net/contents/sub03/sub03_3.asp

Por fim, a categoria *SimuroSot* consiste na simulação da categoria *MiroSot,* nela o custo para participação de equipes é reduzido e estratégias são desenvolvidas para controlar os agentes que compõem os times. Na Figura 20 é visto a interface do simulador Robot Soccer v1.5a utilizado pela categoria *SimuroSot*.

Figura 20 - Interface do simulador utilizado na categoria *SimuroSot*



Fonte: http://www.fira.net/contents/sub03/sub03_7.asp

4 IMPLEMENTAÇÃO

Para desenvolvimento da proposta deste trabalho foi necessário realizar uma avaliação dos ambientes de simulação de futebol de robôs utilizados para nas competições das federações FIRA e da ROBOCUP. Ambas, disponibilizam o software (FIRA) ou o código fonte (ROBOCUP). Porém em ambos os casos adicionar novas funcionalidades não é uma tarefa trivial, pois seria necessário um estudo mais aprofundado do código-fonte, o que não entra no escopo deste trabalho. Sendo assim, optou-se por desenvolver um ambiente de simulação pois sua arquitetura poderia ser moldada para as necessidades deste trabalho com a possibilidade de adicionar novas funcionalidades como arquivos de logs, base de dados, entre outras coisas, auxiliando na análise do comportamento de um agente inteligente, jogadas, ações executadas e progressão da aprendizagem.

O sistema desenvolvido neste trabalho tem como objetivo ser um ambiente de simulação de futebol de robôs de código aberto e arquitetura flexível, com aplicação geral no desenvolvimento de agentes inteligentes e sistemas multiagentes. O ambiente desenvolvido contém os elementos fundamentais para dar suporte a um jogo de futebol de robôs e aprendizagem de máquina, tais como:

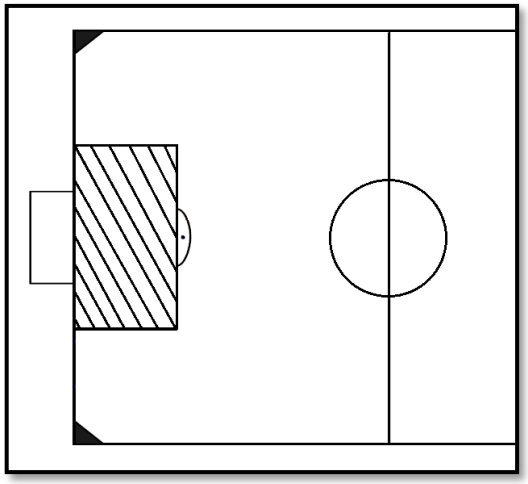
1. Motor de Física
2. Biblioteca de manipulação gráfica
3. Interface de controle
4. Sistema de log
5. Base de dados

O ambiente de simulação de futebol de robôs foi desenvolvido em linguagem de programação C#. A escolha da linguagem se justifica por oferecer uma plataforma de desenvolvimento de software robusta, segura e com amplos recursos matemáticos. A linguagem C# é utilizada em projetos de software de várias naturezas, provendo um código otimizado e com possibilidades de recursos que atendiam os requisitos do sistema a ser desenvolvido. Além disso, havia o conhecimento prévio na linguagem.

4.1 CARACTERÍSTICAS DO AMBIENTE DE SIMULAÇÃO

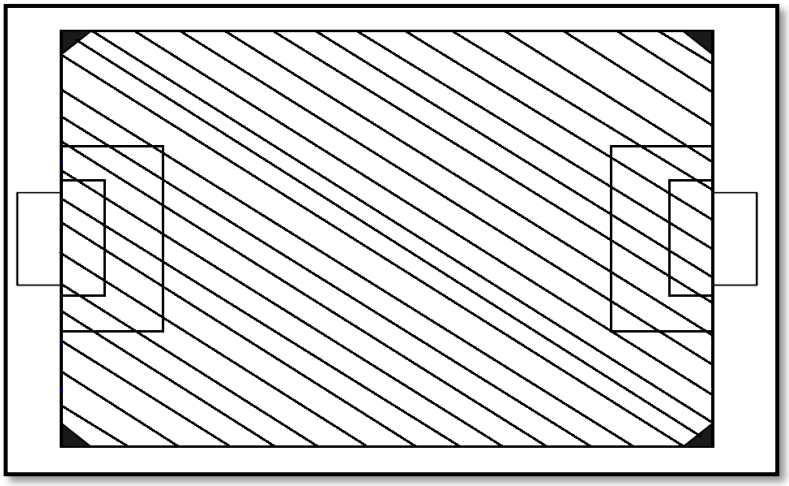
Para aplicar os mecanismos de aprendizagem por reforço e motivação no jogo de futebol de robôs simulado, propõe-se que o time de agentes seja composto de dois jogadores. Eles irão executar duas funções básicas de um jogo de futebol o ataque e a defesa. A defesa será composta pelo goleiro, que terá um espaço definido de trabalho delimitado pela grande área. O atacante irá executar sua função característica e poderá mover-se livremente pelo por todo o ambiente (campo). As Figuras 21 e 22 ilustram as regiões que delimitam a área de atuação do goleiro e do atacante.

Figura 21 - Região de atuação do goleiro (hachurada)



Fonte: Autor

Figura 22 - Região de atuação do atacante (hachurada)



Fonte: Autor

4.1.1 Estratégia de Jogo

A estratégia de jogo será desenvolvida adotando uma metodologia dividida em quatro etapas:

1. Definir e discretizar as ações dos agentes (jogadores);
2. Definir e discretizar os estados do ambiente no qual os agentes estão inseridos;
3. Definir os valores de recompensa da tabela $r(s, a)$, para cada par estado (s) X ação (a);
4. Definir o algoritmo de aprendizagem, relacionado a ação (a) executada no estado (s).

As ações de um agente no campo de futebol de robôs simulado foram definidas a partir dos fundamentos básicos da função de um jogador em um jogo de futebol real. As principais ações escolhidas para executar as funções de goleiro em uma partida de futebol são:

1. Defesa;
2. Posicionamento;
3. Reposição;

A ação de defesa é primordial para a função de goleiro e deve ser executada sempre que o adversário entrar na grande área com o domínio da bola. A ação posicionamento, faz com que o goleiro se posicione logo à frente da linha do gol, no centro da pequena área. A ação descrita neste trabalho como deslocamento refere-se à ação de chutar a bola para fora da grande área quando a mesma vem em direção ao gol sem domínio do adversário ou do próprio time. Por fim, a ação reposição, faz referência ao ato do goleiro repor a bola em jogo para o atacante de seu time, caracterizada como um toque de bola.

Para o atacante, as principais ações escolhidas para caracterizar sua função foram:

1. Chute;

2. Drible;
3. Finta;
4. Desarme;
5. Domínio da bola.

Sempre que uma oportunidade de gol surgir o atacante deve executar a ação chute comportando-se o mais ofensivamente possível. A ação domínio da bola deve ser executada sempre que o atacante desejar deslocá-la de um ponto ao outro do campo (correr em direção ao gol). O drible deve ser executado sem que o atacante possuir a bola e o adversário esteja próximo. O desarme é necessário quando o adversário possui a posse da bola.

Os estados são interpretados como a interação entre os agentes. Foram definidas as características mais relevantes durante uma partida de futebol relacionada às funções goleiro e atacante.

Para o goleiro os estados definidos como relevantes foram:

1. Adversário fora da grande área e bola na direção do gol adversário;
2. Bola e adversário dentro da grande área;
3. Bola dentro da grande área e adversário fora da grande área.

Os estados definidos como relevantes para a função atacante foram:

1. Adversário com a posse da bola;
2. Atacante com a posse da bola e adversário atrás ou fora da trajetória atual do atacante;
3. Atacante com a posse da bola e adversário perto, distância igual ou inferior a três robôs (80 pixels);
4. Atacante com a posse da bola e adversário longe, distância superior a três robôs (80 pixels);
5. Nenhum jogador com a posse de bola.

4.1.2 Matriz de Recompensas

O jogo de futebol de robôs envolve uma grande complexidade em relação ao número de ações para que o time alcance o objetivo global (marcar um gol). O método de aprendizagem por reforço oferece recompensas quando o agente (jogador) escolhe uma ação considerada relevante para alcançar seu objetivo. Por definição, foi escolhido cinco valores referentes à relevância da execução de uma ação a um determinado estado:

- -10 indica que o par estado/ação não é indicado para ser executado;
- 0 indica que o par estado/ação tem muito pouca relevância para ser executado;
- 5 indica que o par estado/ação tem pouca relevância para ser executado;
- 10 indica que o par estado/ação tem média relevância para ser executado;
- 20 indica que o par estado/ação tem alta relevância para ser executado.

Com base nas definições utilizadas, as matrizes de recompensa $r(s, a)$ para a função goleiro e atacante são apresentadas pela tabela 1 e tabela 2 respectivamente, em que as linhas representam os estados do ambiente e as colunas as ações que os agentes podem executar.

Tabela 1 - Matriz de recompensa do goleiro

Estado/Ação	Defender	Posicionar	Reposição
Estado 1	-10	20	-10
Estado 2	20	-10	5
Estado 3	10	0	20

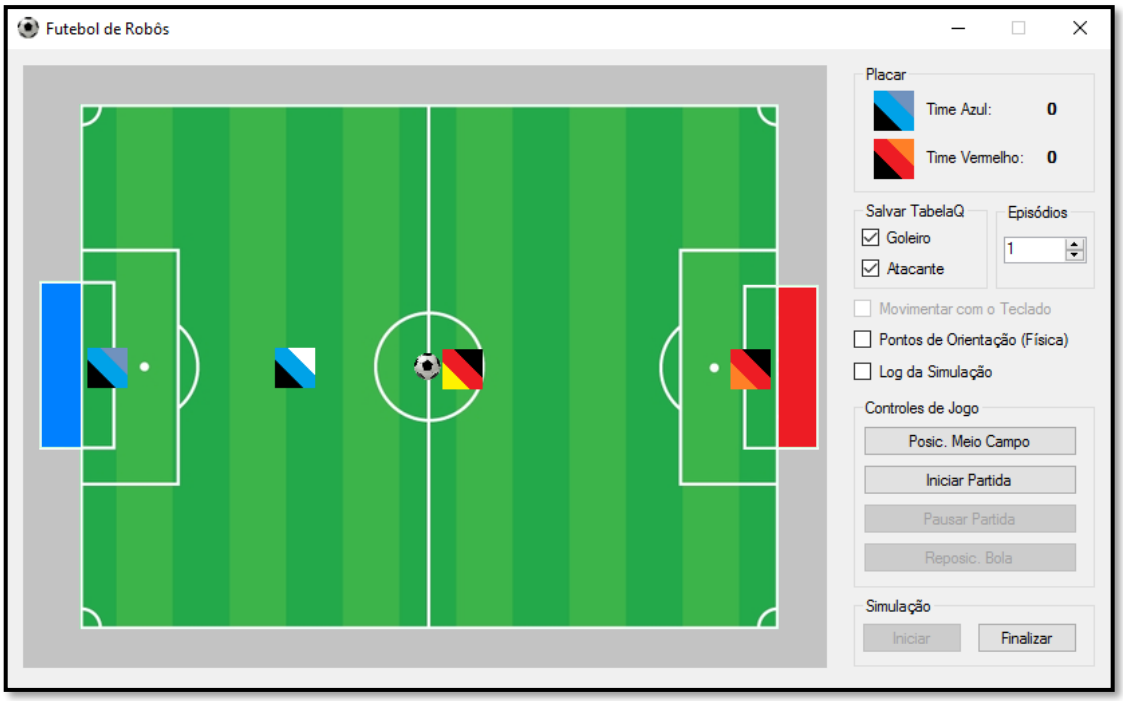
Tabela 2 - Matriz de recompensas do atacante

Estado/Ação	Chutar	Drible	Domínio	Desarme	Finta
Estado 1	-10	-10	-10	20	0
Estado 2	20	0	-10	-10	-10
Estado 3	0	20	0	-10	0
Estado 4	5	0	20	-10	0
Estado 5	-10	-10	-10	-10	20

4.2 INTERFACE DO AMBIENTE

A interface do ambiente de simulação foi pensada para ser simples e intuitiva. Nela o usuário conta com a região de simulação, representado graficamente por toda a extensão da imagem do campo de futebol, um placar do jogo no canto superior direito, as configurações da simulação no centro do painel de controle e os botões de comando, responsáveis por inicializar o simulador, iniciar e pausar o jogo, dentre outras opções. A Figura 23 ilustra a interface principal do ambiente de simulação. Percebe-se na parte principal da tela o desenho do campo de futebol. Na região da direita o usuário pode visualizar o placar e selecionar parâmetros e configurações.

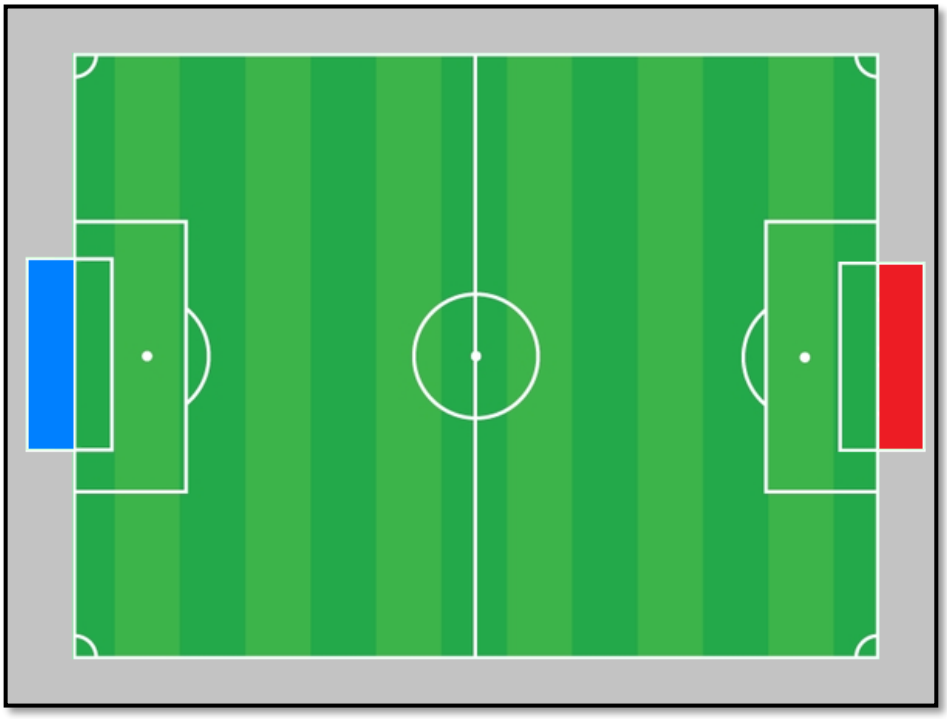
Figura 23 - Visão geral do ambiente de simulação



Fonte: Autor

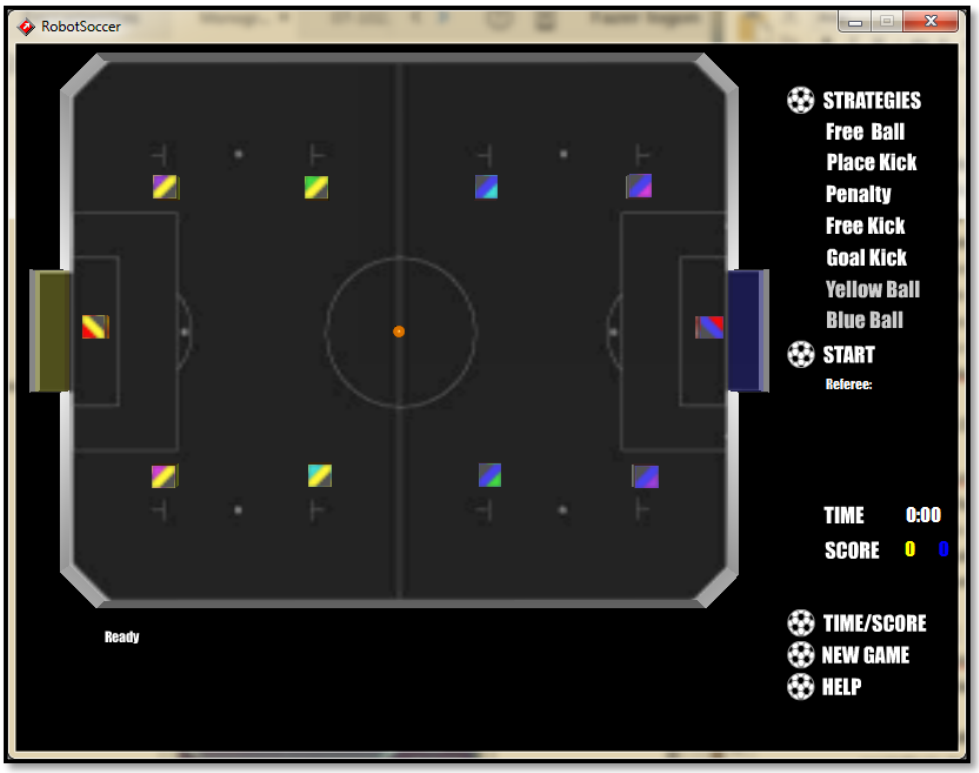
Uma imagem estática foi utilizada para representar o campo de futebol (Figura 24). O deslocamento dos jogadores no campo é delimitado pela borda cinza da imagem. Este comportamento visa imitar o que ocorre na competição de simulação de futebol de robôs da federação FIRA (categoria Simurosot).

Figura 24 - Campo de futebol



Fonte: Autor

Figura 25 - Ambiente de simulação da categoria (Simurosot)

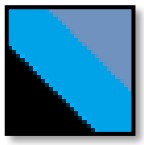


Fonte: http://www.fira.net/

Os jogadores são representados por imagens de formato quadrado, com três faixas coloridas dispostas transversalmente que auxiliam na identificação da direção

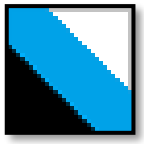
do agente, do seu time e da função executada em campo (Figuras 26, 27, 28 e 29). Todos os jogadores possuem uma área destacada na cor preta sinalizando a regiões traseira e direita dos jogadores. A faixa central indica o time do jogador (azul ou vermelho), e a região situada no canto superior direito diferencia os jogadores do mesmo time com cores definidas arbitrariamente.

Figura 26 - Goleiro do time azul



Fonte: Autor

Figura 27 - Atacante do time azul



Fonte: Autor

Figura 28 - Goleiro do time vermelho

Fonte: Autor

Figura 29 - Atacante do time vermelho

Fonte: Autor

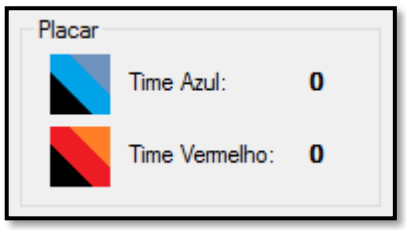
A bola é representada por uma imagem composta de hexágonos pretos e brancos dispostos de forma a representar uma superfície esférica (Figura 30).

Figura 30 - Imagem da bola de futebol

Fonte: Autor

Um dos indicadores de desempenho mais importante em uma partida de futebol é o placar do jogo. A partir dele, sem conhecimentos prévios ou análises das equipes, é possível identificar qual equipe possui o melhor desempenho em campo. Para dar destaque, o placar do ambiente de simulação está situado no canto superior direito, com a imagem dos respectivos goleiros do time azul e vermelho, o nome da equipe, e ao lado o campo numérico que contabiliza os gols feitos pelas equipes (Figura 31).

Figura 31 - Placar

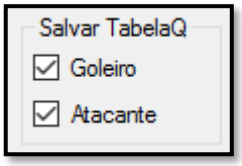


Fonte: Autor

Os times são pré-definidos, sendo que o time azul incorpora o uso da aprendizagem de máquina (método Q-Learning) com regras mais elaboradas para definir os seus objetivos, uma vez que sua estrutura se aproxima mais de um agente cognitivo do que de um agente puramente reativo. Apenas para o time azul, as opções do menu ilustrados pelas Figuras 32 e 33 surtirão efeitos. O time vermelho não incorpora este mesmo algoritmo; o seu comportamento é baseado em percepção ação.

Nas configurações da simulação referentes ao Q-Learning, foram adicionadas ao menu as opções de salvar os dados da tabela Q (Figura 32). A tabela Q vai sendo atualizada a cada iteração durante a simulação. Com esta opção, os dados são salvos no diretório de compilação do programa e carregados em uma próxima execução do programa, para melhorar continuamente o aprendizado dos jogadores do time azul.

Figura 32 - Salvar dados da tabela Q



Fonte: Autor

Outro parâmetro que pode ser ajustado é a quantidade de episódios desejados para execução do Q Learning (Figura 33). Um episódio é uma sequência de ações até que ocorra a realização de um gol por um dos times.

Figura 33 - Número de episódios

Fonte: Autor

O motor de física, necessita de “pontos de orientação”, que são referências em termos de coordenadas para funcionar. Estes pontos de orientação são utilizados para realizar os cálculos e podem ser visualizados se a opção “Pontos de Orientação (Física) ” (Figura 34) estiver marcada. Através da Figura 35 pode ser observado, destacado em vermelho, a opção marcada no painel de configurações, e seus resultados na área de simulação, onde em zoom todas as referências utilizadas pelo motor de Física estão destacadas.

Figura 34 - Mostrar pontos de orientação

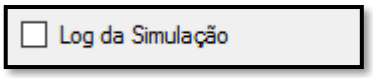
Fonte: Autor

Figura 35 - Pontos de orientação (Física)

Fonte: Autor

Para auxiliar na análise do comportamento dos agentes do time azul, a opção “Log da Simulação” (Figura 36), quando marcada, salva na pasta de compilação do projeto um arquivo no formato CSV (*Comma-separated values*). Este arquivo vai conter as informações dos estados (a cada instante de tempo), a ação escolhida e a recompensa agregada ao par estado-ação.

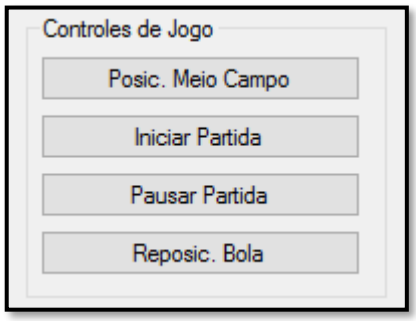
Figura 36 - Salvar log da simulação



Fonte: Autor

Para o controle do jogo, um grupo de quatro botões executam funções específicas descritas abaixo. A Figura 37 ilustra estes componentes:

Figura 37 - Controles de jogo



Fonte: Autor

4.3 COMPONENTES GRÁFICOS

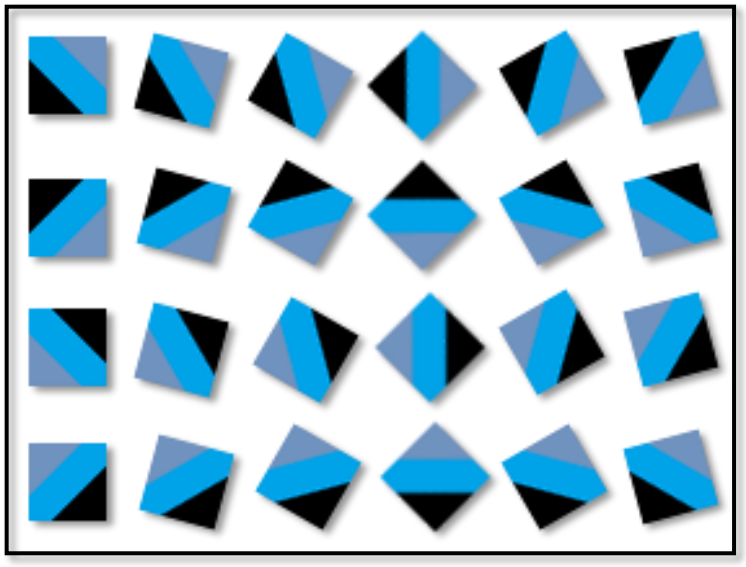
Existem algumas limitações quando se cria um projeto do tipo “*Windows Form*” na linguagem de programação C#, dentre elas a dificuldade de realizar movimentos ou animações com os componentes padrões, salientando a razão pela qual a técnica de animação bidimensional *sprites* foi utilizada neste trabalho.

Em computação gráfica um *sprite* é definido como um objeto gráfico estático que representa algum objeto, personagem ou cenário em um jogo. A técnica de animação gráfica 2D s*prites* que consiste em carregar uma determinada sequência de imagens na memória e realizar um “*grab*” (buscar uma parte da imagem), gerando uma sequência de animação.

Para aplicar esta técnica um componente foi programado um componente canvas para cada agente, o qual trabalha em conjunto com seu respectivo *sprite*, criado a partir da discretização de uma representação da rotação do “corpo” do agente em vinte e quatro quadros que são mostrados de forma a representar uma transição suave de direção. Foi utilizada uma biblioteca livre que realiza algumas operações básicas de manipulação de *sprite* (DJUROVIC), esta foi alterada conforme foi necessário.

A Figura 38, ilustra o movimento de rotação de um jogador, discretizado em uma sequência de 24 *sprites.*

Figura 38 - *Sprites* do goleiro do time azul



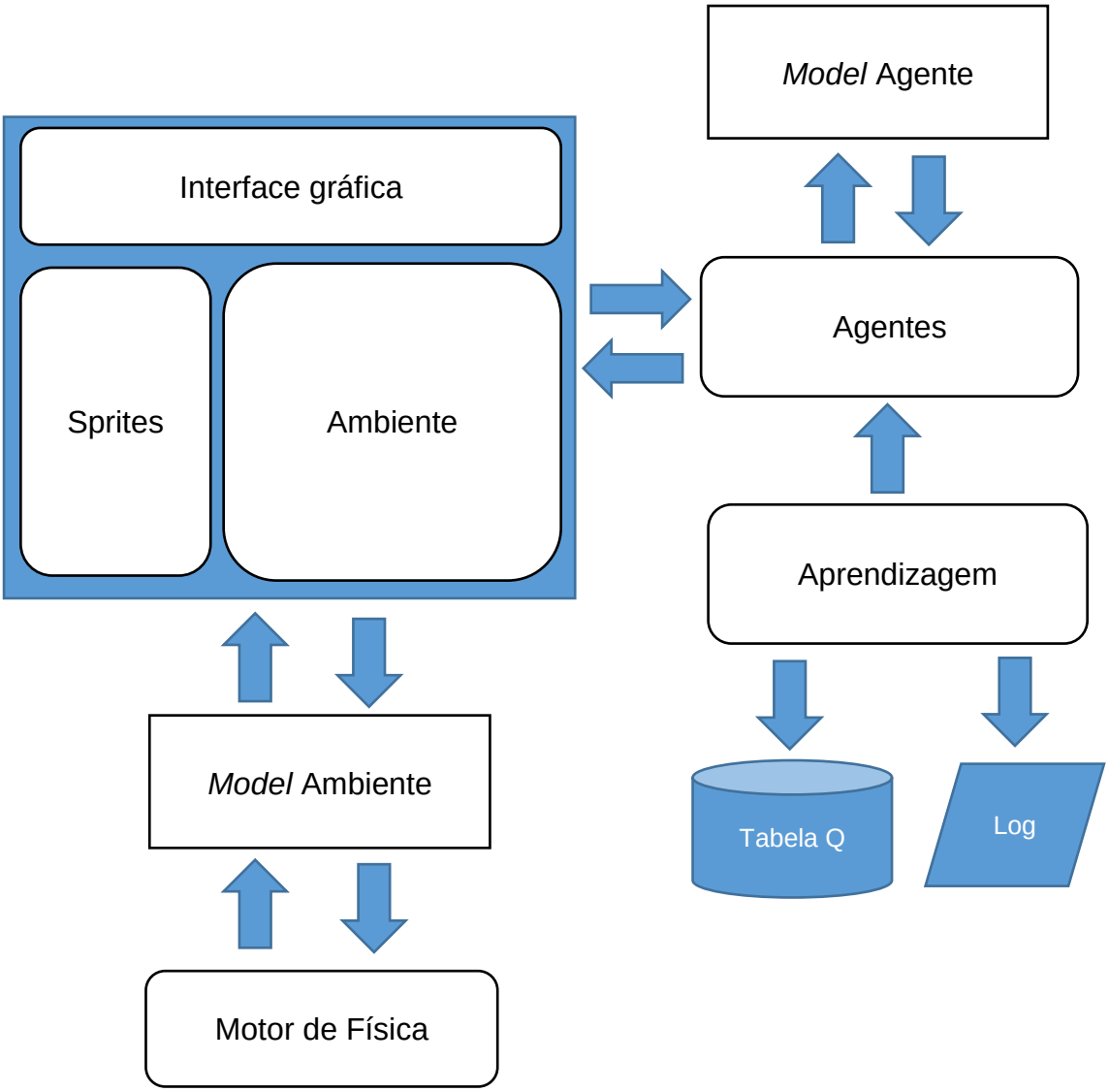
Fonte: Autor

4.4 ARQUITETURA DO AMBIENTE DE SIMULAÇÃO DE FUTEBOL DE ROBÔS

O sistema de simulação de futebol de robôs compreende dois componentes principais: ambiente e agente. A Figura 39 apresenta a arquitetura do sistema. O usuário interage com o sistema por meio da interface gráfica, que apresenta visualmente o ambiente de simulação. O ambiente simula fisicamente um local onde os agentes realizam suas tarefas. O motor de Física guia e determina os comportamentos dos agentes no ambiente. Os agentes são representados na interface gráfica por *sprites* únicos. Os agentes do time azul interagem com o ambiente e se valem da capacidade de aprendizagem por reforço (método Q-Learning). O resultado das ações e interações entre os agentes do time azul

alimentam a tabela Q e são armazenados em um log textual. Cada agente do time azul constrói e acessa somente a sua tabela Q, resultado de suas aprendizagens.

Figura 39 - Arquitetura do ambiente de simulação de futebol de robôs



4.4.1 Padrão de Projetos

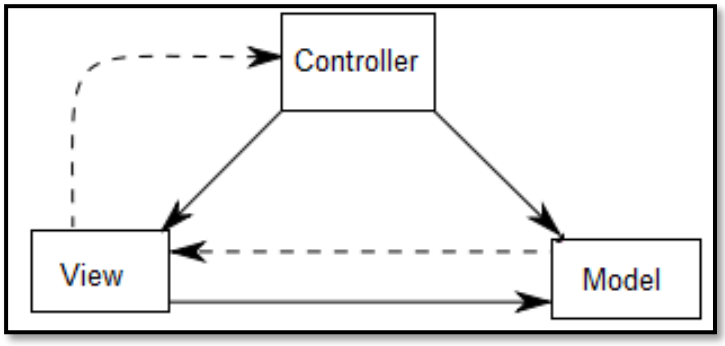
A ideia do uso de padrões para o desenvolvimento de softwares surgiu para formalizar soluções para os problemas que ocorrem frequentemente dentro de um contexto de desenvolvimento de software. Sua primeira publicação foi feita no livro “Design Patterns Elements of Reusable Object-Oriented Software” (GAMMA, HELM, *et al.*, 1995).

Pela necessidade de controle e da constante manipulação dos objetos responsáveis por guardar as informações de Física do simulador, o motor de física teve um peso alto na decisão do padrão de projeto a ser utilizado. Desta forma, o trabalho em questão utiliza uma mistura de dois padrões de arquitetura de software extraindo o melhor de cada um. Os padrões escolhidos foram, o padrão *model-view-controller* (MVC) e o padrão *Observer.*

O MVC tem como idéia principal a reutilização de código e a separação de conceitos. No ambiente de simulação o *model* é responsável por manter todos os objetos que não podem perder seus valores durante a execução do programa e proporciona uma maneira elegante e organizada de modificar a estrutura do programa quando necessário.

A Figura 40 abaixo ilustra a relação entre *view, model* e *controller*. Nela pode ser observado como o padrão MVC é estruturado. Nele a *view* possui uma representação dos dados presentes no *model*. O *controller* faz requisições para o *model* para atualizar os dados ou enviar informações e o *model* armazena as informações relevantes.

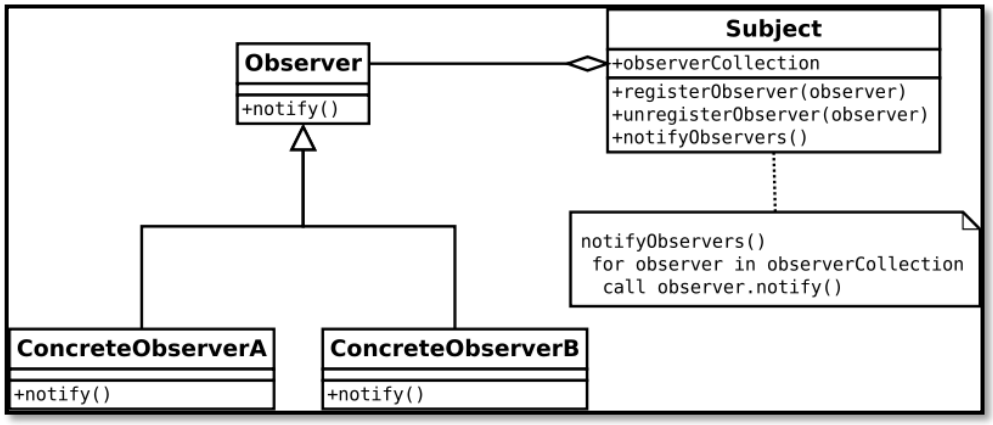
Figura 40 - Diagrama da relação entre *view*, *model* e *controller*



Fonte: Autor

O padrão *Observer* (Figura 41) tem como uma das principais vantagens manter atualizados objetos que são de uso comum (utilizados por mais de uma classe). O padrão *Observer*, define uma dependência de “um-para-muitos” entre os objetos de modo que quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente.

Figura 41 - Exemplo do padrão de projeto *Observer*



Fonte: https://pt.wikipedia.org/wiki/Observer

4.5 AGENTES

Pensando no agente como sendo uma entidade genérica, a classe “Agente” foi desenvolvida para ser a classe que controla o “corpo” do agente. Esta classe contém a estrutura das ações que podem ser executadas pelas funções goleiro e atacante. A definição da ação executada é controlada pela classe “Aprendizagem” que detém os conceitos de aprendizagem aplicados a cada um dos times.

Na classe “Agente”, o método principal “Agir” recebe como parâmetro a ação a ser executada e o objetivo do agente naquele instante. Com base nessa ação realiza os cálculos necessários para coordenar o “corpo” do agente e executar determinada ação relacionada o objetivo em questão.

Cada movimento realizado é calculado para um instante de tempo, sendo que a composição destes movimentos conclui a execução de uma ação seja ela chutar, defender, driblar e etc. O controlador do agente (classe que detém os princípios do aprendizado) é livre para mudar de objetivo e ação a qualquer momento, dando uma maior liberdade para o controlador agir caso o conjunto estado-ação não seja mais o ideal.

4.5.1 Os Movimentos do Agente

O repertório de movimentos disponível para o agente está ligado à função que o agente está desempenhando em campo, quando goleiro o agente pode executar os seguintes movimentos:

1. Defender
2. Posicionar
3. Reposição

Já quando o agente desempenha a função de atacante os movimentos disponíveis em seu repertório são as seguintes:

1. Chutar
2. Driblar
3. Fintar
4. Desarmar
5. Dominar a bola

Os movimentos do atacante acima são baseados em sequências de ações, sendo que a maioria dos movimentos compartilha um conjunto de ações semelhantes. Foram implementados três ações genéricas para servir de base para executar a grande maioria dos movimentos do goleiro e do atacante:

1. Ir para
2. Desviar
3. Conduzir a bola

A ideia principal da ação “Ir Para” é calcular a diferença entre o centro de massa do agente e o objetivo desejado e determinar a direção do corpo. Escolhendo uma das oito direções em que o agente pode mover-se baseado na Tabela 3, à direção escolhida está relacionada com um *Sprite* específico e o movimento de rotação será realizado durante algumas iterações utilizando *sprites* intermediários, com diferença de 15º cada para simular uma transição mais suave entre as direções, até que o corpo do agente esteja na direção escolhida. Por fim, se o objetivo não foi atingido, um valor de deslocamento é atribuído e o agente se desloca, caso contrário o agente permanece parado diante de seu objetivo aguardando novas instruções.

Tabela 3 - Tabela verdade direção do agente

θ	Δx	Δy
0º	$\Delta x > 0$	$\Delta y = 0$
45º	$\Delta x > 0$	$\Delta y > 0$
90º	$\Delta x = 0$	$\Delta y > 0$
135º	$\Delta x < 0$	$\Delta y > 0$
180º	$\Delta x < 0$	$\Delta y = 0$
225º	$\Delta x < 0$	$\Delta y < 0$
270º	$\Delta x = 0$	$\Delta y < 0$
315º	$\Delta x > 0$	$\Delta y < 0$

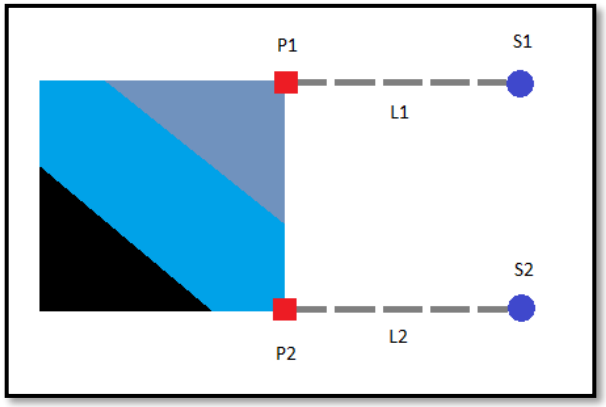
A ação “Ir Para” é composta por três micros ações, mover, parar, e rotacionar corpo, sendo estas as ações mais simples que o agente pode executar. A ação mover atribui a um vetor de controle do deslocamento do agente a quantidade de deslocamento de deve ser realizado no eixo x e y. Esta quantidade deslocamento é pré-determinada por parâmetros configuráveis da classe agente. Em contraste com a ação mover, a ação parar atribui valor de zero ao deslocamento nos eixos x e y. Por fim, a ação rotacionar corpo controla a direção do corpo do agente realizando um movimento rotacional de seu “corpo”. Baseado na resolução do ambiente de simulação foi definido que seria adequado o agente poder deslocar-se em oito direções:

1. Norte
2. Sul
3. Leste
4. Oeste
5. Noroeste
6. Nordeste
7. Sudeste
8. Sudoeste

Isto se deu pelo motivo de que movimento em ângulos menores que 45º geram valores para as componentes X e Y algumas vezes menores que um pixel. Sendo este o valor mínimo possível de movimento, acarretando em erros no deslocamento do agente.

O objetivo da ação “Desviar” é identificar qualquer tipo de obstáculo diferente da bola que esteja diante do agente. Para tal, cada agente foi programado para detectar um obstáculo a uma distância de pré-determinada pelos parâmetros do agente. A detecção dos obstáculos é realizada através de uma análise de intersecção de segmentos de reta, sendo assim é possível determinar se o “sensor” que detectou o obstáculo foi o do lado esquerdo ou direito e assim desviar para a direção correta. A Figura 42 ilustra o sistema de detecção de obstáculos do agente, sendo $P1$ e $P2$ dois pontos que delimitam o corpo do agente, $S1$ e $S2$ os dois pontos responsáveis por detectar obstáculos , e $L1$ e $L2$ os segmentos de reta de detecção.

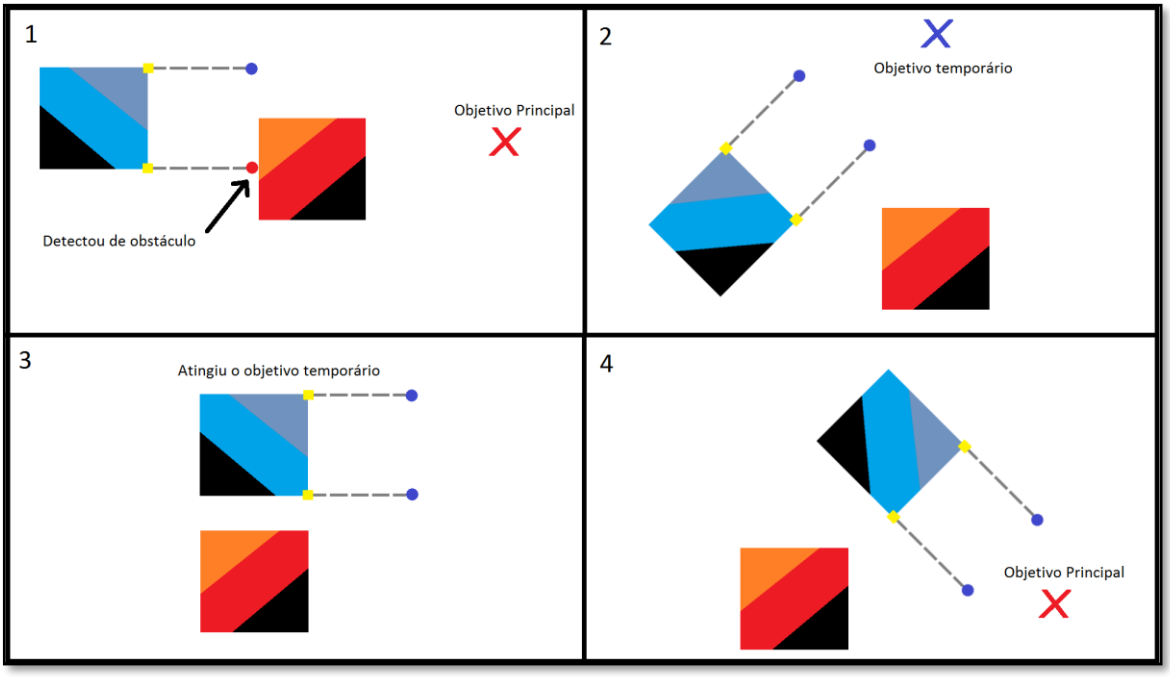
Figura 42 – Sensores virtuais de obstáculos agentes



Fonte: Autor

Realizar o desvio do agente significa calcular a coordenada que servirá como um objetivo temporário para o agente. Este objetivo temporário irá valer até o os sensores deixarem de detectar um obstáculo diante do mesmo. A Figura 43 ilustra a sequência de movimentos para executar a ação desviar.

Figura 43 - Sequência de movimentos da ação Desviar



Fonte: Autor

A ação “Conduzir bola” realiza uma manipulação no deslocamento e coordenadas do centro de massa da bola para que a mesma acompanhe os movimentos do agente enquanto ele executar a ação. O agente está apto a conduzir a bola apenas se a mesa estiver na sua frente e à uma distância inferior a um limiar parametrizado.

A ação “Chutar” assim como a ação “Conduzir bola” realiza uma manipulação no deslocamento da bola, a ação chutar aplica um deslocamento nos parâmetros físicos da bola baseado na força do chute do agente.

Por fim, os movimentos característicos do futebol de cada função são formados por um conjunto de ações que podem ser desmembradas da seguinte forma:

1. Defender;
    1.1. Através de um ponto de defesa calculado a partir do centro de massa da bola, executa-se a ação “Ir para”.
2. Posicionar
    2.1. Executa-se a ação “Ir para” utilizando como parâmetro as coordenadas do centro do gol.
3. Reposição

3.1. Executa-se a ação “Defender”.
3.2. Executa-se a ação “Chutar”.

4. Driblar (Executada apenas com a posse de bola)
   4.1. Executa-se a ação “Desviar”.
   4.2. Executa-se a ação “Ir para”.
   4.3. Executa-se a ação “Conduzir bola”.
5. Fintar (Executada mesmo sem a posse de bola)
   5.1. Executa-se a ação “Desviar” (Se existir obstáculo).
   5.2. Executa-se a ação “Ir para”.
   5.3. Executa-se a ação “Conduzir bola” (Se possuir a posse da bola).
6. Desarmar
   6.1. Executa-se a ação “Ir para” definindo o objetivo como sendo a bola.
7. Dominar a bola
   7.1. Executa-se a ação “Ir para” (objetivo definido sendo o gol adversário).
   7.2. Executa-se a ação “Conduzir bola”.

4.6 MOTOR DE FÍSICA

Motores de Física em aplicações computacionais de jogos ou simulação são bibliotecas, API (*Application Programming Interface*) ou *scripts* que lidam com as leis da Física para simular eventos reais em corpos virtuais. O motor de física desenvolvido neste trabalho utiliza o princípio de análise dos corpos rígidos, uma vez que nesta abordagem os corpos não sofrem deformações devido a forças atuantes sobre os mesmos. As leis da Física que ele incorpora são as mais básicas para descrever colisões entre corpos e o movimento dos corpos.

As colisões são baseadas em colisões elásticas uma vez que partindo do princípio que os corpos que estão no ambiente são rígidos, que não há dispersão da energia da colisão pelo som ou por conversão térmica e que o momento dos corpos é conservado (Princípio da conservação de energia), pode ser considerado que toda a colisão será elástica (conserva toda a energia) (HALLIDAY, RESNICK e WALKER, 2008).

O movimento dos corpos utiliza o princípio do movimento retilíneo uniformemente variado (MRUV), com característica de um movimento

uniformemente retardado quando os princípios de resistência ao movimento são aplicados a um corpo em desaceleração e vice-versa.

A resistência ao movimento dos corpos é baseada nos princípios de atrito onde uma constante definida em produto com a força normal de um corpo gera em uma força com sentido contrário ao movimento, para este trabalho apenas a força de atrito cinética foi levada em consideração.

Aplicar estas leis básicas da Física requer saber algumas informações dos corpos, referências como massa, centro de massa, limites dos corpos, força de atrito devem ser definidos, assim como ferramentas para manipulação vetorial e detecção de colisões são necessárias para a realização dos cálculos.

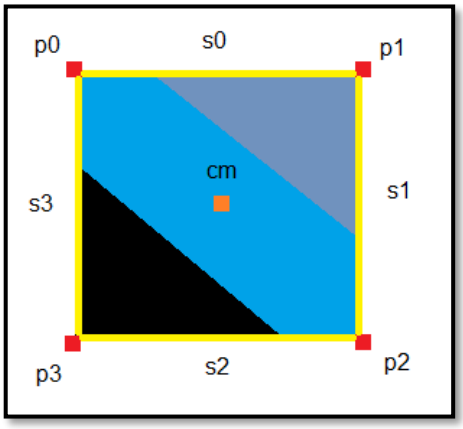
4.6.1 Referências Básicas

Antes de aplicar as leis da Física algumas referências importantes como às citadas nesta seção devem ser definidas.

O centro de massa foi calculado com base nas dimensões pré-definidas de cada corpo tomando como referência as coordenadas do componente canvas utilizado na representação gráfica.

Para diminuir o tempo computacional gasto, cada *Sprite* possui as coordenadas dos seus limites previamente calculadas tendo como origem o centro de massa, realizando apenas uma operação de translação para atualizar seus valores a cada iteração. Por convenção o primeiro ponto que define um corpo está localizado no canto superior esquerdo e o sentido utilizado para o restante foi adotado como sendo horário. No caso de corpos circulares o ponto zero está localizado a uma distância r no eixo x em relação ao centro de massa. Na Figura 44 estão representados os quatro pontos que delimitam um jogador (p0, p1, p2 e p3), assim como os segmentos de retas (s0, s1, s2, s3) e o seu respectivo centro de massa (CM).

Figura 44 - Referenciais base (Jogador)



Fonte: Autor

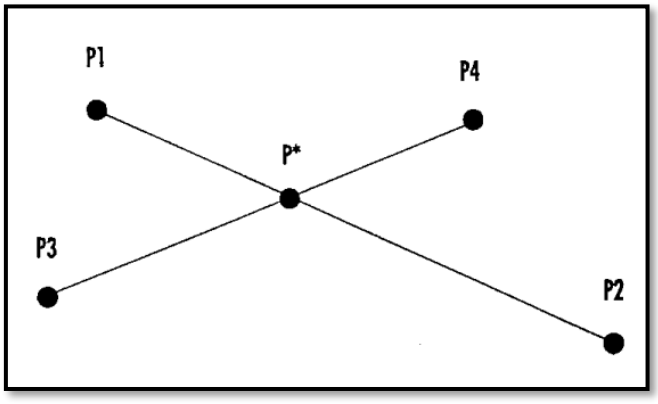
4.6.2 Intersecção de Segmentos de Retas

A intersecção entre os segmentos de retas neste motor de Física vai ter um papel de importância, e terá utilidade de detecção de colisões e auxílio no sensoriamento dos agentes.

Realizar esta análise de forma otimizada é importante, uma vez que diminuindo a quantidade de instruções necessárias para obter o resultado da análise de intersecção, o motor de Física pode reagir mais rapidamente a interação entre os corpos, aumentando a eficácia do motor de Física.

Na literatura existem alguns métodos para tratar deste problema geométrico. Um deles, apresentado no livro *Graphics Gems III* (KIRK, 1992), foi utilizado neste trabalho. Para entender o método em questão vamos considerar um segmento de reta $L12$ definido para dois pontos $P1$ e $P2$, e $L34$ definido para os pontos $P3$ e $P4$, como é ilustrado na Figura 45.

Figura 45 - Análise da intersecção de dois segmentos de reta



Fonte: (KIRK, 1992)

Representando cada ponto como um vetor bidimensional $\boldsymbol{P}1 = (x1, y1)$, um ponto $\boldsymbol{P}$ qualquer pode ser representado parametricamente sobre a linha $L12$ pela combinação linear de $\boldsymbol{P}1$ e $\boldsymbol{P}2$, onde α está no intervalo entre [0, 1] (KIRK, 1992). A equação apresenta a expressão de $\boldsymbol{P}$.

$$\boldsymbol{P} = \alpha\boldsymbol{P}1 + (1 - \alpha)\boldsymbol{P}2 \quad (1)$$

Reescrevendo de uma forma mais conveniente a Equação 2:

$$\boldsymbol{P} = \boldsymbol{P}1 + \alpha(\boldsymbol{P}2 - \boldsymbol{P}1) \quad (3)$$

Desta forma pode-se localizar a intersecção do ponto $\boldsymbol{P}^*$ computando α e β através da solução do sistema linear de equações descrito em 4 e 5.

$$\boldsymbol{P}^* = \boldsymbol{P}1 + \alpha(\boldsymbol{P}2 - \boldsymbol{P}1) \quad (4)$$

$$\boldsymbol{P}^* = \boldsymbol{P}3 + \beta(\boldsymbol{P}4 - \boldsymbol{P}3) \quad (5)$$

Subtraindo as equações tem-se 6:

$$0 = (\boldsymbol{P}1 - \boldsymbol{P}3) + \alpha(\boldsymbol{P}2 - \boldsymbol{P}1) + \beta(\boldsymbol{P}3 - \boldsymbol{P}4) \quad (6)$$

Dando nome para os valores intermediários, conforme as equações 7, 8 e 9:

$$\boldsymbol{A} = \boldsymbol{P}2 - \boldsymbol{P}1 \quad (7)$$

$$\boldsymbol{B} = \boldsymbol{P}3 - \boldsymbol{P}4 \quad (8)$$

$$\boldsymbol{C} = \boldsymbol{P}1 - \boldsymbol{P}3 \quad (9)$$

A solução para α e β é dada por:

$$\alpha = \frac{ByCx - BxCy}{AyBx - AxBy} \quad (10)$$

$$\beta = \frac{AxCy - AyCx}{AyBx - AxBy} \tag{11}$$

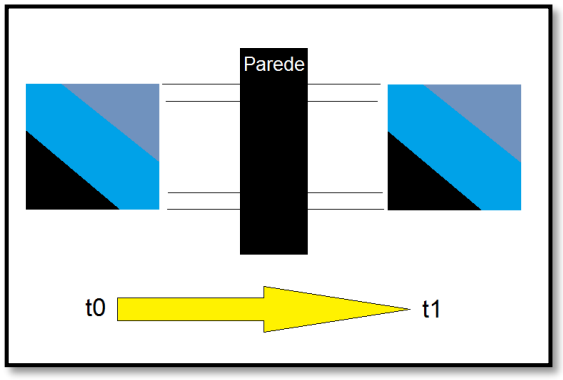
Se os resultados de α e β estiverem entre $[0,1]$ então os segmentos se interceptam. A partir desta analogia foi criada uma classe responsável pela detecção de intersecção entre segmentos de reta, a qual tem como argumentos de entrada dois segmentos de teste e um retorno booleano. Não é necessário saber com precisão o ponto de colisão, uma vez que as leis da Física de rotações não foram aplicadas neste trabalho. Ou seja, uma colisão em movimento linear não terá como resultado rotação de um dos corpos em torno de seu centro de massa.

4.6.3 Análise de Colisões

Pode-se dizer que uma colisão ocorre quando dois objetos se tocam. Por esta analogia, analisar colisões em um ambiente de simulação de futebol de robôs torna-se uma tarefa essencial e de muita importância uma vez que o resultado dela influência na dinâmica do jogo (HALLIDAY, RESNICK e WALKER, 2008).

Existem peculiaridades em relação ao tratamento de colisões em motores de física, uma vez que analisar colisões corresponde a detectar intersecções dos segmentos de reta que representam os limites de cada corpo, por meio da comparação destes com cada corpo contido no ambiente. Sendo assim, deve ser considerado que, por alguns instantes de tempo, dois corpos que estão colidindo irão se sobrepor até que os cálculos sejam realizados e forças contrárias aplicadas gerando uma reação a colisão. Um problema que pode ocorrer em uma colisão é o efeito de tunelamento, que ocorre quando um dos corpos desloca-se com velocidade muito alta, fazendo com que o tempo entre a detecção da colisão e a reação do motor de física a ela não seja rápida o suficiente para evitar que os corpos se atravessem. Este é um problema que deve ser tratado à parte, para não influenciar na dinâmica do jogo. A Figura 46 representa o efeito de tunelamento, onde o corpo no instante t_0 de análise do motor de física está indo em direção à parede, mas no instante $t_1$quando a colisão deveria ser detectada ele já percorreu uma distância suficiente para atravessar a parede.

Figura 46 - Representação do efeito de tunelamento



Fonte: Autor

Cada colisão detectada é tratada baseando-se nos princípios de uma colisão elástica (HALLIDAY, RESNICK e WALKER, 2008). Uma colisão é elástica quando ela conserva toda a energia contida nos corpos em colisão. Este aspecto deve ser levado em consideração na definição dos comportamentos dos corpos.

A relação entre as velocidades de dois corpos que colidem é dada pelas equações de conservação de momento e energia (SMID, 2016). Sendo $\boldsymbol{v}_1$ o vetor velocidade inicial do corpo 1 e $\boldsymbol{v'}_1$ o vetor velocidade do corpo um após a colisão com o corpo 2, temos que:

$$m_1 v_{x,1} = m_1 v'_{x,1} + m_2 \, \Delta v'_{x,2} \tag{12}$$

$$m_1 v_{y,1} = m_1 v'_{y,1} + m_2 \, \Delta v'_{x,2} \tan(\theta) \tag{13}$$

Onde $v_{x,1}$, $v_{y,1}$ são as componentes do vetor velocidade do corpo 1 antes da colisão, $v'_{x,1}$, $v'_{y,1}$ as componentes do vetor velocidade do corpo 1 após a colisão, $\Delta v'_{x,2}$ a diferença da velocidade do corpo 2 antes e após a colisão no eixo x e θ é a soma do ângulo da velocidade relativa entre os corpos 1 e 2.

$$\frac{m_1}{2(v^2_{x,1} + v^2_{y,1})} = \frac{m_1}{2(v'^2_{x,1} + v'^2_{y,1})} + \frac{m_2}{2\Delta v'^2_{x,2}(1 + \tan^2(\theta))} \tag{14}$$

Resolvendo a Equação 12 para $v'_{x,1}$ e a Equação 13 para $v'_{y,1}$ temos:

$$v'_{x,1} = v_{x,1} - \frac{m_2}{m_1 \Delta v'_{x,2}} \tag{15}$$

$$v'_{y,1} = v_{y,1} - \frac{m_2}{m_1 \Delta v'_{x,2} \tan(\theta)} \tag{16}$$

Substituindo as Equações 15 e 16 na Equação 14 e realizando algumas manipulações algébricas temos a solução indicada na Equação 17:

$$\Delta v'_{x,2} = \frac{2\left(v_{x,1} + \tan(\theta)\, v_{y,1}\right)}{(1 + \tan^2(\theta))(1 + \frac{m_2}{m_1})} \tag{17}$$

Colocando em evidência a componente velocidade do corpo 2. O ângulo θ pode ser relacionado como sendo o ângulo de impacto α e o ângulo da velocidade relativa γ_v, como pode ser observado na Figura 47.

Fazendo,

$$a = \tan(\theta) = \tan(\gamma_v + \alpha) \tag{18}$$

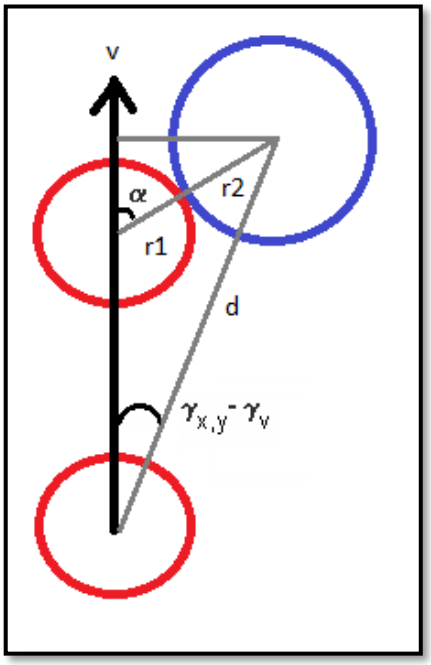
$$\gamma_{x,y} = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{19}$$

$$\gamma_v = \arctan\left(\frac{v_{y,1} - v_{y,2}}{v_{x,1} - v_{x,2}}\right) \tag{20}$$

$$\alpha = \sin^{-1}\left(\frac{d \sin(\gamma_{x,y} - \gamma_v)}{r_1 + r_2}\right) \tag{21}$$

Onde d, é a distância entre os centros de massa dos círculos (Figura 47).

Figura 47 - Colisão bidimensional



Fonte: Autor

Então temos o resultado na Equação 22.

$$\Delta v'_{x,2} = \frac{2\left[v_{x,1} - v_{x,2} + a\left(v_{y,1} - v_{y,2}\right)\right]}{(1 + a^2)(1 + \frac{m_2}{m_1})} \tag{22}$$

Das equações anteriores as componentes de velocidade após a colisão são dadas pelas relações:

$$v'_{x,2} = v_{x,2} + \Delta v'_{x,2} \tag{23}$$

$$v'_{y,2} = v_{y,2} + a\Delta v'_{x,2} \tag{24}$$

$$v'_{x,1} = v_{x,1} - \frac{m_2}{m_1}\Delta v'_{x,2} \tag{25}$$

$$v'_{y,1} = v_{y,1} - a\frac{m_2}{m_1}\Delta v'_{x,2} \tag{26}$$

Com base nas equações 23, 24, 25 e 26, a classe colisão elástica foi criada. Nela é realizada a detecção de colisões assim como todos os cálculos envolvidos e ajustes necessários nos deslocamentos instantâneos dos corpos em colisão.

4.6.4 Resistência ao Deslocamento

É inevitável que as forças de atrito façam parte do nosso cotidiano. Não podemos negligenciar a sua importância. Sem elas não seria possível caminhar, andar de bicicleta, segurar um objeto, escrever com um lápis ou mesmo utilizar pregos e parafusos. Contudo, se deixássemos o atrito agir sozinho, tudo em movimento rotacional pararia.

Segundo (HALLIDAY, RESNICK e WALKER, 2008), são demonstradas experimentalmente algumas das propriedades da força de atrito, quando um corpo se encontra pressionado sobre uma superfície (estando seco e não lubrificado):

1. Se o corpo não se move, então a força de atrito estática e a componente de força paralela à superfície são iguais em módulo e têm sentidos opostos.
2. O módulo da força de atrito tem seu valor máximo dado pelo produto escalar da componente normal e do coeficiente de atrito estático μ_e da superfície (27).

$$f_{e\,máx} = \mu_e\, N_s \tag{27}$$

3. Se o corpo deslizar sobre a superfície o módulo da força de atrito decresce rapidamente, visto que o coeficiente de atrito cinético μ_c é menor que o coeficiente de atrito estático.

$$f_c = \mu_c\, N_s \tag{28}$$

Os coeficientes μ_e e μ_c são adimensionais e devem ser determinados experimentalmente já que seus valores dependem do corpo e da superfície.

Baseado nestas premissas, uma classe denominada “Resistência ao deslocamento” foi criada e nela foram adicionadas tratativas para desacelerar qualquer movimento existente no ambiente.

Para movimentos com valores de deslocamentos próximos de zero, adicionou-se um limiar para o qual, deslocamentos abaixo deste valor é considerado deslocamento igual a zero (corpo em repouso). Esse limiar se fez necessário por limitações gráficas, uma vez que o movimento mínimo dos corpos é de um pixel.

Valores abaixo de um são armazenados em um acumulador para promoverem deslocamento nos próximos instantes de tempo.

4.7 APRENDIZAGEM DO TIME AZUL

Para fins de comparação de desempenho apenas o time azul utiliza o *Q-Learning* para controlar suas ações. O time vermelho utiliza uma lógica de percepção-ação onde para um determinado estado apenas uma ação específica pode ser executada.

A estratégia de jogo para ambas as equipes foi desenvolvida adotando as seguintes etapas:

1. Codificar as ações dos agentes (goleiro e atacante)
2. Codificar os estados do ambiente no qual os agentes estão inseridos
3. Codificar o algoritmo de aprendizagem por reforço Q-Learning (Time Azul)

As ações das funções goleiro e atacante foram codificadas na classe “Agente”. Os estados são interpretados como sendo as interações mais relevantes entre os agentes, o ambiente e a bola.

Para o goleiro os estados implementados foram:

1. Adversário fora da grande área e bola na direção do gol adversário;
2. Bola e adversário dentro da grande área;
3. Bola dentro da grande área e adversário fora da grande área;

Os estados definidos como relevantes para a função atacante foram:

1. Adversário com a posse da bola;
2. Atacante com a posse da bola e adversário atrás ou fora da trajetória atual do atacante;
3. Atacante com a posse da bola e adversário longe, distância superior a três robôs (80 pixels);
4. Atacante com a posse de bola e adversário perto, distância igual ou inferior a três robôs (80 pixels);
5. Nenhum jogador com a posse de bola;

4.7.1 Matriz de Recompensas

Baseado nas ações e estados definidos foi arbitrado valores para as recompensas do agente conforme a relevância da execução de certa ação em determinado estado. Estes valores são:

- -10 indica que o par estado/ação não é indicado para ser executado;
- 0 indica que o par estado/ação tem muito pouca relevância para ser executado;
- 5 indica que o par estado/ação tem pouca relevância para ser executado;
- 10 indica que o par estado/ação tem media relevância para ser executado;
- 20 indica que o par estado/ação tem alta relevância para ser executado;

Com base nas definições utilizadas, as matrizes de recompensa $r(s, a)$ para a função goleiro e atacante são apresentadas pela tabela 4 e tabela 5 respectivamente, em que as linhas representam os estados do ambiente e as colunas as ações que os agentes podem executar.

Tabela 4 - Matriz de recompensa do goleiro

Estado/Ação	Defender	Posicionar	Reposição
Estado 1	-10	20	-10
Estado 2	20	-10	5
Estado 3	10	0	20

Tabela 5 - Matriz de recompensas do atacante

Estado/Ação	Chutar	Drible	Domínio	Desarme	Finta
Estado 1	-10	-10	-10	20	0
Estado 2	20	0	-10	-10	-10
Estado 3	0	20	0	-10	0
Estado 4	5	0	20	-10	0
Estado 5	-10	-10	-10	-10	20

4.8 LOG DO SISTEMA

Para auxiliar na análise dos resultados da evolução do aprendizado do time azul, a classe “QLearning” conta com um log de informações que armazena os dados no formato CSV (*Comma-separated values*). Este arquivo vai conter as seguintes informações:

1. O tempo entre iterações (milissegundos);
2. O estado atual;
3. A ação escolhida para o estado o atual;
4. A recompensa agregada para o par estado-ação;
5. A identificação do goleiro que realizar uma defesa;
6. A identificação do atacante com a posse de bola;
7. Um sinalizador lógico para indicar um gol feito pelo time azul;
8. Um sinalizador lógico para indicar um gol feito pelo time vermelho.

Estas informações são importantes na análise da evolução do agente, pois a partir do arquivo de log é possível comparar o desempenho do time vermelho e do time azul em termos de saldo de gol, posse de bola, e jogadas efetuadas a cada instante de jogo.

5 TESTES E RESULTADOS FINAIS

Por definição, a fase de aprendizagem dos jogadores do time azul foi dividida em cinco partidas de futebol, cada partida teve a duração de dez gols.

A política adotada usa como base um valor randômico gerado em um intervalo de 0 a 100 e o compara com um limiar ε para controlar a exploração do agente. Quando o valor gerado é menor que ε é escolhida uma ação randomicamente para ser executada, caso contrário escolhe-se a ação com a maior recompensa. Em caso de conflito (duas ações com as mesmas recompensas) é escolhido uma das ações por sorteio. Durante a aprendizagem, a cada gol realizado por um dos times o parâmetro de exploração épsilon (ε) foi diminuído em 1% (método guloso) iniciando em 50% de exploração e terminando em 1% de exploração, buscando assim executar a ação com a maior recompensa um maior número de vezes. O valor de lambda (λ), que determina a importância das recompensas futuras, foi mantido por regra fixo em 50%. Desta forma observou-se uma rápida convergência para os pares de estado-ações desejados. A taxa de aprendizagem alpha (α) recebeu o valor de 1 para todos os pares estado-ação, dando grande importância ao aprendizado atual.

Inicializando a tabela Q do goleiro e do atacante conforme os valores demonstrados nas Tabelas 4 e Tabela 5, e seguindo a estrutura do algoritmo do Q-Learning que está explícita na Seção 2.5 (Aprendizagem por Reforço), realizou-se a fase de aprendizagem do time azul, com o objetivo de obter os valores de convergência para os pares estados-ações do goleiro e do atacante.

Os resultados dos valores de recompensa agregados a cada estado-ação são apresentados pelas Tabelas 6 e Tabela 7.

Tabela 6 - Matriz de recompensa final do goleiro

Estado/Ação	Defender	Posicionar	Reposição
Estado 1	10	40	10
Estado 2	40	10	25
Estado 3	30	20	40

Tabela 7 - Matriz de recompensas final do atacante

Estado/Ação	Chutar	Drible	Domínio	Desarme	Finta
Estado 1	10	10	10	40	20
Estado 2	40	20	10	10	10
Estado 3	20	40	20	10	20
Estado 4	25	20	40	10	20
Estado 5	10	10	10	10	40

Para fins de comparação do desempenho dos times, foi realizada uma amostragem de 35 jogos, sendo que cada jogo corresponde a um número definido de cinco gols no total. Nestes jogos foram analisados o número de vitórias, o saldo de gols e a porcentagem de posse de bola de cada time.

O time azul utilizou as tabelas 6 e 7 como base de sua aprendizagem, já o time vermelho manteve seu comportamento baseado em percepção-ação.

Para a análise de desempenho, o parâmetro de aleatoriedade épsilon (ε) utilizado no algoritmo Q-Learning foi mantido em 0% para que não haja exploração por parte dos jogadores do time azul durante os testes. A Tabela 8 apresenta os resultados dos jogos

Tabela 8 - Resultados após 35 jogos

Time	Azul	Vermelho
Nº de vitórias	**24**	11
Total de Gols	**100**	75
Média	2,857142857	2,142857143
Desvio Padrão	1,331581482	1,331581482
Variância	1,773109244	1,773109244

Pelos resultados apresentados na Tabela 8, pode ser observado que o time azul treinado pelo Q-Learning obteve um melhor desempenho em número de vitórias e saldo de gols em comparação com o time vermelho que utiliza e estratégia de percepção-ação.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foram compreendidos e aplicados os conceitos de arquitetura de agentes, sistemas multiagentes, método de aprendizagem por reforço (Q-Learning) e todos os conceitos necessários para desenvolver um ambiente de simulação de futebol de robôs; padrões de projeto, detecção de colisões entre corpos, algumas leis básicas da Física para interação entre corpos, componentes gráficos (*sprites)* e arquitetura de software.

A partir dos conhecimentos que foram extraídos das categorias de simulação de futebol de robôs, foi desenvolvido um ambiente de simulação de futebol de robôs o qual contém os elementos básicos de um simulador, como um motor de física, elementos gráficos e uma interface de comunicação com o usuário. Os agentes (jogadores) foram separados em dois times (azul e vermelho) onde cada time contém um jogador atacante e um jogador goleiro. O ambiente de simulação cumpriu seu objetivo proporcionando a comparação dos dois times (azul e vermelho), e ressaltando o desempenho superior do time azul que incorpora o uso da aprendizagem de máquina (método Q-Learning) com regras mais elaboradas para definir os seus objetivos, uma vez que sua estrutura se aproxima mais de um agente cognitivo do que de um agente puramente reativo.

6.1 CONTRIBUIÇÕES DO TRABALHO

Este trabalho contribui para a área dos sistemas multiagentes de forma que proporciona uma plataforma de simulação onde é possível analisar todos os conceitos discutidos neste trabalho. Para a área de Inteligência Artificial esta plataforma é um caminho para aplicar métodos e conceitos a fim de tornar os agentes (jogadores) mais perceptíveis em relação ao ambiente, aos outros jogadores, podendo desenvolver um time com jogadas mais complexar e até com estratégias de jogo.

A área da computação afetiva, tem potencial para realizar análises neste ambiente de conceitos emocionais em um sistema em cooperativo de agentes e comparar como a computação afetiva influencia no desempenho de um time de futebol de robôs simulado.

6.2 TRABALHOS FUTUROS

Com o desenvolvimento do sistema, notou-se que, para aplicar conceitos emocionais de uma forma mais elaborada é necessário que o agente possua uma percepção mais completa do seu ambiente e de tudo que o cerca, pois, a emoção relaciona o resultado do que é percebido pelo agente diante do contexto o qual este foi inserido.

Um possível trabalho futuro é aplicar com os conceitos emocionais uma vez que, neste trabalho houve dificuldade de aplicar os conceitos emocionais devido à complexidade que relaciona percepção-emoção, ou seja, a necessidade de o agente reconhecer situações que não foram inicialmente previstas e interpretá-las. Por exemplo, identificar quando a ação escolhida é correta, mas o resultado não é o esperado devido à dinâmica do jogo de futebol.

Outro possível trabalho é ampliar a quantidade de ações que cada jogador possui e proporcionar ao agente a escolhe de uma dentre várias ações possíveis para poder assim aplicar conceitos de decisão e comparar os resultados.

Pode ser realizado a melhoria do motor de Física como trabalho futuro para melhorar a dinâmica de jogo do ambiente de simulação e evitar possíveis erros de simulação de Física.

Por fim, outro trabalho futuro é a reestruturação do ambiente de simulação em uma plataforma de desenvolvimento de jogos 2D ou 3D, proporcionando uma interface gráfica mais adequada e com menores limitações.

REFERÊNCIAS BIBLIOGRÁFICAS

ABOUT - FIRA. **FIRA**. Disponivel em: <http://www.fira.net/contents/sub01/sub01_1.asp>. Acesso em: 16 jun. 2016.

ANÁLISE do Domínio de Aplicação: Futebol Robótico. **Universidade do Porto:** Faculdade de Engenharia, 2016. Disponivel em: <https://web.fe.up.pt/~lpreis/Tese/Capitulo8.PDF>. Acesso em: 27 maio 2016.

ASML net niet op tijd klaar voor Brazilië. **VisionandRobotics**. Disponivel em: <http://visionandrobotics.nl/2014/07/22/asml-net-niet-op-tijd-klaar-voor-brazilie/>. Acesso em: 15 Maio 2016.

AZEVEDO, L. L.; MENEZES, C. S. Agentes. Disponivel em: <http://www.inf.ufes.br/~liviaufmt/Disciplina/1.1.1%20agentes.htm>. Acesso em: 24 Abril 2016.

BAZZAN, A. Sistemas Multiagentes: Intrdução e Aplicações em Simulação e Controle de Tráfego e Simulação de Situações de Emergência. **Revista de Sistemas de Informação da FSMA**, p. 12-41, 2010.

BITTENCOURT, G. **Inteligência Artificial Ferramentas e Teorias**. 3nd. ed. Florianópolis: [s.n.], 2006. 16 p.

C plusplus. **Wikipedia**. Disponivel em: <https://en.wikipedia.org/wiki/C%2B%2B>. Acesso em: 16 jun. 2016.

CAMPONOGARA, E.; SERRA, M. Aprendizagem por Reforço: Uma Primeira Introdução, Agosto 2005.

CRUCIOL, L. Modelagem de Apoio à Decisão para o Problema de Espera no Ar Utilizando Sistema Multiagentes e Aprendizagem por Reforço, 08 Março 2012.

CS344M: Autonomous Multiagent Systems -- Fall 2015. **The University of Texas at Austin - Computer Science**. Disponivel em: <http://www.cs.utexas.edu/~patmac/cs344m/>. Acesso em: 15 Maio 2016.

DJUROVIC, S. Sprite manipulation in C#. **Code Project**. Disponivel em: <http://www.codeproject.com/Articles/1896/Sprite-manipulation-in-C>. Acesso em: 7 ago. 2016.

DURFEE, E. Distributed Problem Solving and Planning. **Artificial Intelligence Laboratory**.

GAMMA, E. et al. **Design Patterns - Elements of Reusable Object-Oriented Software**. 1º. ed. [S.l.]: [s.n.], v. I, 1995.

GRUPO de pesquisa em robótica da UFS, 2015. Disponivel em: <http://gprufs.org/#>. Acesso em: 15 jun. 2016.

H5AI, 21 Abril 2016. Disponivel em: <http://chaosscripting.net/files/competitions/RoboCup/WorldCup/2014/3DSim/RCSoccerSim3D_Rules2013.pdf>.

H5AI, 21 Abril 2016. Disponivel em: <http://chaosscripting.net/files/competitions/RoboCup/WorldCup/2014/3DSim/RCSoccerSim3D_RuleChanges2014.pdf>.

HALLIDAY, D.; RESNICK, R.; WALKER, J. **Fundamentos de Física 1**. 8º. ed. [S.l.]: LTC, 2008.

I WANT MY ROBOT PET. **JFDUBEAU**, 2015. Disponivel em: <http://www.jfdubeau.com/babbling-eloquently/2015/6/19/i-want-my-robot-pet>. Acesso em: 15 Maio 2016.

JAQUES, P. Estado da Arte em Ambiente Inteligentes de Aprendizagem que consideram a Afetividade do Aluno, Janeiro 2005. 20.
JUCHEM, M.; BASTOS, R. Engenharia de Sistemas Multiagentes: Uma Investigação sobre o Estado da Arte, Abril 2001.
JÚNIOR, F. Algoritmo Q-learning como Estratégia de Exploração e/ou Explotação para as Metaheurísticas GRASP e Algorítmo Genético, 11 Maio 2009. 1-140.
KIRK, D. **Graphics Gems III**. California: [s.n.], 1992.
LINGO (programming language). **Wikipedia**. Disponivel em: <https://en.wikipedia.org/wiki/Lingo_(programming_language)>. Acesso em: 16 jun. 2016.
LUGER, G. F. **Inteligência Artificial**. 4nd. ed. [S.l.]: [s.n.], 2004.
LUSTOSA, V. G. O Estado da Arte em Inteligência Artificial. **Revista Digital da CVA**, Costa Mesa, Cafifornia, p. 11, 2004.
MOREIRA, M. B.; MEDEIROS, C. A. **Princípios básicos de análise do comportamento**. 1nd. ed. [S.l.]: Artmed Editora S.A, 2007.
NAO. **Wikia**. Disponivel em: <http://robotics.wikia.com/wiki/NAO>. Acesso em: 15 Maio 2016.
ORDOÑEZ, B. et al. Sistema Multiagente para a Logística de um Centro de Distribuição, 2009.
PICARD, R. What does it mean for a computer to "have"emotions? **Emotions in Humans and Artifacts**, Massachusetts, 2002. 12.
RAO, A. S.; GEORGEFF, M. P. BDI Agents: From Theory to Practice, San Francisco, p. 14, Abril 1995.
ROBOCUP. Soccer Simulation League. **Wikipedia**, New York, p. 59, 16 Abril 2016. Disponivel em: <http://wiki.robocup.org/wiki/Soccer_Simulation_League#Rules>. Acesso em: 06 abril 2016.
ROBOCUP Brasil - Site Oficial. **Robo Cup Brasil**, 2012. Disponivel em: <http://www.robocup.org.br/objetivo_brasil.php>. Acesso em: 01 Maio 2016.
ROBOT Soccer Images. **Carnegie Mellon University - School of Computer Science**. Disponivel em: <http://www.cs.cmu.edu/~robosoccer/image-gallery/index.html>. Acesso em: 15 Maio 2016.
RUSSELL, S.; NORVIG, P. **Artificial Intelligence A Modern Approach**. New Jersey: Alan Apt, v. I, 1997.
SIMSPARK. **sourceforge**, p. 50, 6 Janeiro 2012. Disponivel em: <http://simspark.sourceforge.net/wiki/index.php/Main_Page>.
SMID, T. Elastic and Inelastic Collision in Two Dimensions. **Plasmaphysics.org.uk**, 2016. Disponivel em: <http://www.plasmaphysics.org.uk/collision2d.htm>. Acesso em: 18 out. 2016.
SOCCER Simulation. **SourceForge**. Disponivel em: <http://simspark.sourceforge.net/wiki/index.php/Soccer_Simulation>. Acesso em: 15 Maio 2016.
STEHOUWER, H. Emotional versus non-Emotional Agents in a predator-prey environment, 14 Junho 2004. 3.
STONE, P. Learning Multiagent Reasoning for Autonomous Agents. **Computers and Thought Paper**, p. 13-30.
WANG, F.; YANG, M.; YANG, R. The Intelligent Vehicle Coordination of the Cybernetic Transportation System. **International Journal of Advanced Robotic Systens**, p. 53-58, 2009.
WOOLDRIDGE, M. **MultiAgent Systems**. Liverpool: John Wiley & Sons, 2002.

ZAMBERLAM, A.; GIRAFFA, L. Modelagem de agentes utilizando a arquiterura BDI, Abril 2001. 1-32.

Apêndice A

Resultado dos jogos disputados

	Gols		Posse de Bola (milissegundos)	
Jogos	Time Azul	Time Vermelho	Time Azul	Time Vermelho
1	5	0	47880	74160
2	3	2	55400	78240
3	3	2	34600	83400
4	3	2	65920	64160
5	1	4	33640	26800
6	5	0	13280	40920
7	1	4	75760	53560
8	4	1	107200	85440
9	0	5	39400	89120
10	3	2	115400	106440
11	3	2	61640	63120
12	1	4	60680	96440
13	5	0	31160	76240
14	5	0	70000	68400
15	3	2	40120	67120
16	1	4	70800	60000
17	3	2	57440	17160
18	2	3	47640	81200
19	3	2	82800	206480
20	2	3	54640	60800
21	4	1	71880	119920
22	1	4	48760	67760
23	4	1	74200	12680
24	1	4	67520	51040
25	4	1	57560	60320
26	3	2	64600	29240
27	3	2	46000	63920
28	2	3	79280	85400
29	5	0	89960	227320
30	3	2	43520	65520
31	3	2	42680	37560
32	3	2	55880	107920
33	3	2	60800	59880
34	2	3	64080	59000
35	3	2	147960	124240