UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

PROPOSTA DE ALGORITMO DE TOMADA DE DECISÃO PARA PROGRAMAÇÃO DE ROBÔS PARTICIPANTES DA ROBOCUP

TRABALHO DE GRADUAÇÃO

Fernando Faé

Santa Maria, RS, Brasil
2016

PROPOSTA DE ALGORITMO DE TOMADA DE DECISÃO PARA PROGRAMAÇÃO DE ROBÔS PARTICIPANTES DA ROBOCUP

Fernando Faé

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Nº 414

Orientador: Prof. Dr. Giovani Rubert Librelotto

Santa Maria, RS, Brasil

2016

UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

A comissão Examinadora, abaixo assinada, aprova o Trabalho de Graduação

PROPOSTA DE ALGORITMO DE TOMADA DE DECISÃO PARA PROGRAMAÇÃO DE ROBÔS PARTICIPANTES DA ROBOCUP

elaborado por
Fernando Faé

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Giovani Rubert Librelotto, Prof. Dr. (UFSM)
(Presidente/Orientador)

Ana Trindade Winck, Profª. Drª. (UFSM)

Rodrigo da Silva Guerra, Prof. Dr. (UFSM)

Santa Maria, 11 de Julho de 2016.

RESUMO

Trabalho de conclusão de curso
Ciência da Computação
Universidade Federal de Santa Maria

PROPOSTA DE ALGORITMO DE TOMADA DE DECISÃO PARA PROGRAMAÇÃO DE ROBÔS PARTICIPANTES DA ROBOCUP

AUTOR: FERNANDO FAÉ
ORIENTADOR: GIOVANI RUBERT LIBRELOTTO
Local da Defesa e Data: Santa Maria, 11 de Julho de 2016.

Tendo como objetivo promover as pesquisas nas áreas de Inteligência artificial e robótica, a RoboCup promove anualmente competições de futebol de robôs. Estas competições demandam trabalho em diferentes áreas do conhecimento como: robótica, inteligência artificial, visão computacional, eletrônica, etc. No ano de 2015 a Taura Bots, equipe de futebol de robôs da Universidade Federal de Santa Maria, participou pela primeira vez da competição. Assim surgiram as necessidades de aprimoramento das capacidades motoras dos robôs e também da parte de comportamento para o controle dos mesmos. Nesse contexto foi criado o simulador TauraSim para auxiliar na parte de criação do comportamento para os robôs, eliminando a necessidade de se ter um robô disponível constantemente. Durante uma partida de futebol existem várias situações onde o agente tem mais de uma opção de ação, seja um passe, drible, chute etc. Nesse contexto faz-se necessário um algoritmo que seja capaz de realizar a melhor ação possível em determinada situação de jogo. Fazendo o uso do simulador TauraSim, foi possível criar um algoritmo de tomada de decisão para uma jogada específica, utilizando o classificador Naive Bayes para o cálculo das probabilidades, obtendo um resultado consideravelmente bom quando comparado a um algoritmo randômico, com uma porcentagem de 74% de acertos. Além de seu bom desempenho pode-se destacar também que: como o algoritmo Naive Bayes tem o seu conjunto de treinamento à parte de seu código principal, o algoritmo pode ser facilmente adaptado para outras jogadas, necessitando apenas a modificação de seu conjunto de treinamento.

Palavras-chave: Robocup. Simulação Computacional. Futebol de Robôs. Comportamento. Naive Bayes. TauraSim. Robótica.

ABSTRACT

Undergraduate Final Thesis
Graduation Program in Informatics
Federal University of Santa Maria

PROPOSAL OF AN ALGORITHM FOR A DECISION MAKING SYSTEM FOR ROBOTS PARTICIPANTS OF THE ROBOCUP

AUTHOR: FERNANDO FAÉ
ADVISOR: GIOVANI RUBERT LIBRELOTTO
Defence Place and Date: Santa Maria, July 11, 2016.

Aiming to promote research on fields such as artificial intelligence and robotics, RoboCup promotes annual robot soccer competitions. These competitions demand work on different knowledge fields like robotics, artificial intelligence, computer vision, electronics, etc. In 2015, the Taura Bots, robotics football team of Federal University of Santa Maria, participated for the first time in the RoboCup competition. Thus, came up the need to enhance the motor skills and the behaviour responsible for controlling the robot. In this context, was create a simulator called TauraSim, to assist on creation of the behaviour for the robot, abandon the need to have the robot always available. During a football match, there are many situations where the agent have more than one option of action, which can be a pass, kick, dribble, etc. With this in mind, it becomes necessary the creation of an algorithm that can be able to execute the best action given the situation of the match. Using the TauraSim simulator it was possible to create a decision-making algorithm for a specific situation of the match, making use of the Naive Bayes classifier to calculate the probabilities, getting considerable results, with a high percentage of correct answers. Another positive aspect is the fact that the Naive Bayes algorithm has its training data apart from its main code, it means that the algorithm can be easily adjusted to other match situations, just by modifying its training data.

Keywords: Robocup. Computer Simulation. Robot Soccer. Behaviour. Naive Bayes. TauraSim. Robotics.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

RIA *Robotic Industries Association*

IA *Inteligência Artificial*

FIFA *Fédération Internationale de Football Association*

UFSM *Universidade Federal de Santa Maria*

ANN *Artificial Neural Network*

NB *Naive Bayes*

JSON *JavaScript Object Notation*

RUR *Rossum's Univeral Robots*

KNN *k-Nearest Neighbors*

SUMÁRIO

1 INTRODUÇÃO

De acordo com a definição adotada pela *Robotic Industries Association* (RIA), o robô é um equipamento multifuncional e reprogramável, projetado para movimentar materiais, peças, ferramentas ou dispositivos especializados através de movimentos programados para a execução de uma infinidade de tarefas (RIVIN, 1988), essa definição pode ser complementada ainda pela definição utilizada pela *Oxford Dictionaries* onde diz: "Robô é uma máquina que se assemelha a um ser humano e é capaz de replicar certos movimentos e funcionalidades humanas automaticamente". Desde o primeiro uso do *Unimate*, o primeiro robô industrial, na produção automobilística (NOF, 1997), dispositivos robóticos vêm sendo aplicados em diferentes áreas com o objetivo de duplicar ou melhorar a função do trabalho humano e realizar tarefas que não podem ser executadas pelo ser humano por serem muito arriscadas ou até mesmo impossíveis de se executar (HOCKSTEIN et al, 2007).

Tendo em vista que a robótica é um campo preocupado com a conexão de percepção para ação (POMEROL, 1997), Inteligência Artificial deve ter um papel central na robótica se essa conexão for para ser inteligente. Em 1997 o robô Deep Blue, que foi criado especificamente para jogar xadrez, derrotou o então campeão mundial de xadrez Garry Kasparov (CAMPBELL et al, 2002), esse foi um dos principais feitos da IA e a partir de então novos desafios foram buscados pelos pesquisadores do ramo. Com essa vitória as pesquisas em IA buscaram novos desafios e dessa forma surgiu a RoboCup, com o objetivo de desenvolver até meados do século XXI uma equipe de robôs humanoides completamente autônomos capaz de derrotar o atual Campeão da Copa do Mundo de humanos, usando as regras oficiais da FIFA (KITANO et al., 1998).

A RoboCup é uma competição internacional cujo objetivo é proporcionar o crescimento da robótica e da inteligência artificial, proporcionando um desafio à comunidade científica focada nestas áreas (ROBOCUP, 2012). O futebol de robôs por sua vez, propicia um ambiente com condições para a validação de diversos assuntos relacionados à robótica como a inteligência artificial, visão computacional, eletrônica etc. A pesquisa e o desenvolvimento de futebol entre robôs além de ser extremamente motivante por possibilitar o surgimento de um espírito de ciência e tecnologia nas universidades, constitui uma atividade que possibilita a realização de experimentos reais para o desenvolvimento e testes de robôs que apresentam comportamento inteligente que cooperam entre si para a execução de uma tarefa formando um time (GOVEIA, 2006).

Em 2015 a RoboCup foi sediada na cidade de Hefei na China e teve a participação da equipe Taura Bots da Universidade Federal de Santa Maria (UFSM) em parceria com a equipe alemã WF-Wolves da Universidade de Ostfalia. A equipe Taura Bots conta com seu próprio simulador 2D, o TauraSim, que permite a realização de testes sem a necessidade do robô físico e além disso, que o código programado no simulador seja portado diretamente para o robô real (MONTENEGRO, 2015).

Durante uma partida de futebol existem várias situações onde o agente tem mais de uma opção de ação, seja um passe, drible, chute etc. Dessa maneira o agente deve entender o que está acontecendo em campo e analisar qual das opções possíveis é a melhor a ser realizada. Nesse contexto faz-se necessário um algoritmo que seja capaz de realizar a melhor ação possível em determinada situação de jogo.

O objetivo deste trabalho é a criação de um algoritmo que possa ser usado pelo simulador TauraSim para decidir entre a melhor ação possível para determinada formação de campo, analisando a situação de jogo onde o agente estará com a bola dominada e precisará decidir entre chutar, passar ou ficar com a bola (driblar/correr).

Para atingir o objetivo deste trabalho, foi criada uma estratégia de jogo definindo qual a melhor ação a ser tomada analisando as posições do agente, do companheiro, do adversário e das traves. Assim, conforme uma determinada posição de campo, é calculada a probabilidade de acerto do passe, chute e drible/corrida, usando a análise das posições. Para o cálculo das probabilidades é usada uma implementação do algoritmo Naive Bayes. Desta forma a ação com a maior probabilidade de acerto será a ação executada.

Este trabalho está estruturado da forma apresentada a seguir: o Capítulo 2 apresenta a revisão bibliográfica, dando destaque a definição de conceitos teóricos como robótica e RoboCup, Inteligência Artificial e o simulador TauraSim. O Capítulo 3 apresenta a metodologia, requisitos da proposta, modelagem e implementação. O Capítulo 4 apresenta os resultados obtidos neste estudo. Por fim, no Capítulo 5 são feitas as considerações finais sobre o trabalho desenvolvido e sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo destina-se à definição de conceitos teóricos necessários para a compreensão do trabalho, os quais são citados a seguir: Robótica e RoboCup, IA na robótica e o Simulador TauraSim.

2.1 Robótica e RoboCup

No início do século 20, robôs não faziam parte ainda da popular ficção científica. Isso continuou até 1917, quando Joseph Capek escreveu o conto *Opilec* e em 1921 quando seu irmão Karel Capek escreveu a peça *Rossum's Univeral Robots* (RUR), em que o conceito da robótica entrou para a consciência popular (HOCKSTEIN ET AL, 2007).

A transição da ficção científica para a realidade ocorreu em 1958 quando a General Motors apresentou o *Unimate,* o primeiro robô industrial, para ajudar na produção automobilística (NOF, 1997). Desde seu primeiro uso na linha de montagem em 1961, a aplicação de robôs na indústria tem aumentado e robôs têm sido usados numa grande variedade de aplicações incluindo uso militar, exploração do espaço e missões de busca e resgate. Em todos os casos, a robótica é usada para duplicar ou melhorar a função humana ou para fazer trabalhos muito perigosos para o trabalho humano direto (HOCKSTEIN et al, 2007).

A ideia do futebol robótico foi mencionada pela primeira vez pelo professor Alan Mackworth, da Universidade British Columbia, Canadá, onde ele tinha como objetivo a criação de equipes que pudessem trabalhar de forma cooperativa e competitiva. Mackworth ainda cita em seu artigo *On Seeing Robots* que além dos problemas com a parte da robótica e percepção, seriam enfrentados desafios na parte de representação de planejamento e ação (MACKWORTH, 1993). Mas, independentemente dele, um grupo de pesquisadores japoneses organizaram um *workshop* sobre os grandes desafios da Inteligência Artificial em outubro de 1992, em Tóquio. Este evento desencadeou uma serie de discussões sobre como utilizar o jogo de futebol para fomentar a pesquisa em ciência e tecnologia. Realizou-se, então, uma investigação sobre a viabilidade tecnológica, financeira e o impacto social. Juntamente, foram desenvolvidos regras e protótipos de robôs-jogadores e simuladores. Como resultado destes estudos, concluíram que o projeto era possível e desejável (MAEDA, 2004).

Assim, em junho de 1993, um grupo de pesquisadores incluindo, Minoru Asada, Yasuo Kuniyoshi e Hiroaki Kitano decidiram iniciar uma competição, primeiramente com o nome de *Robot J-League*. Esta acabou se transformando na primeira liga profissional de futebol robótico do mundo. Logo, devido a pedidos de pesquisadores de todo mundo também interessados, mudou-se o nome da liga e do projeto como um todo para *Robot World Cup Initiative*, ou simplesmente RoboCup (MAEDA, 2004).

A RoboCup é uma iniciativa para a continua pesquisa em robótica e Inteligência Artificial que usa o jogo de futebol como motivação. Desde sua primeira edição, em 1997, a RoboCup é realizada todo o ano e tem como seu maior objetivo desenvolver um time de robôs humanoides autônomos que possam derrotar a equipe humana campeã mundial em um jogo de futebol (KITANO et al., 1998). Atualmente a RoboCup Soccer é dividida em cinco ligas: Liga de simulação, Liga de tamanho médio, Liga de tamanho pequeno, Liga humanoide e a Liga padrão. Ainda existem outras divisões da RoboCup que são: RoboCup@Work, RoboCup Rescue, RoboCup Junior, entre outras.

Em 2015, a RoboCup foi sediada na cidade de Hefei na China e teve a participação da equipe Taura Bots da UFSM em parceria com a equipe alemã WF-Wolves da Universidade de Ostfalia (MONTENEGRO, 2015), onde os mesmos obtiveram o segundo lugar nos desafios técnicos na Liga Humanoide na categoria *Kid-Size* que comtempla robôs de 40 a 90 centímetros de altura.

2.2 Robótica e Inteligência artificial

Inteligência Artificial (IA) é um ramo da ciência da computação que tem como objetivo a criação de máquinas inteligentes e isto tem se tornado uma parte essencial da indústria tecnológica. A principal característica da IA é tentar entender os princípios que tornam o comportamento inteligente possível em sistemas naturais e artificiais (POMEROL, 1997). O campo da IA é muito vasto e bastante interdisciplinar de um modo que várias disciplinas têm estudos nessa área, incluindo ciência da computação, matemática, neurociências, filosofia, entre outras (POOLE; MACKWORTH; GOEBEL, 1998).

Tendo em vista que a robótica é um campo preocupado com a conexão de percepção para ação, a Inteligência Artificial deve ter um papel central na robótica se essa conexão for para ser inteligente. IA aborda algumas importantes questões como: qual conhecimento é necessário em qualquer aspecto do pensamento; como este conhecimento deve ser

representado; e como este conhecimento deve ser usado. Robótica desafia a IA forçando a mesma a lidar com objetos reais no mundo real (BRADY, 1984).

Para compreender o comportamento inteligente em seres humanos, animais ou máquinas, é necessário considerar o termo *embodiment* (personificação, incorporação). A teoria de *embodiment*, sugere que agentes autônomos do mundo natural são em sua maioria seres biológicos e que sua inteligência está diretamente ligada a seus corpos e a forma com a qual eles interagem com o ambiente. Isto é, o comportamento de todo o sistema não é apenas o resultado de uma estrutura de controle interno, mas também é afetado pelo nicho ecológico no qual o sistema é fisicamente incorporado e por sua morfologia, a forma do seu corpo e membros (PFEIFER et al. 2007).

O conjunto de técnicas estatísticas avançadas, acesso a grandes quantidades de dados e computadores muito rápidos produzem grande vantagem para o aprendizado de máquina e percepção. Muitas aplicações de sucesso utilizando a técnica de aprendizagem de máquina estão sendo usadas pelo mundo. Essa técnica que é incorporada aos programas consegue fazer muitas coisas que pareciam impossíveis na década de 80 (PRENSKY, 2002).

Existem várias técnicas de aprendizagem de máquina, entre elas temos: *Neural Network* (Redes Neurais), *Decision Tree* (Árvore de Decisão), *Naive Bayes Classifier* (Classificador Naive Bayes), *Support Vector Machine* (SVM), *Genetic Algorithms* (Algoritmos Genéticos), *k-Nearest Neighbors* (kNN), *Logistic Regression* (Regressão Logística), *Reinforcement Learning* (Aprendizagem por Reforço), entre várias outras.

2.2.1 Redes Neurais

Uma Rede Neural Artificial, do inglês *Artificial Neural Network* (ANN), é um paradigma de processo de informação inspirado no modelo que um sistema nervoso biológico, como o cérebro, processa informação. Seu sistema de processo de informação é composto de um grande número de elementos de processamento (neurônios) altamente interligados, trabalhando em união para resolver problemas específicos. Assim como os humanos, as ANNs aprendem através de exemplos. Nesse sentido, elas são configuradas para um específico trabalho ou aplicação, através de um processo de aprendizagem (STERGIOU; SIGANOS, 2011).

As ANNs são essencialmente simples modelos matemáticos que definem uma função $f: X \rightarrow Y$ ou uma distribuição sobre X ou ambos X e Y, mas em alguns casos esses modelos

também são associados com algum particular algoritmo ou regra de aprendizagem (GURNEY, 1997).

Em seu artigo *Towards Collaborative and Adversarial Learning: A case Study in Robotic Soccer*, o professor Peter Stone apresenta os resultados de seu experimento no uso de redes neurais para aprendizagem de comportamento, no contexto de futebol de robôs (STONE; VELOSO, 1998).

Fazendo o uso de uma rede neural para a melhoria do chute a gol, quando a bola está em movimento e sem obstáculos à frente, o agente praticante do chute foi capaz de marcar 91,5% das vezes, com a bola em diferentes velocidades de movimento (STONE; VELOSO, 1998).

2.2.2 Árvore de Decisão

Árvore de decisão é uma estrutura do tipo fluxograma onde cada nó representa um teste em um atributo, cada ramo representa o resultado desse teste e cada folha representa um rótulo de classe. O caminho da raiz às folhas representa a classificação das regras (DE VILLE, 2006). Também consideradas uma poderosa forma de analisar múltiplas variáveis, árvores de decisão oferecem capacidades únicas para complementar e substituir formas tradicionais de análise estatística e uma variedade de técnicas e ferramentas de mineração de dados (DE VILLE, 2006).

Uma proposta para avaliação do passe no futebol de robôs usando algoritmo de árvore de decisão é proposto pelo Prof. Peter Stone em *Layred Learning in Multi-Agent Systems* (STONES, 1998). O ato do passe requer a ação de dois agentes diferentes, o passador que deve chutar a bola em direção ao recebedor, que por sua vez deve segurar a bola. Partindo do princípio de que o agente recebedor tem a capacidade de interceptar bolas em andamento e que essa ação é idêntica a função de receber a bola, em um cenário com apenas o passador e o recebedor, a execução do passe não se torna difícil. Isto se torna mais difícil quando se tem a presença de um oponente que irá tentar interceptar o passe (STONES, 1998).

Para testar a performance da arvore de decisão foram rodados 5000 testes com o passador usando a arvore de decisão para escolher o recebedor, sendo que a arvore de decisão retorna uma estimativa de confiança da sua classificação, o passador pode escolher o melhor candidato a recebedor. Usando essa técnica o total de acertos foram de 65%, muito maior

quando comparado com a taxa de sucesso de 51% obtida com um algoritmo que escolhe o recebedor aleatoriamente (STONES, 1998).

2.2.3 Classificador Naive Bayes

O classificador Naive Bayes (NB) é um simples classificador probabilístico baseado na aplicação do Teorema de Bayes, assumindo que o valor de uma determinada característica é independente do valor de outra determinada característica (DOMINGOS; PAZZANI, 1997), por exemplo, uma fruta pode ser considerada um limão se for verde, redonda e com aproximadamente 6cm de diâmetro. O classificador NB considera a contribuição de cada uma dessas características independentes para a probabilidade da fruta ser um limão, eliminando qualquer relação entre cor, forma e tamanho (HALDANKAR; DESAI, 2015).

Apesar de ser muito simples o classificador NB já tem sua eficácia provada em muitas aplicações práticas incluindo classificação de texto, diagnósticos médicos e em sistemas de performance. Em 2004 uma análise do algoritmo NB mostrou que há sólidas razões teóricas para a aparente implausível eficácia do classificador (ZHANG, 2004).

Para entender melhor o classificador NB, primeiro devemos entender como funciona o modelo probabilístico de Bayes, já que o mesmo é baseado em sua estrutura.

O modelo probabilístico pode ser representado por um vetor $X = (X_1, \dots, X_n)$ representando algumas *n* características (variáveis independentes), esse vetor atribui para essas instâncias probabilidades $p(C_k|X_1, \dots, X_n)$ para cada resultado possível de *k*. Usando o teorema de Bayes esse modelo probabilístico pode ser reescrito como $p(C_k|X) = \frac{p(C_k)p(X|C_k)}{p(X)}$ onde :

- C_k representa qualquer hipótese cuja probabilidade pode ser afetada por dados.
- X corresponde a novos dados que não foram utilizados no cálculo da probabilidade anterior.
- $p(C_k)$ é a probabilidade de C_k antes de X ser observado.
- $p(C_k|X)$ é a probabilidade de C_k dado X, ou seja, depois de X ser observado.
- $p(X|C_k)$ é a probabilidade de observar X dado C_k.
- $p(X)$ é chamado de evidência modelo, este fator não entra em determinar as probabilidades relativas de diferentes hipóteses, pois é o mesmo para todas as hipóteses possíveis que estão sendo consideradas.

Tendo em vista o teorema supracitado, assumindo as condições de independência do classificador NB e usando a regra da cadeia para aplicações repetidas, podemos reescrever a expressão da seguinte maneira: $p(C_k|X_1, \dots, X_n) = p(C_k) \prod_{i=1}^{n} p(X_i|C_k)$

Uma abordagem usando uma variação do classificador NB é usada em (BUSTAMANTE; GARRIDO; SOTO, 2007) para criação de um algoritmo para a tomada de decisão em um time participante da RoboCup 3D, especialmente para a avaliação do passe.

Usando a variação Fuzzy Naive Bayes, com um classificador treinado com 1000 exemplos em um conjunto de 250 testes, foi possível obter um total de 76% de acerto nos cenários de teste (BUSTAMANTE; GARRIDO; SOTO, 2007), o que é relativamente melhor que os resultados mostrados por Stones que usou Árvore de Decisão para avaliação do mesmo processo (STONES, 1998).

2.2.4 Justificativa da escolha da abordagem Naive Bayes

Após verificar os resultados obtidos nos experimentos citados anteriormente em cada um dos diferentes tipos de algoritmos de aprendizagem de máquina, decidiu-se usar a abordagem do algoritmo Naive Bayes. Isso porque o mesmo mostrou-se superior ao algoritmo de Árvore de Decisão no quesito passe e mesmo que o uso de Redes Neurais tenha obtido um bom resultado no quesito chute a gol, esse resultado foi obtido quando não haviam adversários. Outro ponto também considerado foi que o uso do NB requer menos dados de treinamento, pois dado que as interações entre os atributos são ignoradas nesse modelo, não se faz necessário ter exemplos dessas interações e consequentemente menos dados de treinamento do que em outros algoritmos, como por exemplo: Regressão Logística e *K-Nearest Neighbor* (KNN) (ABNEY, 2014).

2.3 Simulador TauraSim

Para se fazer o desenvolvimento de um robô, é necessário o conhecimento em diversas áreas da tecnologia entre elas, engenharia mecânica, de controle e automação, elétrica e computação. Tendo em vista todas essas áreas envolvidas no projeto, fica necessário fazer uma subdivisão da equipe em equipes menores. A programação do robô dentro da equipe Taura Bots, está dividida em três subequipes: visão, comportamento e caminhada (MONTENEGRO, 2015).

A visão é a parte responsável por analisar o ambiente e registrar as informações encontradas sobre o mundo, que podem ser dos tipos: linhas, poste, robô, bola e objeto desconhecido. Após o registro dessas informações, elas são passadas para a parte do comportamento.

O comportamento, considerado a mente do robô, é responsável por analisar os dados recebidos pela visão e executar as estratégias de jogo. A forma que as informações são coletadas pela visão não é importante para o comportamento, pois para o seu funcionamento, o mesmo só precisa saber o tipo e posição dos objetos encontrados.

A parte da caminhada é responsável por executar o que foi processado pelo comportamento, executando os comandos de caminhada, chute ou defesa (para o robô goleiro) (BONINI, 2015).

Dentre essas subequipes, algumas necessitam o uso constante do robô, nesse contexto foi desenvolvido o simulador TauraSim para eliminar a dependência do robô na criação de estratégias de comportamento (MONTENEGRO, 2015).

O simulador permite ao programador manter o foco apenas na lógica de sua estratégia, pois a programação dos movimentos funciona e a forma como o robô coleta os dados sobre o ambiente é abstraída, como podemos ver na arquitetura do sistema na Figura 2.1.

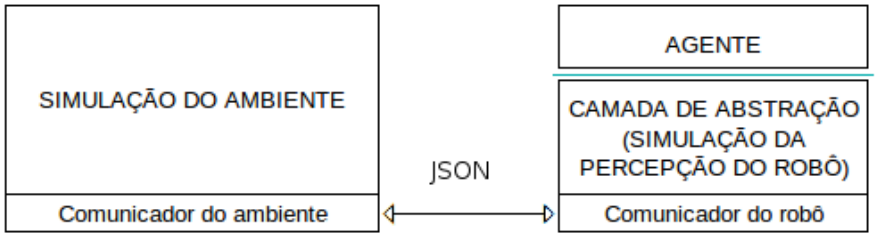

Figura 2.1: Arquitetura do sistema TauraSim (MONTENEGRO, 2015).

A representação dos objetos do ambiente é feita usando uma simulação 2D, que é a modelagem de um sistema real, imaginário ou hipotético, em um espaço bidimensional, ou seja, um espaço formado somente pelas dimensões altura e largura. A Figura 2.2 apresenta uma perspectiva global de objetos no espaço 2D.

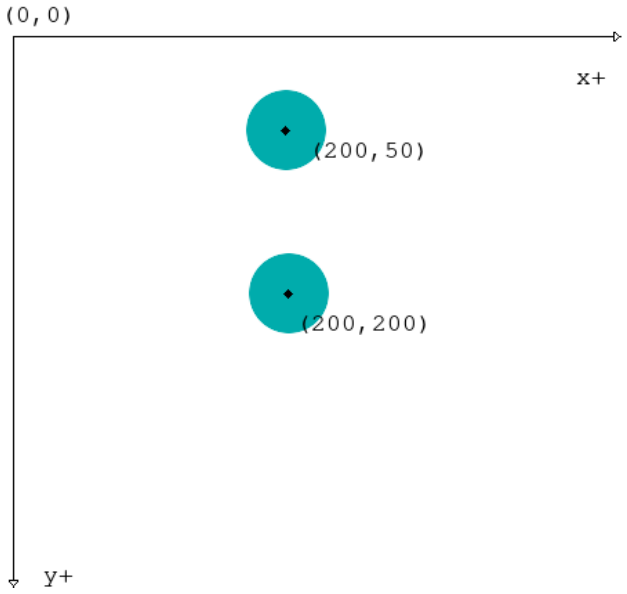

Figura 2.2: Objetos representados em uma perspectiva global no espaço 2D (MONTENEGRO, 2015).

A comunicação dos dados é feita através do formato *JSON*, o que permite a abstração por parte do programador sobre o recebimento dos dados, uma vez que esse formato representa os objetos em uma *string*. A Figura 2.3 representa um exemplo de mensagem enviada pela simulação do mundo.


```
{
  "head_angle": 0.0,
  "objects_list": [
    {
      "kind": "ball",
      "position": [
        239.9031999919439,
        -0.6876952725551101
      ]
    }
  ]
}
```


Figura 2.3: Exemplo de mensagem enviada no formato JSON (MONTENEGRO, 2015).

Neste exemplo temos os atributos *head_angle* e *objects_list*, onde o primeiro indica o ângulo da cabeça do robô em relação ao seu corpo e o segundo é um *array* contendo os objetos presentes no mundo, onde cada objeto é representado pelo atributo *kind*. Neste caso é passado somente um objeto *ball*, e sua posição que é representada pelo atributo *position*, que é um *array* de duas posições representando as coordenadas *r* e θ.

A mensagem enviada pela simulação do mundo, contém somente os objetos no campo de visão do robô e não todos os objetos que estão no mundo criado. A Figura 2.4 demonstra uma situação de mundo criada no simulador onde a mensagem passada seria semelhante à da Figura JSON.

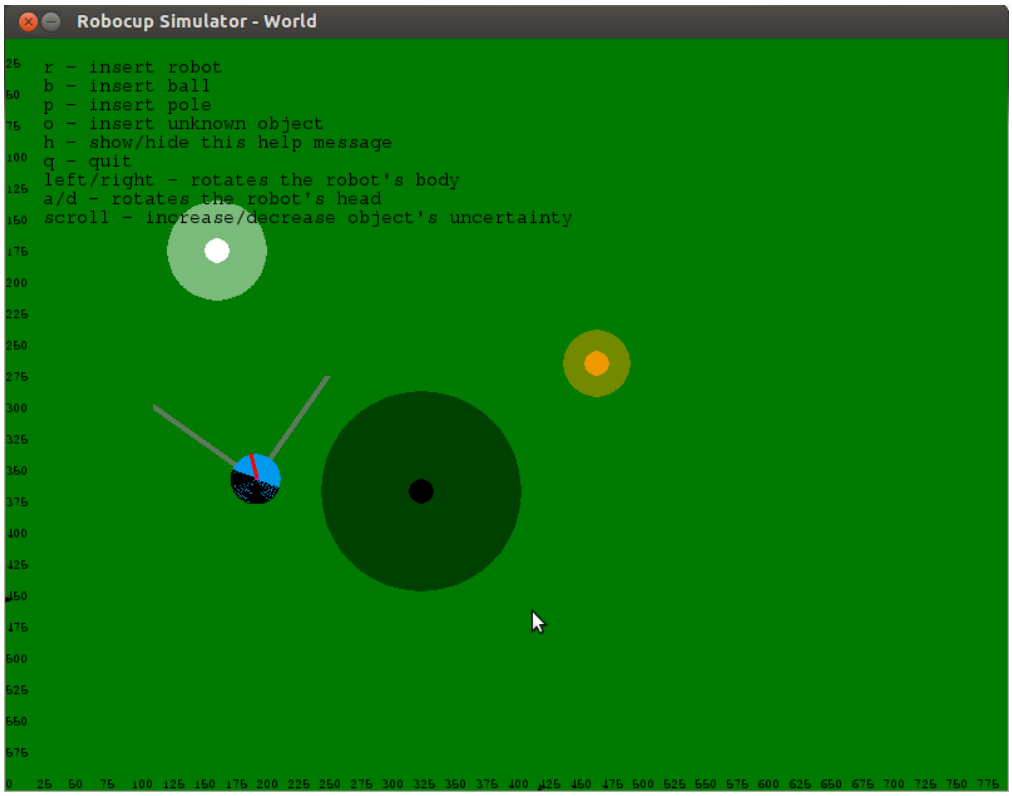

Figura 2.4: Situação de mundo criada no simulador TauraSim (MONTENEGRO, 2015).

2.4 Sumário do Capítulo

Enquanto o conceito de robótica começou como ficção científica, hoje em dia os robôs já fazem parte da vida moderna e pesquisas nesse ramo são constantes. Com o intuito de aumentar as pesquisas na área de robótica e também inteligência artificial, foi criada a RoboCup, objetivando o desenvolvimento de robôs autônomos.

Robótica e inteligência artificial são duas áreas que estão interligadas, já que a robótica é um campo preocupado com a conexão de percepção para ação, a inteligência artificial exerce um papel fundamental na robótica se essa conexão for para ser inteligente.

Hoje em dia com os computadores muito rápidos é possível utilizar da técnica de aprendizagem de máquina, que é incorporada aos programas, conseguindo fazer muitas coisas

que pareciam impossíveis na década de 80. Alguns exemplos de técnicas de aprendizagem de máquina são *Neural Network* (Redes Neurais), *Decision Tree* (Árvore de Decisão) e *Naive Bayes Classifier* (Classificador Naive Bayes).

3 METODOLOGIA

Este capítulo destina-se à contextualização, especificação e implementação do algoritmo proposto. Os tópicos estão distribuídos da seguinte forma: Requisitos da proposta e Modelagem e especificação.

3.1 Requisitos da Proposta

Para a criação do algoritmo de comportamento, foi necessário a definição de requisitos fundamentais para a programação dentro do ambiente do simulador TauraSim, como: entrada recebida pela visão e funções de controle.

No robô real as informações passadas para a visão são feitas através do formato *JSON*, como o simulador faz a abstração do robô, as informações passadas pelo mundo criado também são feitas nesse formato. Os dados que são passados para o comportamento no formato visto na Figura 2.3 e podem ser dos tipos: *ball*, *pole*, *robot* e *unknown*.

Cada objeto recebido, além de seu tipo contém um campo *position* que representam as coordenadas polares (r, θ) daquele determinado objeto em relação ao robô. No caso do exemplo mostrado na Figura 2.3, temos o objeto *ball* com $r \approx 239.90$ e $\theta \approx -0.687$.

Para a movimentação do agente é usado a função *setMovimentVector(speed, angle, phi)*, onde:

- *speed* é um número entre 0 e 1, de 0% para 100% da velocidade máxima do robô. Exemplo: *speed* = 0.5 vai configurar para 50% da velocidade máxima do robô.
- *angle* é a direção para onde o robô deve se mover variando entre -π até π. Exemplo: *angle* = 0 o robô vai se movimentar para a frente; *angle* = π/2 o robô vai se movimentar 90° para a esquerda.
- *phi* é um ângulo entre π e -π que ajusta a rotação do robô. Exemplo: *phi* = 0 o robô vai estar olhando para a frente; *phi* = π o robô vai rotacionar para a esquerda tentando olhar para trás.

Para realizar a ação do chute é usado a função *setKick(kick)* onde:

- *kick* é um número que irá representar o chute, sendo 1 para chute com a perna esquerda, e -1 para chute com a perna direita.

Como os objetos são tratados como pontos em um plano 2D, fica necessário também fazer o uso de algumas fórmulas matemáticas para obter algumas informações necessárias para a criação da tabela de treinamento usada no algoritmo. A principal usada é a fórmula para descobrir a distância entre dois pontos, que é descrita na Figura 3.1.

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Figura 3.1: Fórmula da distância Euclidiana entre dois pontos no plano 2D.

A fórmula apresentada na Figura 3.1, representa a distância entre dois pontos utilizando as coordenadas retangulares dos mesmos em um plano 2D. No caso apresentado temos os pontos A(X_1, Y_1) e B(X_2, Y_2) e a variável D que será o resultado da equação.

3.2 Modelagem e Especificação

Nesta seção serão apresentadas: a base de treinamento usada (*dataset*) para treinar o algoritmo e o algoritmo NB criado.

3.2.1 Criação do *Dataset*

Para a utilização do classificador Naive Bayes é necessário utilizar um *dataset* contendo dados de teste que serão usados para treinar o algoritmo, esses dados são exemplos de entradas com seus respectivos resultados de saída. Sendo assim, foi necessária a criação desse *dataset* com base nos dados obtidos através da comunicação do agente com o mundo.

Primeiramente, para poder criar a base de dados de treinamento é necessário criar uma estratégia de jogo. No futebol de humanos essa estratégia é definida pelo treinador do time, que treina seus jogadores para executarem as jogas que o mesmo pensa serem as melhores. A estratégia de jogo elaborada para a criação do *dataset* utilizado para este estudo foi criado pelo autor.

Como visto anteriormente, as informações sobre o mundo contêm as coordenadas polares dos objetos (r e θ), sendo assim foi montado o seguinte esquema apresentado na Figura 3.2.

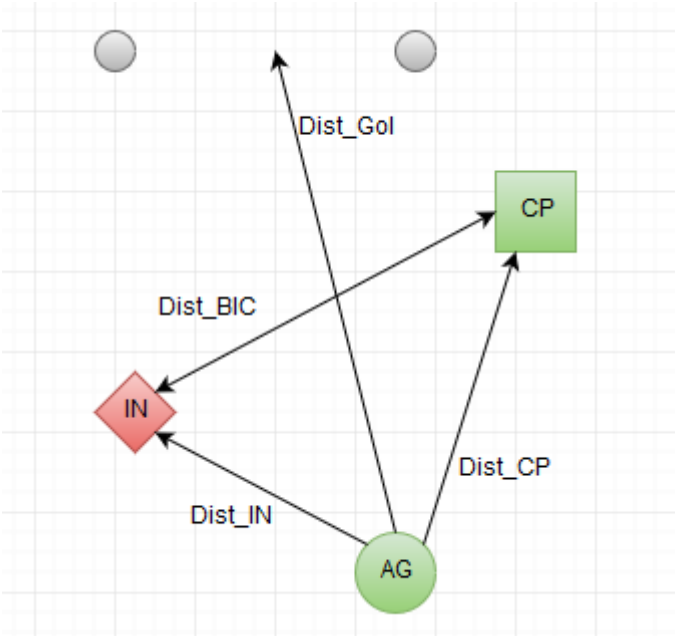

Figura 3.2: Esquema para criação do *dataset*.

Na Figura 3.2 temos o esquema que foi criado para a representação da estratégia usada para a criação dos dados de treinamento. A estratégia parte do princípio de que se o agente estiver pressionado, com o oponente muito perto, ele tem duas opções, o passe ou o chute. Se o agente estiver livre, com o oponente longe, o agente tem três opções, passe, chute ou corrida/drible. Como o chute do agente faz a bola andar a uma determinada distância *X*, se o agente estiver a uma distância do gol maior que *X*, o chute não é mais uma opção. Seguindo essa lógica, para o passe estar como uma ação válida, o companheiro não pode estar pressionado pelo oponente e também não pode estar atrás do mesmo.

Para saber qual das situações citadas uma determinada jogada se encaixa, usamos as informações que estão sendo representadas na Figura 3.2, distância do agente para o oponente (Dist_IN), distância do agente para o companheiro (Dist_CP), distância entre o oponente e o companheiro (Dist_BIC) e a distância do agente até o gol (Dist_Gol).

Após alguns testes, foi visto que fazendo somente o uso das informações descritas na imagem 3.2, não era possível obter todas as respostas necessárias, como por exemplo, saber se o companheiro está atrás do adversário ou não. Para isso foi feito o uso de mais algumas informações, que podem ser vistas na Figura 3.3, como o ângulo entre a cabeça do robô e o oponente (AG_IN) e o ângulo entre a cabeça do robô e o companheiro (AG_CP).

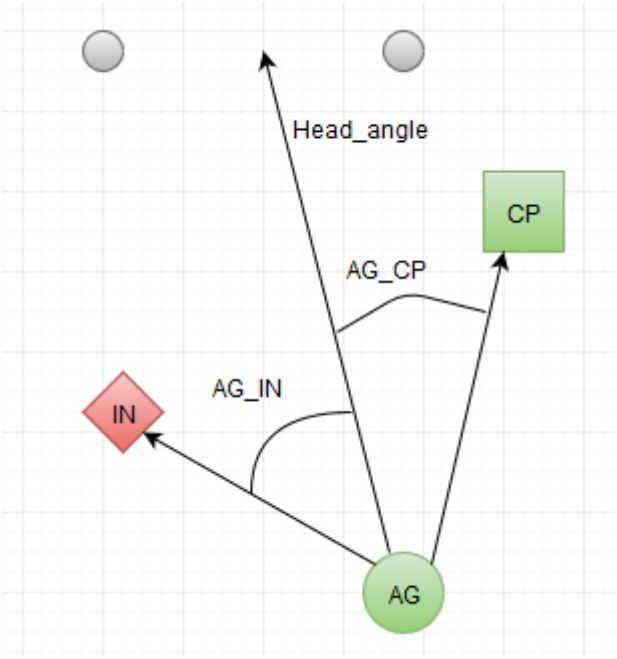

Figura 3.3: Esquema para criação do *dataset*.

Analisando o esquema das Figuras 3.2 e 3.3 temos as seguintes informações:

- *Dist_IN*: Representa a distância entre o agente e o oponente;
- *Dist_CP*: Representa a distância entre o agente e o companheiro;
- *Dist_Gol*: Representa a distância entre o agente e o gol;
- *Dist_BIC*: Representa a distância entre o oponente e o companheiro;
- *Head_angle*: Representa o ângulo da cabeça do robô;
- *AG_IN*: Representa o ângulo entre a cabeça do robô e o oponente;
- *AG_CP*: Representa o ângulo entre a cabeça do robô e o companheiro.

As informações representadas na Figura 3.2, são as informações usadas para a criação do *dataset* de treinamento, juntamente com a variável *Dist_BAN*, que é a diferença entre as variáveis *AG_IN* e *AG_CP* apresentadas na Figura 3.3, e é usada para saber o posicionamento do oponente em relação ao companheiro e vice-versa. Podemos ver na Figura 3.4 uma pequena amostra do *dataset* criado, que contém 100 casos de teste.

```
[DIST_GOAL, DIST_IN, DIST_CP, DIST_BIC, DIST_BAN]
[263.0877998050076, 58.96665925596525, 123.91473981554779, 64.96930634403232, 0.019427083026327563],
[172.85973631242842, 129.1654228802437, 96.72547834613913, 45.652220552085595, 0.28837315608239199],
[200.31458573105976, 144.7233648229155, 106.5412385014656, 46.483519415470619, 0.21390908216805013],
[200.31458573105976, 120.8575902169843, 106.5412385014656, 17.060697078485042, 0.11532952352364165],
[108.31009613060397, 57.355299740373226, 81.74226327492576, 45.434814598332586, 0.5674555295803516],
[163.01117283566419, 106.3315167894194, 109.73165600532072, 46.505285972282024, 0.4327484924091036],
[164.01111076819674, 33.47726750741192, 93.755600432720269, 72.838349801967699, 0.7471119500056504],
[236.19238687954083, 122.92443989449805, 124.80209942465825, 26.615506024795277, 0.2147613700217202],
[206.19238687954083, 122.92443989449805, 124.80209942465825, 26.615506024795277, 0.2147613700217202],
[195.19238687954083, 122.92443989449805, 124.80209942465825, 26.615506024795277, 0.2147613700217202],
[180.70536342458993, 128.55448799674122, 76.62490971798168, 52.566361718254974, 0.08221277229582574],
[197.52751579129094, 113.25169244539896, 146.86420292145527, 36.0847459896795, 0.10182632675732109],
[272.17529387509956, 75.18113840043894, 162.81854914943366, 107.38479063902444, 0.5685324276339765],
[272.17529387509956, 122.2931668593933, 164.03021448316588, 41.83765716288033, 0.020473946712390945],
[272.17529387509956, 166.61708572858126, 149.90034933704905, 48.18903037554931, 0.2869696567241582],
```

Figura 3.4: Amostra do *dataset* criado para o algoritmo NB.

Podemos ver a estrutura do *dataset* na Figura 3.4, onde temos um vetor com os dados recebidos, utilizando as variáveis apresentadas anteriormente no seguinte formato: [Dist_Gol, Dist_IN, Dist_CP, Dist_BIC, Dist_BAN].

Os dados vistos na Figura 3.4, estão no mesmo formato em que são recebidos pelo mundo (visão), como pode ser visto na Figura 2.3. Onde por exemplo, a variável Dist_IN seria o valor *r* das coordenadas polares (*r* e θ) do oponente.

3.2.2 Modelagem Naive Bayes

Conforme mostra a Figura 3.5, a classe *PreparaDados* contém as funções que servem para organizar os dados recebidos antes de passá-los para a classe *NaiveBayes* que irá executar as probabilidades.

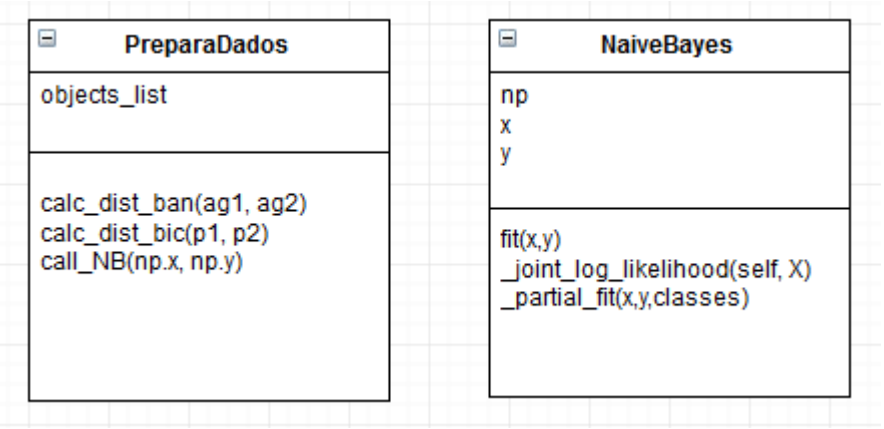

Figura 3.5: Representação das classes de funções.

Tendo em vista as classes utilizadas, podemos analisar o fluxograma do algoritmo que faz uso dessas classes na Figura 3.6, para analisar o seu comportamento e funcionamento.

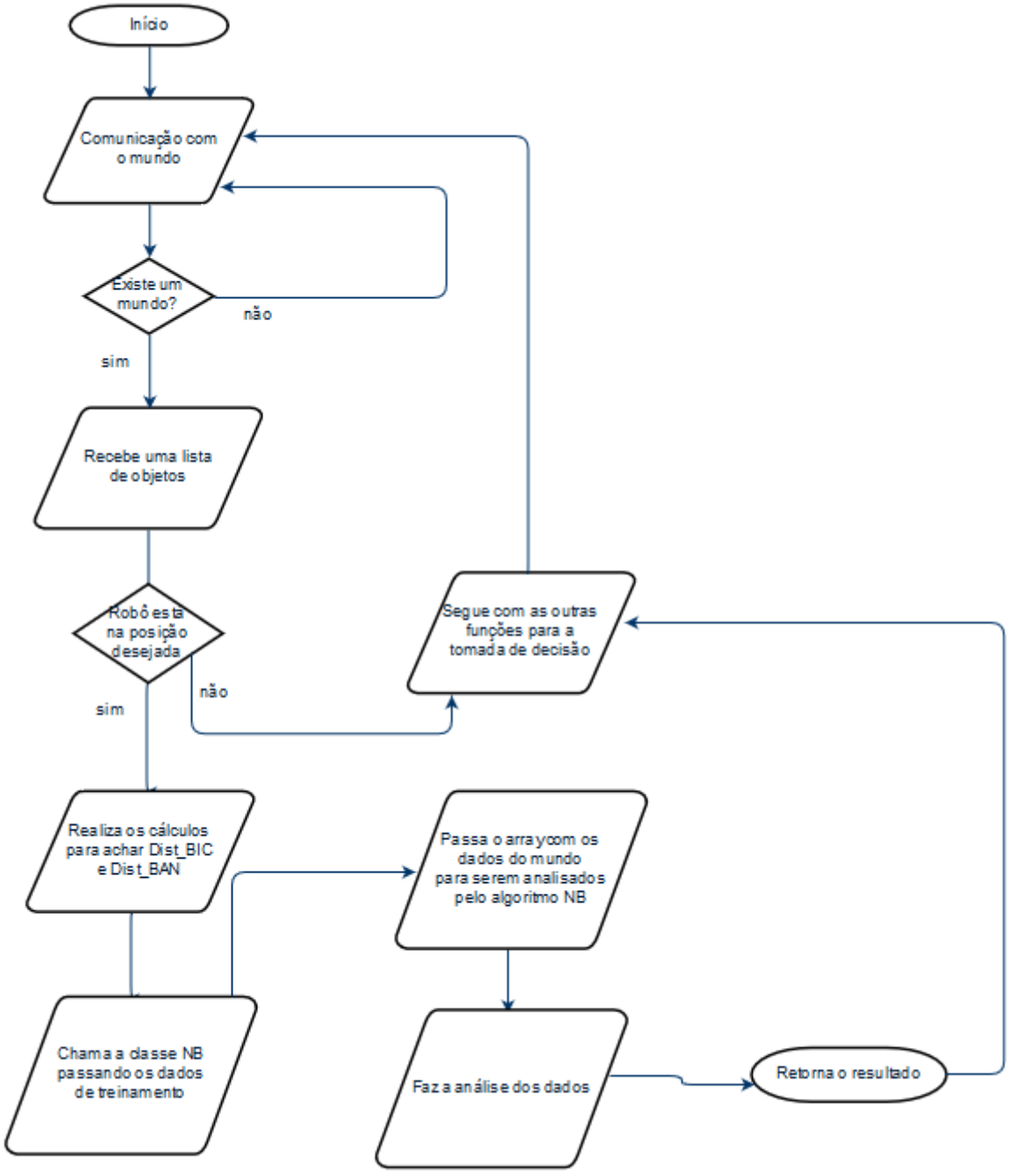

Figura 3.6: Fluxograma de funcionamento do algoritmo.

Inicialmente, o algoritmo geral irá tratar o comportamento do robô. Esse algoritmo irá fazer a comunicação com o mundo através do recebimento de um *array* contendo as posições dos objetos no mundo, para poder saber quais deles estão no campo de visão do robô. Após a

análise da lista de objetos, é então feita uma verificação para saber se a posição de campo condiz com a posição específica necessária para se fazer o uso do algoritmo criado, caso contrário o programa segue com as demais funções de tomada de decisão.

Quando a posição desejada de campo for passada, o algoritmo NB é então acionado, recebendo os dados do mundo, para fazer a análise dos mesmos. O algoritmo irá calcular a probabilidade de acerto para cada uma das ações possíveis, então uma saída é gerada contendo um valor que será a ação que deverá ser executada pelo robô. Esse valor será passado para as demais funções de tomada de decisão, que iram analisá-lo para poder saber qual ação se deverá executar.

3.3 Implementação

A implementação do algoritmo foi feita utilizando a linguagem *Python*, seguindo a modelagem apresentada na Figura 3.5 e o fluxograma apresentado na Figura 3.6.

Primeiramente são utilizadas as funções do próprio simulador para obter a lista de objetos do mundo, Figura 3.7, que irá ser lida para encontrar os componentes necessários para a criação do *array*, que serão gravados em suas respectivas variáveis para serem passados ao algoritmo NB.

```
while robot.updateSimulation():

    world = robot.perceiveWorld()
    robot.setKick(0)
    if not world:
        sys.exit("No world received")
```

Figura 3.7: Chamada para receber a lista de objetos.

Antes de passar os dados para serem analisados pelo algoritmo NB, é preciso passar os dados de treinamento que serão usados para calcular as probabilidades. Para isso são usados dois *arrays* de elementos, *X* e *Y*, contendo as entradas de teste e as saídas de exemplo respectivamente, como pode ser visto no exemplo da Figura 3.8.

```
X = np.array([[263,58,123,64,0.01], [172,129,96,45,0.28], [200,144,106,46,0.21]])
#CORRE = 1 | CHUTA = 2 | PASSE = 3
Y = np.array([1,2,3])
```

Figura 3.8: *Arrays* com os dados de treinamento.

Com os dados recebidos pelo mundo já analisados e salvos e com os dados de treinamento passados para o algoritmo, é então feita a chamada para o algoritmo NB para analisar os dados, calcular as probabilidades e retornar o resultado. Um pedaço do código utilizado para o cálculo das probabilidades pode ser visto na Figura 3.9.

```
def _joint_log_NB(self, X):
    check_is_fitted(self, "classes_")

    X = check_array(X)
    joint_log_NB = []
    for i in range(np.size(self.classes_)):
        jointi = np.log(self.class_prior_[i])
        n_ij = - 0.5 * np.sum(np.log(2. * np.pi * self.sigma_[i, :]))
        n_ij -= 0.5 * np.sum(((X - self.theta_[i, :]) ** 2) /
                             (self.sigma_[i, :]), 1)
        joint_log_NB.append(jointi + n_ij)

    joint_log_NB = np.array(joint_log_NB).T

    return joint_log_NB
```

Figura 3.9: Função utilizada para o cálculo da probabilidade.

3.4 Sumário do Capítulo

A criação do algoritmo para a tomada de decisão do robô, busca fazer com que o comportamento do robô seja mais inteligente e preciso, conseguindo realizar suas ações baseado em uma estratégia definida.

Foi definida uma estratégia de jogo e a partir da mesma criado um algoritmo para se fazer os cálculos das probabilidades de acerto de cada ação possível de ser executada, para saber qual delas o agente deve executar.

4 RESULTADOS

No futebol de robôs é muito comum a situação onde o agente tem pela frente um adversário e um companheiro, sendo assim é necessário que o agente faça uma leitura da situação para saber qual a melhor ação a tomar, se ela será um passe, chute ou drible/corrida, dependendo da situação de jogo.

Fazendo-se o uso do algoritmo de tomada de decisão que foi criado para o tipo de situação de jogo descrito anteriormente, foi obtido um bom resultado quando comparado com um algoritmo randômico usado para o mesmo propósito.

Podemos verificar na Figura 4.1 um dos testes feitos, onde o robô, seguindo a estratégia adotada, deverá seguir com a bola (correr/driblar), pois está muito longe das traves para tentar um chute, e tem o passe como arriscado, pois o adversário está muito perto do companheiro, e ligeiramente à frente do mesmo.

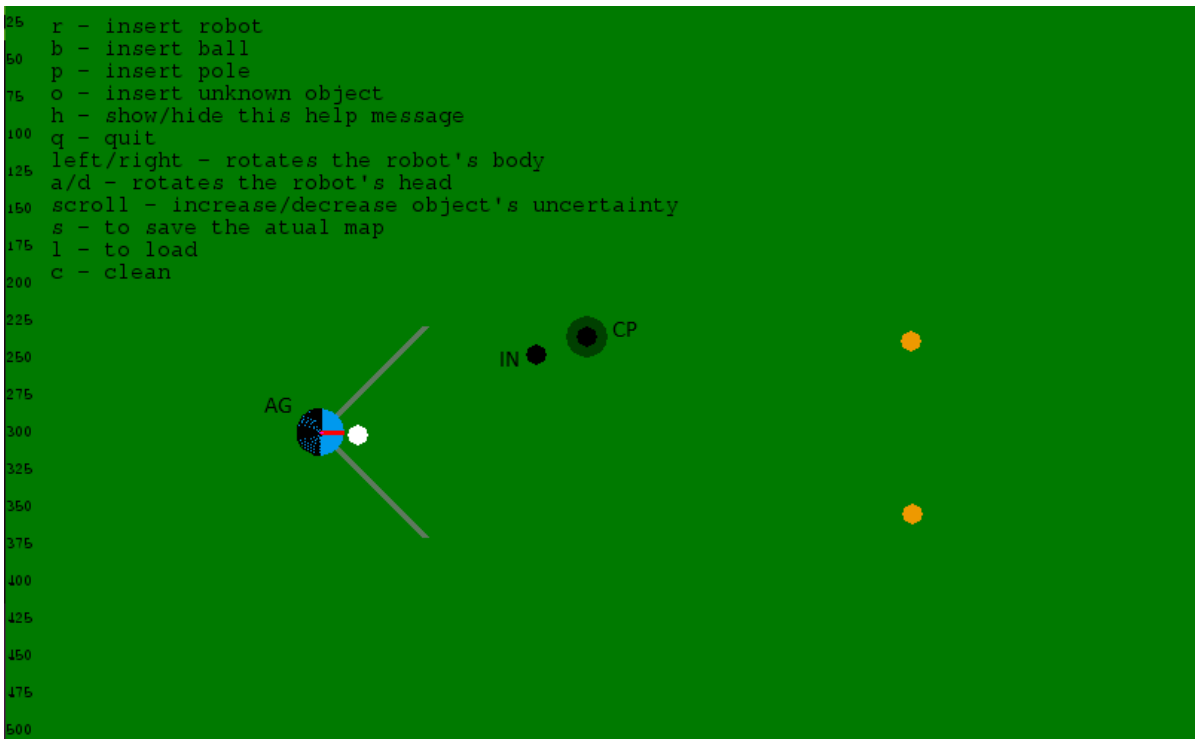

Figura 4.1: Exemplo de caso de teste criado para verificação do algoritmo.

Neste caso, o robô vai receber do mundo as informações contendo as coordenadas de cada objeto em seu campo de visão. Em seguida, será criado o *array* com as informações para serem calculadas pelo algoritmo. Esse *array* irá conter o mesmo tipo de variáveis apresentadas na Seção 3.2.1, um exemplo pode ser visto na Figura 4.2.

```
NB.predict([[pole2.position.r, enemy.position.r, mate.position.r, dist3, dist4]])
```

Figura 4.2: *Array* passado para o algoritmo NB.

Esse *array* apresentado na Figura 4.2 é do mesmo tipo de *array* usado para o treinamento do algoritmo, contendo assim: distância entre o agente e o gol, distância entre o agente e o oponente, distância entre o agente e o companheiro, distância entre o companheiro e o oponente e a diferença entre os ângulos do robô em relação ao oponente e ao companheiro.

O algoritmo NB pega então esses dados recebidos e começa a fazer os cálculos baseado nos dados de treinamento que foram passados, calculando a probabilidade de o resultado ser um chute, passe ou uma corrida, sendo que a distância entre o robô e o gol é igual a variável *pole2.position.r*, e faz o mesmo processo para todas as outras variáveis. Assim o algoritmo irá retornar a ação que obteve a maior probabilidade.

Como o algoritmo NB faz o cálculo das probabilidades baseado nos dados de treinamento, o mesmo algoritmo NB terá diferentes resultados para dados de teste diferentes. A estratégia adotada para esse trabalho, leva em consideração a posição de campo dos objetos como foi mostrado na Seção 3.2.1. Seguindo essa estratégia, quando passado as posições de campo do caso apresentado na Figura 4.1 para o algoritmo NB, o resultado esperado deverá ser: correr/driblar, para ser considerado correto, caso contrário o resultado será considerado errado.

Foi usado para o treinamento do algoritmo um *dataset* contendo 100 exemplos de jogadas com seus respectivos resultados, um exemplo do *dataset* usado pode ser visto na Figura 3.4, logo após, foram feitos 100 testes para 100 novas jogadas e obtido um resultado de 74% de acerto conforme a estratégia planejada. Fazendo o uso de um algoritmo randômico para a tomada de decisão e usando os mesmos 100 testes que foram usados para o outro algoritmo, foi obtido um resultado total de 38% de acerto conforme a estratégia planejada.

Os testes realizados foram feitos um a um, criando situações de jogo no simulador TauraSim, que então eram executadas e o seu resultado era anotado, comparando com o resultado que seria o esperado para a situação, para verificar se o mesmo era correto ou incorreto.

A Tabela 4.1 mostra alguns dos testes realizados e seus respectivos resultados. Os valores mostrados estão no mesmo modelo descrito na Seção 3.2.1.

Dist_Gol	Dist_IN	Dist_CP	Dist_BIC	Dist_BAN	Resultado Esperado	Resultado Obtido
280	122	158	36	0.04	Corrida	Corrida
184	99	126	32	0.15	Chute	Chute
184	99	124	102	0.92	Chute	Corrida
220	134	160	104	0.70	Passe	Passe
278	189	216	106	0.51	Corrida	Corrida
278	229	166	62	0.01	Corrida	Corrida
270	221	118	103	0.10	Passe	Corrida
252	46	154	124	0.74	Passe	Passe
250	49	59	39	0.72	Passe	Chute
177	49	59	40	0.73	Chute	Chute

Tabela 4.1: Exemplo de testes realizados no algoritmo NB.

Sendo assim podemos considerar que a abordagem criada leva vantagem sobre uma abordagem randômica, mesmo quando usado um *dataset* de treinamento considerado pequeno. Seu desempenho pode aumentar se um conjunto de dados maior for usado para treinamento ou que uma verificação maior das posições mais críticas e específicas seja feita.

Um exemplo de posição crítica pode ser visto na Figura 4.3. Temos no campo o agente e um adversário muito próximo dele, o que segundo a estratégia inviabilizaria o agente de permanecer com a bola, assim ele tentaria o chute ou passe, mas como o seu companheiro está atrás do adversário, o passe se torna muito ariscado, desse modo então ficaria sobrando somente a opção do chute.

Porém, sabendo que a bola só chega a uma determinada distancia com o chute, e que o robô está a uma distância maior que esta do gol, o chute também não seria uma opção.

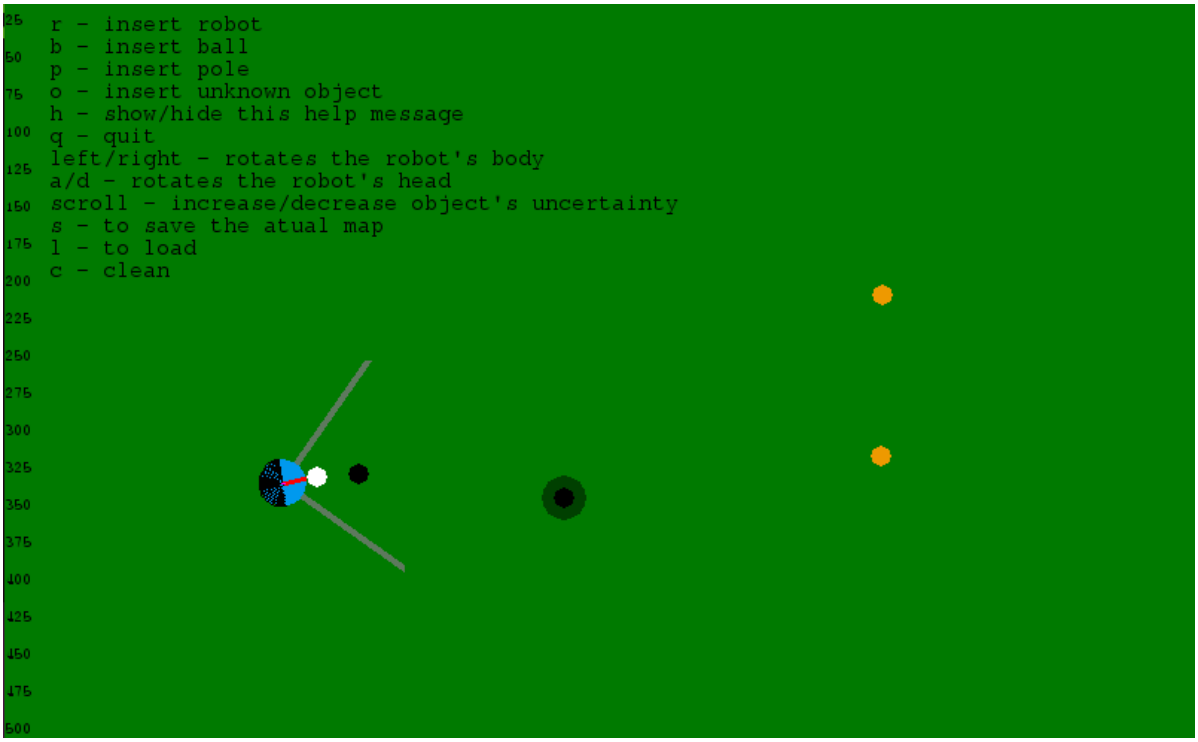

Figura 4.3: Exemplo de jogada crítica.

Sendo assim o algoritmo irá calcular dentre as 3 possibilidades de ação a melhor possível para executar, mesmo que a mesma não seja a correta segundo a estratégia de jogo. Esta situação deixa em aberto para trabalhos futuros, uma maior verificação nas possibilidades de ação para o agente incluindo por exemplo mais tipos de ação, como por exemplo uma ação "afastar" que serviria para a situação apresentada na Figura 4.3, ou fazendo uma mudança na estratégia de jogo.

4.1 Sumário do Capítulo

Para a criação de algoritmos para a tomada de decisão no cenário da RoboCup, o Classificador Naive Bayes se faz uma ferramenta muito útil, pois oferece uma portabilidade muito boa, já que o seu conjunto de teste é à parte de seu código, o que faz com que ele possa trazer resultados diferentes com conjuntos de treinamento diferentes.

5 CONCLUSÃO

Pesquisas nas áreas da robótica e inteligência artificial veem tendo um aumento muito grande nos últimos anos, nesse cenário a RoboCup tem um papel importante, pois ela fomenta o desenvolvimento de pesquisas nessas áreas, usando o tema futebol para a construção de robôs autônomos para este propósito. Em 2015 a Universidade Federal de Santa Maria, através da equipe Taura Bots em parceria com uma equipe alemã, fez sua primeira participação na RoboCup.

A equipe Taura Bots conta com um simulador próprio criado para a programação dos robôs, onde o código gerado no simulador pode ser facilmente transportado para o robô físico. Assim, tornasse menos complicada a criação de algoritmos para a tomada de decisão do robô, e como o simulador contém funções que ainda não foram implementadas no robô, é possível criar algoritmos que poderão ser usados no futuro quando essas funcionalidades forem adicionadas ao robô.

Tendo isso em vista, este trabalho tinha por objetivo criar um algoritmo para a tomada de decisão do robô utilizando o classificador Naive Bayes, para uma jogada especifica, onde o robô tem pela frente um adversário e um companheiro, e precisa decidir se ele chuta em gol, passa ou continua com a bola (dribla/corre), que tenha um bom desempenho para a estratégia de jogo que for adotada.

Para a criação do algoritmo foi usado o classificador Naive Bayes, que é conhecido pela sua simplicidade e eficácia, e que para a estratégia adotada para treinamento neste estudo funcionou muito bem, tendo sua eficácia comprovada pelos bons resultados obtidos.

Levando em conta a execução do trabalho, os objetivos principais foram alcançados: O algoritmo de tomada de decisão foi criado e testado, usando uma estratégia de jogo criada para este propósito. Os resultados obtidos foram consideravelmente bons, tendo um percentual de acerto de 74% nos testes realizados, utilizando um conjunto de dados relativamente pequeno para o treinamento do algoritmo.

Pode-se então concluir que o simulador TauraSim é de grande ajuda para a programação das estratégias para o robô, e que o uso do classificador Naive Bayes pode ter algumas vantagens para a criação de algoritmos de tomada de decisão para certos tipos de jogada, como a que foi apresentada neste trabalho.

Para trabalhos futuros, poderia ser feita uma análise mais aprofundada de estratégias, para tentar eliminar as posições críticas como a que foi vista anteriormente. Por ter seu

conjunto de treinamento à parte de seu código principal, o algoritmo NB também pode ser facilmente adaptado para outros tipos de jogada, necessitando apenas a modificação de seu conjunto de treinamento, e também a modificação da parte que irá gerenciar o resultado obtido pelo algoritmo. Assim fica possível também utilizar o algoritmo criado para ouros tipos de jogadas.

REFERÊNCIAS

ABNEY, S. **Anders Søgaard: Semi-Supervised Learning and Domain Adaptation in Natural Language Processing.** Machine Translation, v. 28, n. 1, p. 61-63, 2014.

BONINI, J. **A Definição de uma Linguagem para a Programação do Comportamento de Robôs Dentro do Contexto da RoboCup.** Rio Grande do Sul, Brasil, 2015.

BRADY, M. **Artificial Intelligence and Robotics.** Robotics and Artificial Intelligence, p. 47-63, 1984.

BUSTAMANTE, C.; GARRIDO, L.; SOTO, R. **Fuzzy Naive Bayesian Classification in RoboSoccer 3D: A Hybrid Approach to Decision Making.** RoboCup 2006: Robot Soccer World Cup X, p. 507-515, 2007.

CAMPBELL, M.; HOANE, A. J.; HSU, F.-h. **Deep Blue. Artificial Intelligence,** [S.l.], v.134, n.1, p.57–83, 2002.

DE VILLE, B. **Decision trees for business intelligence and data mining.** Cary, NC: SAS Institute, 2006.

DOMINGOS, P.PAZZANI, M. **On the Optimality of the Simple Bayesian Classifier under Zero-One Loss.** In: Machine Learning, v. 29, n. 2/3, p. 103-130, 1997.

GOVEIA, H. **Sistema Estrategista Para Futebol De Robôs.** São Paulo, Brasil, 2006.

GURNEY, K. **An introduction to neural networks**. London: UCL Press, 1997.

HALDANKAR, O.; DESAI, P. **Smart Predictor: A Road Incident Prediction System using Hybrid Data Mining Approach on Traffic Data.** In: International Journal of Science and Research, v. 4, 2015.

HOCKSTEIN, N. et al. **A History of Robots: From Science Fiction to Surgical Robotics.** J Robotic Surg, v. 1, n. 2, p. 113-118, 2007.

KITANO, H. et al. **RoboCup: A Challenge Problem for AI and Robotics.** RoboCup-97: Robot Soccer World Cup I, [S.l.], v.18, n.1, p.1–19, 1998.

MACKWORTH, A. **On Seeing Robots.** In: Computer Vision: system, theory, and applications. Cingapura: World Scientific Press, 1993.

MAEDA, Y. **Introduction to RoboCup Research in Japan.** In: Modeling Decisions for Artificial Intelligence, p. 1-6, 2004.

MONTENEGRO, F. **Simulador Computacional Para Auxiliar No Desenvolvimento De Estratégias De Comportamento Para Futebol de Robôs**. Rio Grande do Sul, Brasil, 2015.

NOF, S. **Handbook of industrial robotics**. New York: John Wiley, 1999.

PFEIFER, R.; BONGARD, J. C**. How the body shapes the way we think: a new view of intelligence**. Cambridge, Mass.: MIT Press, 2007.

POMEROL, J. **Artificial Intelligence and Human Decision Making.** European Journal of Operational Research, v. 99, n. 1, p. 3-25, 1997.

POOLE, D.; MACKWORTH, A.; GOEBEL, R. **Computational Intelligence.** New York: Oxford University Press, 1998.

PRENSKY, M. **The Motivation of Gameplay.** On the Horizon, v. 10, n. 1, p. 5-11, 2002.

RIVIN, E. **Mechanical Design of Robots**. 1.ed. Nova Iork: [s.n.], 1988.

ROBOCUP. **Objetivo da RoboCup Federation.** Site Oficial RoboCup Brasil, 2012. Disponível em: <www.robocup.org.br/objetivo.php>. Acesso em: 15 abr. 2016.

STERGIOU, C.; SIGANOS, D. **Neural Networks.** London, United Kingdom 2011.

STONE, P. **Layered Learning in Multi-Agent.** Pittsburgh, United States, 1998.

STONE, P.; VELOSO, M. **Towards Collaborative and Adversarial Learning: A Case Study in Robotic Soccer.** In: International Journal of Human-Computer Studies, v. 48, p. 83-104, 1998.

ZHANG, H. **The Optimality of Naive Bayes.** New Brunswick, Canadá, 2004.