Aplicando Inteligência Artificial na Simulação de Futebol de Robôs

CAIO ARCE NISHIBE [1]
RENATO GIRARDI GASOTO [1]

[1]UTFPR – Universidade Tecnológica Federal do Paraná
DAINF – Departamento Acadêmico de Informática
{caionishibe,re_gasoto}@gmail.com

Resumo: Este artigo apresenta uma abordagem que emprega inteligência artificial, mais especificamente Lógica Fuzzy e Redes Neurais, para melhorar o desempenho de um atacante perante um goleiro em um ambiente simulado de futebol de robôs.

Palavras Chaves: Tewnta, Futebol de Robôs, Fuzzy, Redes Neurais, Inteligência Artificial.

1 Introdução

Criar soluções automatizadas para a sociedade é um desejo utópico desde a formação do conceito de robôs. Um mundo onde a única ocupação do ser humano é filosófica ou deleite, e todo o trabalho para manter cidades funcionando é desempenhado por robôs. Contudo, adquirir capacidade de resolução de problemas não é uma tarefa trivial e requer muito esforço de programadores.

Todo esse esforço gerou diversas técnicas que, apesar de ainda não ser exatamente a inteligência que se esperava obter, ajudam o sistema a se adaptar a e/ou prever novas situações, a partir de experiências passadas.

Para demonstrar esta capacidade, temos como exemplo o futebol com robôs, no qual se tem um objetivo claro: marcar gols e evitar que o outro time faça gols. Apesar de parecer um objetivo simples, atingi-lo é uma tarefa dispendiosa, deve-se estudar o oponente, encontrar suas fraquezas e explorá-las, para facilitar o alcance do objetivo.

1.1 Proposta

Utilizando o simulador Tewnta, desenvolvido pelo Cientista da Computação Gabriel Detoni [1]. Foi implementado um algoritmo para um atacante fazer gols em um goleiro fornecido .

Este é um sistema bastante simplificado, pois é somente a interação entre goleiro, bola e um atacante. Isto se deve ao curto prazo dado para a execução do projeto.

2 Controladores Fuzzy

Os controladores fuzzy são caracterizados pelo emprego de regras lógicas no algoritmo de controle, com o propósito de descrever neste a experiência humana, intuição e heurística para controlar um processo [6].

Neste trabalho foram empregados dois controladores fuzzy: um para a decisão do melhor local para chute no gol e outro para o deslocamento do robô para as proximidades da bola.

Para implementar esses controladores, foi utilizada uma biblioteca chamada **jFuzzyLogic** – pacote *open source* para lógica fuzzy escrito em Java que implementa a linguagem de controle fuzzy [3] .

A Linguagem de Controle Fuzzy (FCL – *Fuzzy Language Control*) é uma linguagem empregada para implementar lógica fuzzy, especialmente controle fuzzy [2].

2.1 Controlador para Melhor Posição de Chute a Gol

Esse controlador tem por objetivo encontrar o melhor ponto no gol para chutar, ou seja, o ponto cuja a probabilidade de marcar seja a maior possível [6].

As variáveis de entrada são: posição do goleiro no eixo y e a posição do atacante no eixo y. Os termos lingüísticos utilizados são apresentados nas figuras 1 e 2 respectivamente.

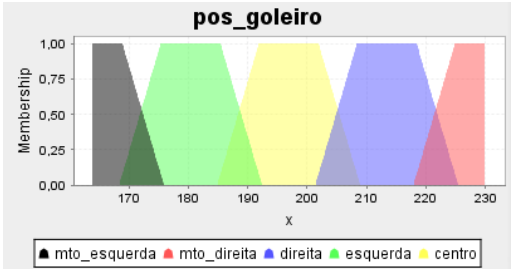

Figura 1: Conjunto Fuzzy da variável de entrada posição do goleiro

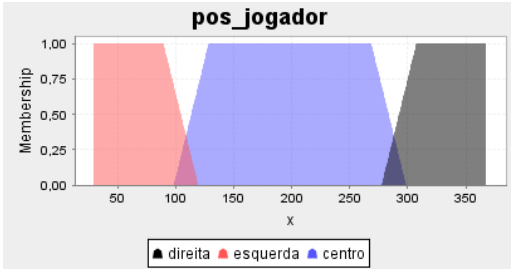

Figura 2: Conjunto Fuzzy da variável de entrada posição jogador

A variável de saída é a posição do chute no eixo y. Os termos lingüísticos utilizados são apresentados na figura 3.

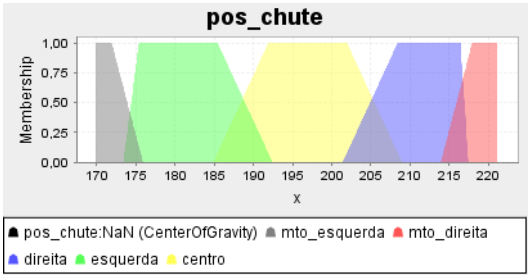

Figura 3: Conjunto Fuzzy da variável de saída posição de chute

O conjunto de regras desse controlador foram criadas combinando as variáveis de entrada visando a posição mais conveniente para o chute. Todas as regras possuem o mesmo peso. Algumas regras são apresentadas na tabela 1. Para maiores detalhes consulte o apêndice A.

Posição Goleiro	Posição Jogador	Posição Chute
Muito Esquerda	Direita	Direita
Esquerda	Esquerda	Direita
Centro	Centro	Esquerda
Direita	Direita	Esquerda
Muito Direita	Esquerda	Esquerda

Tabela 1: Regras de inferência da posição de chute

2.2 Controlador para posicionamento próximo a bola

Esse controlador tem por objetivo, permitir que o robô atacante se posicione nas proximidades da bola.

Para simplificar a montagem das regras, esse controlador foi dividido em dois: um para o eixo x e outro para o eixo y.

Ao final as saídas dos dois controladores formam uma coordenada no campo onde o robô deve se deslocar.

O conjunto de regras desse controlador foram desenvolvidas combinando as variáveis de entrada. Para maiores detalhes consulte os apêndices B e C.

2.2.1 Controlador para posicionamento no eixo x

As variáveis de entrada são: posição da bola no eixo x e posição do atacante no eixo x. Os termos lingüísticos utilizados são apresentados nas figuras 4 e 5 respectivamente.

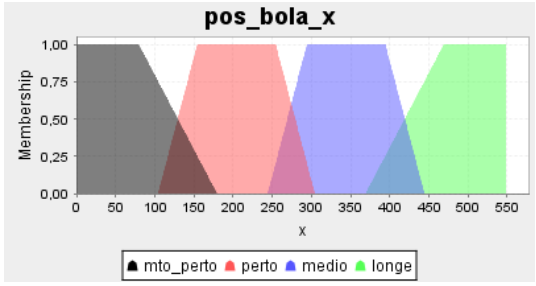

Figura 4: Conjunto Fuzzy da variável de entrada posição x da bola

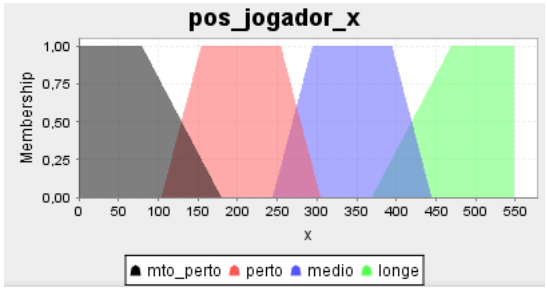

Figura 5: Conjunto Fuzzy da variável de entrada posição x do jogador

A variável de saída é a posição no eixo x que o atacante deve se dirigir. Os termos lingüísticos utilizados são apresentados na figura 6.

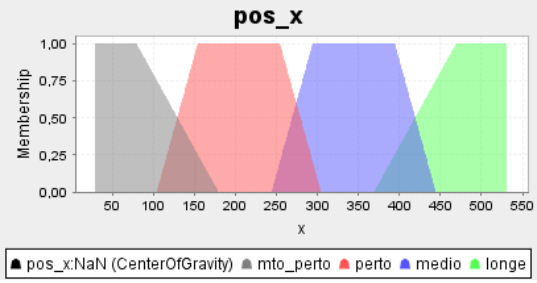

Figura 6: Conjunto Fuzzy da variável de saída posição x do jogador

2.2.2 Controlador para posicionamento no eixo y

As variáveis de entrada são: posição da bola no eixo y e posição do atacante no eixo y. Os termos lingüísticos utilizados são apresentados nas figuras 7 e 8 respectivamente.

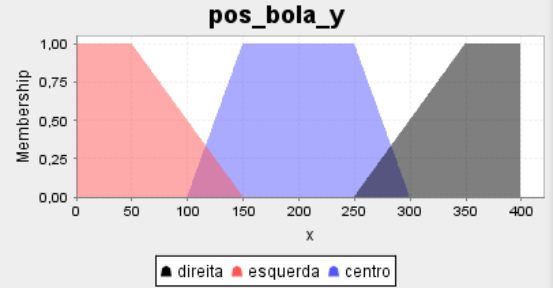

Figura 7: Conjunto Fuzzy da variável de entrada posição y da bola

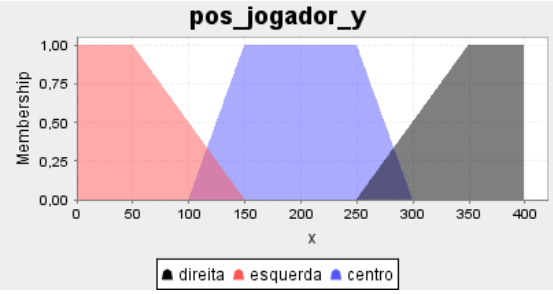

Figura 8: Conjunto Fuzzy da variável de entrada posição y do jogador

A variável de saída é a posição no eixo y que o atacante deve se dirigir. Os termos lingüísticos utilizados são apresentados na figura 9.

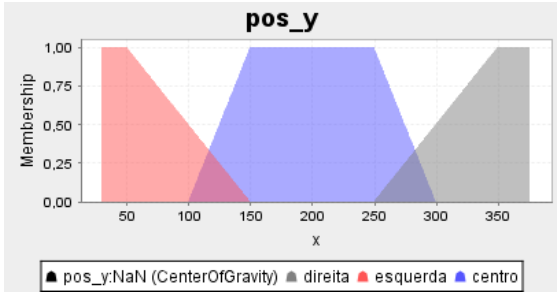

Figura 9: Conjunto Fuzzy da variável de saída posição y do jogador

3 Redes Neurais

Pode-se afirmar que as redes neurais artificiais são uma maneira de abordar a solução de problemas empregando inteligência artificial. A medida que uma rede neural aprende um conceito, ela é capaz de responder a conceitos similares que não foram aprendidos [5].

Partindo desse conceito, uma rede neural simples foi montada para decidir quando o jogador deve chutar a bola. Ela é composta por dois neurônios (do tipo *Perceptron*) na entrada e um neurônio da saída.

Como entrada essa rede recebe a posição x do jogador e a posição y do jogador. Se este estiver próximo ao gol ele deve chutar, caso contrário apenas empurra a bola.

Essa rede foi implementada com o auxílio de uma biblioteca chamada **Neuroph** – um *framework open source* de redes neurais em Java utilizado para desenvolver algumas arquiteturas comuns de redes neurais [4].

Essa biblioteca facilita a implementação da rede neural pois possui uma interface gráfica onde é possível montar a rede, treiná-la e salvá-la em um arquivo que é posteriormente importado no código [4].

4 Métodos de Movimentação

No Tewnta, o robô se movimenta a partir de uma força que é definida a ele e para quando não há mais força sendo aplicada e acaba seu momento. Não existe, nativamente, um método que ao madar ele ir para um ponto com coordenadas (x,y), garanta que ele vá para esse ponto.

Para contornar essa situação foi implementado um algoritmo que, dada a posição atual e um ponto de chegada desejado, calcula a resposta que o robô deve ter

para se aproximar do ponto sem passar deste, por conta de seu momento. Esta lógica foi feita com um controlador PID, pois notou-se comportamento característico à resposta ao degrau, podendo assim modelar o sistema como sendo uma função de transferência (apêndice D).

Para o robô rotacionar para a direção desejada nota-se o mesmo comportamento. Mas, por se tratar de uma movimentação mais simples, com entrada de dados controlada (de -180° a +180°), não foi necessário uso dos reguladores integrador e derivativo, ficando um controlador com apenas o ganho proporcional.

Estes métodos são de suma importância para garantir a precisão dos sistemas de decisão, pois a posição do robô é parte crítica nas decisões seguintes, como pra onde chutar, ou se deve se deslocar para escolher um ponto melhor para chute.

O código do goleiro utilizado para os testes pode ser encontrado no apêndice E.

5 Conclusão

Dado o tempo de execução deste projeto, acredita-se que todos os objetivos foram cumpridos com sucesso, na medida em que o atacante consegue rastrear a bola em qualquer ponto do campo, buscá-la e chutar para o gol com uma porcentagem de acerto bem alta.

Para trabalhos futuros fica a sugestão de fazer com que uma rede neural ajuste as funções de pertinência do controlador de chute a gol e que a rede neural que decide o momento do chute sejam treinadas durante o jogo.

6 Referências

[1] TEWNTA. *Robocup Small Size League (SSL) F180 Simulator.* Disponível em: <http://code.google.com/p/tewnta/>.

[2] INTERNACIONAL ELECTROTECHNICAL COMMISION. *Part 7 – Fuzzy Control Programming*. Disponível em: <http://www.fuzzytech.com/binaries/ieccd1.pdf>.

[3] CINGOLANI, P. *jFuzzyLogic.* Disponível em: <http://jfuzzylogic.sourceforge.net/html/index.html> .

[4] NEUROPH PROJECT. *Java Neural Network Framework.* Disponível em: <http://neuroph.sourceforge.net/about_project.html>.

[5] BARRETO, J. M. *Introdução às Redes Neurais Artificiais.* Disponível em: <http://www.inf.ufsc.br/~barreto/tutoriais/Survey.pdf>. Florianópolis, 2002.

[6] SILVA, H. da L. & SIMÕES, M. & ARAGÃO, H. G. *Desenvolvimento de controladores fuzzy para robôs jogadores de futebol simulado do tipo atacante.* Disponível em: <https://intranet.dcc.ufba.br/pastas/mecateam/material_de_estudo/27069.pdf>. Bahia.

Apêndice A – Descrição do Controlador Fuzzy para Chute a Gol em FCL

```
//Definicao do bloco
FUNCTION_BLOCK pos_chute

// Define as variáveis de entrada
VAR_INPUT
    pos_goleiro : REAL;
    pos_jogador : REAL;
END_VAR

// Define a variavel de saida
VAR_OUTPUT
    pos_chute : REAL;
END_VAR

// Fuzzificando a variável de entrada 'pos_goleiro' (164 - 230)
FUZZIFY pos_goleiro
    TERM mto_esquerda := (164,1) (169,1) (176, 0);
    TERM esquerda := (168.5,0) (175.5,1) (185.5, 1) (192.5, 0);
    TERM centro := (185,0) (192,1) (202,1) (209,0);
    TERM direita := (201.5, 0) (208.5, 1) (218.5, 1) (225.5, 0);
    TERM mto_direita := (218, 0) (225, 1) (230, 1);
END_FUZZIFY

//Fuzzificando a variável de entrada 'pos_jogador' (30 - 368)
FUZZIFY pos_jogador
    TERM esquerda := (30, 1) (90, 1) (120, 0);
    TERM centro := (99, 0) (129,1) (269, 1) (299, 0);
    TERM direita :=(278,0) (308, 1) (368, 1);
END_FUZZIFY



// Defuzzificando a variável de saída 'pos_chute'
DEFUZZIFY pos_chute
    TERM mto_esquerda := (170,1) (172,1) (176, 0);
    TERM esquerda := (173.5,0) (175.5,1) (185.5, 1) (192.5, 0);
    TERM centro := (185,0) (192,1) (202,1) (209,0);
    TERM direita := (201.5, 0) (208.5, 1) (216.5, 1) (217.5, 0);
    TERM mto_direita := (214, 0) (218, 1) (220, 1);
    // Utiliza 'Centro de Gravidade' como método de defuzzificação
    METHOD : COG;
    // Valor default é 197 (centro) (se nenhuma regra for ativada)
    DEFAULT := 197;
END_DEFUZZIFY
```

```
RULEBLOCK No1
    // Use 'min' for 'and' (also implicit use 'max'
    // for 'or' to fulfill DeMorgan's Law)
    AND : MIN;
    // Use 'min' activation method
    ACT : MIN;
    // Use 'max' accumulation method
    ACCU : MAX;

    //goleiro mto_esquerda
    RULE 1 : IF pos_goleiro IS mto_esquerda AND pos_jogador IS esquerda
                THEN pos_chute IS direita;
    RULE 2 : IF pos_goleiro IS mto_esquerda AND pos_jogador IS centro
                THEN pos_chute IS direita;
    RULE 3 : IF pos_goleiro IS mto_esquerda AND pos_jogador IS direita
                THEN pos_chute IS direita;
    //goleiro esquerda
    RULE 4 : IF pos_goleiro IS esquerda AND pos_jogador IS esquerda
                THEN pos_chute IS direita;
    RULE 5 : IF pos_goleiro IS esquerda AND pos_jogador IS centro
                THEN pos_chute IS direita;
    RULE 6 : IF pos_goleiro IS esquerda AND pos_jogador IS direita
                THEN pos_chute IS direita;
    //goleiro centro
    RULE 7 : IF pos_goleiro IS centro AND pos_jogador IS esquerda
                THEN pos_chute IS esquerda;
    RULE 8 : IF pos_goleiro IS centro AND pos_jogador IS centro
                THEN pos_chute IS esquerda;
    RULE 9 : IF pos_goleiro IS centro AND pos_jogador IS direita
                THEN pos_chute IS direita;
    //goleiro direita
    RULE 10 : IF pos_goleiro IS direita AND pos_jogador IS esquerda
                THEN pos_chute IS esquerda;
    RULE 11 : IF pos_goleiro IS direita AND pos_jogador IS centro
                THEN pos_chute IS esquerda;
    RULE 12 : IF pos_goleiro IS direita AND pos_jogador IS direita
                THEN pos_chute IS esquerda;
    //goleiro mto_direita
    RULE 13 : IF pos_goleiro IS direita AND pos_jogador IS esquerda
                THEN pos_chute IS esquerda;
    RULE 14 : IF pos_goleiro IS mto_direita AND pos_jogador IS centro
                THEN pos_chute IS esquerda;
    RULE 15 : IF pos_goleiro IS mto_direita AND pos_jogador IS direita
                THEN pos_chute IS esquerda;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Apêndice B – Descrição do Controlador Fuzzy para Posicionamento no Eixo X em FCL

```
//Definicao do bloco
FUNCTION_BLOCK pos_x

// Define as variáveis de entrada
VAR_INPUT
    pos_bola_x : REAL;
    pos_jogador_x : REAL;
END_VAR

// Define a variavel de saida
VAR_OUTPUT
    pos_x : REAL;
END_VAR

// Fuzzificando a variável de entrada 'pos_bola_x' (0 - 550)
FUZZIFY pos_bola_x
    TERM longe := (370,0) (470,1) (550, 1);
    TERM medio := (245,0) (295,1) (395, 1) (445, 0);
    TERM perto := (105,0) (155,1) (255,1) (305,0);
    TERM mto_perto := (0,1) (80,1) (180, 0);
END_FUZZIFY

//Fuzzificando a variável de entrada 'pos_jogador_x' (0 - 550)
FUZZIFY pos_jogador_x
    TERM longe := (370,0) (470,1) (550, 1);
    TERM medio := (245,0) (295,1) (395, 1) (445, 0);
    TERM perto := (105,0) (155,1) (255,1) (305,0);
    TERM mto_perto := (0,1) (80,1) (180, 0);
END_FUZZIFY

// Defuzzificando a variável de saída 'pos_x'
DEFUZZIFY pos_x
    TERM longe := (370,0) (470,1) (520, 1);
    TERM medio := (245,0) (295,1) (395, 1) (445, 0);
    TERM perto := (105,0) (155,1) (255,1) (305,0);
    TERM mto_perto := (30,1) (80,1) (180, 0);
    // Utiliza 'Centro de Gravidade' como método de defuzzificação
    METHOD : COG;
    // Valor default é 275 (centro) (se nenhuma regra for ativada)
    DEFAULT := 275;
END_DEFUZZIFY
```

```
RULEBLOCK No1
    // Use 'min' for 'and' (also implicit use 'max'
    // for 'or' to fulfill DeMorgan's Law)
    AND : MIN;
    // Use 'min' activation method
    ACT : MIN;
    // Use 'max' accumulation method
    ACCU : MAX;

    //bola longe
    RULE 1 : IF pos_bola_x IS longe AND pos_jogador_x IS longe
                THEN pos_x IS longe;
    RULE 2 : IF pos_bola_x IS longe AND pos_jogador_x IS medio
                THEN pos_x IS longe;
    RULE 3 : IF pos_bola_x IS longe AND pos_jogador_x IS perto
                THEN pos_x IS longe;
    RULE 4 : IF pos_bola_x IS longe AND pos_jogador_x IS mto_perto
                THEN pos_x IS longe;
    //bola medio
    RULE 5 : IF pos_bola_x IS medio AND pos_jogador_x IS longe
                THEN pos_x IS medio;
    RULE 6 : IF pos_bola_x IS medio AND pos_jogador_x IS medio
                THEN pos_x IS medio;
    RULE 7 : IF pos_bola_x IS medio AND pos_jogador_x IS perto
                THEN pos_x IS medio;
    RULE 8 : IF pos_bola_x IS medio AND pos_jogador_x IS mto_perto
                THEN pos_x IS medio;
    //bola perto
    RULE 9 : IF pos_bola_x IS perto AND pos_jogador_x IS longe
                THEN pos_x IS perto;
    RULE 10 : IF pos_bola_x IS perto AND pos_jogador_x IS medio
                THEN pos_x IS perto;
    RULE 11 : IF pos_bola_x IS perto AND pos_jogador_x IS perto
                THEN pos_x IS perto;
    RULE 12 : IF pos_bola_x IS perto AND pos_jogador_x IS mto_perto
                THEN pos_x IS perto;
    //bola mto_perto
    RULE 13 : IF pos_bola_x IS mto_perto AND pos_jogador_x IS longe
                THEN pos_x IS mto_perto;
    RULE 14 : IF pos_bola_x IS mto_perto AND pos_jogador_x IS medio
                THEN pos_x IS mto_perto;
    RULE 15 : IF pos_bola_x IS mto_perto AND pos_jogador_x IS perto
```

```
                THEN pos_x IS mto_perto;
    RULE 16 : IF pos_bola_x IS mto_perto AND pos_jogador_x IS mto_perto
                THEN pos_x IS mto_perto;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Apêndice C – Descrição do Controlador Fuzzy para Posicionamento no Eixo Y em FCL

```
//Definicao do bloco
FUNCTION_BLOCK pos_y

// Define as variáveis de entrada
VAR_INPUT
    pos_bola_y : REAL;
    pos_jogador_y : REAL;
END_VAR

// Define a variavel de saida
VAR_OUTPUT
    pos_y : REAL;
END_VAR

// Fuzzificando a variável de entrada 'pos_bola_y' (0 - 400)
FUZZIFY pos_bola_y
    TERM esquerda := (0,1) (50,1) (150, 0);
    TERM centro := (100,0) (150,1) (250, 1) (300, 0);
    TERM direita := (250,0) (350,1) (400,1);
END_FUZZIFY

//Fuzzificando a variável de entrada 'pos_jogador_y' (0 - 400)
FUZZIFY pos_jogador_y
     TERM esquerda := (0,1) (50,1) (150, 0);
    TERM centro := (100,0) (150,1) (250, 1) (300, 0);
    TERM direita := (250,0) (350,1) (400,1);
END_FUZZIFY

// Defuzzificando a variável de saída 'pos_y' (30-368)
DEFUZZIFY pos_y
     TERM esquerda := (30,1) (50,1) (150, 0);
    TERM centro := (100,0) (150,1) (250, 1) (300, 0);
    TERM direita := (250,0) (350,1) (368,1);
    // Utiliza 'Centro de Gravidade' como método de defuzzificação
    METHOD : COG;
    // Valor default é 200 (centro) (se nenhuma regra for ativada)
    DEFAULT := 200;
END_DEFUZZIFY

RULEBLOCK No1
```

```
    // Use 'min' for 'and' (also implicit use 'max'
    // for 'or' to fulfill DeMorgan's Law)
    AND : MIN;
    // Use 'min' activation method
    ACT : MIN;
    // Use 'max' accumulation method
    ACCU : MAX;

    //bola esquerda
    RULE 1 : IF pos_bola_y IS esquerda AND pos_jogador_y IS esquerda
                THEN pos_y IS esquerda;
    RULE 2 : IF pos_bola_y IS esquerda AND pos_jogador_y IS centro
                THEN pos_y IS esquerda;
    RULE 3 : IF pos_bola_y IS esquerda AND pos_jogador_y IS direita
                THEN pos_y IS esquerda;

//bola centro
    RULE 4 : IF pos_bola_y IS centro AND pos_jogador_y IS esquerda
                THEN pos_y IS centro;
    RULE 5 : IF pos_bola_y IS centro AND pos_jogador_y IS centro
                THEN pos_y IS centro;
    RULE 6 : IF pos_bola_y IS centro AND pos_jogador_y IS direita
                THEN pos_y IS centro;
//bola direita
    RULE 1 : IF pos_bola_y IS direita AND pos_jogador_y IS esquerda
                THEN pos_y IS direita;
    RULE 2 : IF pos_bola_y IS direita AND pos_jogador_y IS centro
                THEN pos_y IS direita;
    RULE 3 : IF pos_bola_y IS direita AND pos_jogador_y IS direita
                THEN pos_y IS direita;
END_RULEBLOCK



END_FUNCTION_BLOCK
```

Apêndice D – Código do Controlador PID

```
/**
     * Método que faz com que o robo vá para uma posição desejada
     * @param cliente Cliente do simulador
     * @param j jogador
     * @throws Exception
     */
    public static void irLinhaReta(Player cliente, Jogador j) throws
                                                                    Exception {

        RobotInformation p1 = cliente.getPlayerInformation(j.getNome());
        //posicao do jogador
        Point posJog = p1.getPosition();
        //posição desejada
        Point p = j.getPosicaoDesejada();

        System.out.println("Posição: " + posJog.getY() + " - " + p.getY());

        //se na estiver no ponto desejado
        if (posJog != p) {

            //controle PID

            //diferença entre a posicao desejada e a atual
            Double rX = p.getX() - posJog.getX();
            Double rY = p.getY() - posJog.getY();

            //zera os erros mínimos
            if (Math.abs(rX) < ERRO_MIN) {
                rX = 0.0;
            }
            if (Math.abs(rY) < ERRO_MIN) {
                rY = 0.0;
            }

            //Integral
            double iX = dt * kI * ((rX + j.getXErro()) / 2 + j.getiXAnt());
            double iY = dt * kI * ((rY + j.getYErro()) / 2 + j.getiYAnt());

            //Proporcional
            double pX = rX * kP;
            double pY = rY * kP;

            //Derivativo
            double dX = kD * (rX - j.getXErro()) / dt;
```

```
        double dY = kD * (rY - j.getYErro()) / dt;

        //velocidade em x
        double vX = pX + iX + dX;
        //velocidade em y
        double vY = pY + iY + dY;

        //redefine os atributos do jogador
        j.setvXAnterior(vX);
        j.setvYAnterior(vY);

        j.setiYAnt((rY + j.getYErro()) / 2 + j.getiYAnt());
        j.setiXAnt((rX + j.getXErro()) / 2 + j.getiXAnt());

        j.setXErro(rX);
        j.setYErro(rY);

        //define os vetores velocidade do jogador
        cliente.setPlayerVelocity(j.getNome(), vX, vY);
        System.out.println("Vx: " + vX);
        System.out.println("Vy: " + vY);
    }
}
```

Apêndice E – Código do Goleiro Utilizado para os Testes

```
/**
  * Método que modela as atitudes do goleiro
  * @throws Exception
  */
private static void goleiro() throws Exception {

     cliente.setPlayerDribble("Goleiro", Boolean.TRUE);

     //posicao da bola
     Point posBola = cliente.getBallInformation().getPosition();

     //posicao do goleiro
     Point posGole = cliente.getPlayerInformation("Goleiro").getPosition();

     //cria um objeto jogador com os atributos de goleiro
     Jogador goleiro = new Jogador(posGole.getX(), posGole.getY());
     goleiro.setNome("Goleiro");
     goleiro.setPosicaoDesejada(new Point((Double) posGole.getX(),
                                          (Double) posGole.getY()));

     //se a bola estiver acima da posicao do goleiro
     if (posBola.getY() > posGole.getY()) {

         //se a bola estiver acima da trave do gol
         if (posBola.getY() > 230) {
             //goleiro fica na trave
             goleiro.setPosicaoDesejada(new Point(40.0, 229.0));
             MetodosAux.irLinhaReta(cliente, goleiro);

             //se a bola estiver dentro dos limites do gol
         } else {

             //goleiro se posiciona conforme bola
             goleiro.setPosicaoDesejada(new Point(40.0, posBola.getY()));
             MetodosAux.irLinhaReta(cliente, goleiro);
         }

          //se a bola estiver abaixo da posicao do goleiro
     } else if (posBola.getY() <= posGole.getY()) {

         //se a bola estiver abaixo da trave do gol
         if (posBola.getY() < 170) {

             //goleiro fica na trave
```

```
            goleiro.setPosicaoDesejada(new Point(40.0, 171.0));
            MetodosAux.irLinhaReta(cliente, goleiro);

            //se a bola estiver dentro dos limites do gol
        } else {
            //goleiro se posiciona conforme bola
            goleiro.setPosicaoDesejada(new Point(40.0, posBola.getY()));
            MetodosAux.irLinhaReta(cliente, goleiro);
        }
    }

}
```