Dhiego Cristiano Oliveira da Silva Sad

Sistema de detecção e controle para futebol de robôs com processamento remoto

Juiz de Fora - MG, Brasil

8 de dezembro de 2010

Dhiego Cristiano Oliveira da Silva Sad

Sistema de detecção e controle para futebol de robôs com processamento remoto

Monografia apresentada para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade Federal de Juiz de Fora.

Orientador:
Marcelo Bernardes Vieira

Co-orientador:
Marcelo Lobosco

Instituto de Ciências Exatas
Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora

Juiz de Fora - MG, Brasil

8 de dezembro de 2010

Monografia de Projeto Final de Graduação sob o título *"Sistema de detecção e controle para futebol de robôs com processamento remoto"*, defendida por Dhiego Cristiano Oliveira da Silva Sad e aprovada em 8 de dezembro de 2010, em Juiz de Fora, Estado de Minas Gerais, pela banca examinadora constituída pelos professores:

Prof. Dr. Marcelo Bernardes Vieira
Orientador

Prof. Dr. Marcelo Lobosco
Universidade Federal de Juiz de Fora

Prof. Dr. André Marcato
Universidade Federal de Juiz de Fora

Prof. Dr. Rodrigo Luis de Souza da Silva
Universidade Federal de Juiz de Fora

Agradecimentos

Agradeço primeiramente aos meus pais e à Karoline, minha namorada, pelo total apoio e dedicação em todos os passos desta caminhada.

Aos meus colegas do Grupo de Computação Gráfica e principalmente ao Eder e ao Paiva, por colaborarem no desenvolvimento do sistema proposto neste trabalho.

Também sou muito grato aos orientadores desse projeto pelo tempo e paciência dedicados.

Finalmente, agradeço à Fapemig pelo auxílio financeiro e pelos equipamentos disponibilizados.

Resumo

Este trabalho apresenta métodos para resolver problemas de segmentação em imagens e controle em tempo real de robôs, usando processamento remoto. Uma máquina tem a função de fazer todo o processamento de imagem, enquanto outra máquina tem como objetivo fazer todo o cálculo da inteligência artificial.

Serão abordados os fundamentos matemáticos do filtro de Shen-Castan, para detecção de bordas em uma imagem. Abordaremos também sobre a transformada de Hough, para detecção dos centros de cada círculo na imagem, possibilitando que os robôs e a bola sejam detectados. Por último, será feito uma abordagem de como é feita a comunicação entre duas máquinas na rede para transferência de pacotes.

Cada uma dessas abordagens foram implementadas formando um sistema capaz de reconhecer os robôs, a bola e o campo em tempo real. Esse sistema resultante fornece um ambiente adequado para controlar os robôs de maneira precisa e robusta.

Palavras-Chave: Filtro de Shen-Castan, Transformada de Hough, Protocolo de Comunicação.

Abstract

This work presents methods for solving problems in image segmentation and real-time control of robots, using remote processing. A machine has to do all image processing, while another machine makes the calculation of all artificial intelligence.

The Shen-Castan filter mathematical foundations will be approached for the detection of edges in an image. We will also use Hough transform to detect the centers of each circle in the image, allowing the robots and the ball to be detected. Finally, it makes an approach on how well the communication between two machines on the network works to transfer the packages.

Each of these approaches have been implemented into a system capable of recognizing the robots, the ball and the field in real time. This resulting system provides a suitable environment to control the robots with precision and robustness.

Keywords: Shen-Castan filter, Hough transform, Communication protocol.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O futebol de robôs é alvo de pesquisas que levaram ao desenvolvimento de áreas como Visão Computacional, Inteligência Artificial e Robótica Inteligente. Apesar de aparentemente ser uma simples "diversão", a competição entre robôs visa o aperfeiçoamento do controle individual de robôs móveis e da coordenação entre múltiplos robôs, além do aprendizado de diversas técnicas usadas na área científica.

A competição ocorre utilizando aproximadamente as regras do futebol. Basicamente, dois times de robôs se enfrentam em um campo, três robôs para cada lado, sendo diferenciados pelas cores em cima de cada um. Cada time deve ser totalmente autônomo, ou seja, nenhuma intervenção humana é permitida após o início de uma partida (FIRA, 2010).

Utilizando-se desta aplicação, pode-se fazer a avaliação de várias técnicas, como a filtragem de imagens, calibração de cores, detecção de bordas, reconhecimento de padrões, paralelismo, processamento remoto, dentre outros.

1.1 Definição do Problema

Este trabalho trata principalmente dos problemas da área de visão computacional, como a detecção incompleta de objetos de interesse na cena e a redução do ruído no processo de aquisição de imagens. São utilizadas técnicas de filtragem de imagens e calibração de cores para que seja possível diminuir esses ruídos gerados. Pode-se reduzir o impacto da falha de detecção de bordas, problema crucial em visão computacional, através de uma transformada que aumenta a robustez da identificação de objetos de interesse.

Os objetos de interesse que devem ser detectados e analisados são as coordenadas dos robôs e da bola, assim como reconhecer os padrões do campo de futebol. Todas essas informações serão processadas em tempo real, para que se possa ter um sistema competitivo e eficiente. Todo o processamento foi dividido em etapas numa *pipeline*, e

toda a *Inteligência Artificial* (IA) é processada remotamente, diminuindo o tempo de resposta do sistema.

Para que os robôs atendam de forma precisa aos comandos do sistema de IA, como por exemplo, percorrer uma determinada trajetória ou girar um determinado número de graus, faz-se necessário um controle refinado dos mesmos. Este controle é definido pelo cálculo da IA e ajustado através do controle baseado em *Proportional Integral Derivative* (PID).

1.2 Objetivos

O objetivo deste trabalho é pesquisar e desenvolver métodos computacionais e aplicá-los, a fim de se obter resultados satisfatórios no controle em tempo real de robôs. Mais especificamente, este trabalho trata de:

- Aplicação do filtro de Shen-Castan para detecção de bordas;
- Aplicação da transformada de Hough para detecção dos robôs;
- Processamento remoto de IA.

Estes temas são complementares ao trabalho de Perez (2009) que trata dos seguintes problemas:

- Calibração de cor;
- Reconhecimento de padrões no campo;
- Aplicação do filtro de Kalman para correção dos ruídos;
- Controle dos robôs em tempo real usando PID.

2 Fundamentos

2.1 Filtros Ótimos

Os filtros derivativos são usados para detectar e extrair informações dos objetos de interesse em uma imagem, como por exemplo, suas bordas.

Uma borda é definida por uma mudança no nível de cinza, quando ocorre uma descontinuidade na intensidade, ou quando o gradiente da imagem tem uma variação muito grande. Um operador de derivada que seja sensível a estas mudanças operará como um detector de bordas. A interpretação de uma derivada neste caso, seria a taxa de mudança dos níveis de cinza em uma imagem, que é maior perto das bordas e menor em áreas constantes. Pegando os valores da intensidade da imagem e achando os pontos onde a derivada é um ponto de máximo, têm-se suas bordas.

A detecção de bordas pode ser o resultado final de um sistema ou apenas um pré-processamento para um passo posterior. É crucial para um sistema de Visão Computacional que esta detecção seja eficiente e confiável.

Portanto, para conseguir detectar bordas em imagens cuja grandeza é a intensidade da luz, é necessário fazer uma diferenciação na imagem. Logo, quando isso é feito, todas as variações dos níveis de cinza são detectadas e, por isso, há uma grande suscetibilidade a ruídos.

Para diminuir esses ruídos, deve-se suavizar a imagem antes da detecção. Porém, dependendo do operador diferencial usado, teremos bordas diferentes. Logo, não há operador diferencial que possua um bom desempenho em diferentes situações (ZIOU; TABBONE, 1998).

Consequentemente, uma variedade de detectores de bordas têm sido desenvolvidos visando diferentes propósitos, com formulações matemáticas diferenciadas e com propriedades algorítmicas distintas.

Segundo Canny (1986), um filtro ótimo de detecção de arestas precisa atender a três critérios básicos.

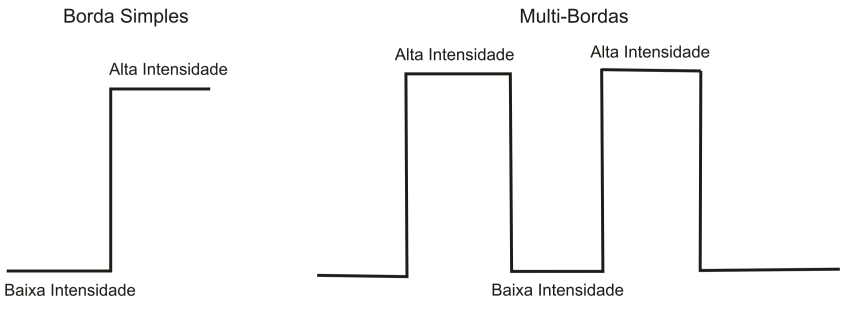
O primeiro deles é denominado Taxa de Erro ou Detecção, consistindo na maximização da *razão sinal/ruído* (SNR). Quanto maior for o SNR, maior é a probabilidade de se detectar as bordas verdadeiras da imagem. Isto define que o detector deve ter baixa probabilidade de falha na detecção de arestas, ou seja, baixa probabilidade de escolher pixels que estão fora das bordas.

O segundo critério especifica que pontos de bordas devem estar bem localizados, isto é, as distâncias entre os pontos extraídos pelo detector e as respectivas posições verdadeiras devem ser minimizadas. Este critério é chamado de *Localização* (L) e é definido como sendo o inverso da distância entre um ponto detectado e a respectiva posição verdadeira.

Por último, o terceiro critério para um filtro ótimo é o da *Unicidade*, onde detector de bordas não deve identificar múltiplas bordas onde existe somente uma, ou seja, deve haver apenas um único pixel de borda encontrada para um único pixel de borda real na imagem (CANNY, 1986).

2.1.1 Filtro de Shen-Castan (ISEF)

O filtro de Shen-Castan (SHEN; CASTAN, 1992), também chamado de *Infinite Symmetric Exponential Filter* - ISEF, é considerado um filtro ótimo para detecção de multi-bordas, pois fornecem uma taxa melhor da *razão sinal-ruído* (SNR) do que o detector de Canny (1986), e apresentam também uma melhor localização (L) de bordas.



(a) (b)

Figura 1: (a) Borda simples e (b) Multi-bordas.

Dado que as imagens são em duas dimensões, é importante considerar mudanças nos níveis de cinza em cada uma das direções $(\vec{x}, \vec{y})$.

Muitos métodos de detecção de bordas têm sido propostos, como por exemplo, o

gradiente de Robert, operadores de Sobel e o Laplaciano (GONZALEZ; WOODS, 1993). Em geral, estes métodos utilizam filtros com núcleo de tamanho reduzido.

O trabalho de Shen e Castan (1992) tomou por base a teoria sobre detecção de bordas desenvolvida por Marr e Hildreth (MARR; HILDRETH, 1980), os quais mostraram que filtros com uma máscara de suavização (filtro passa-baixa) de grande largura são menos sensíveis a ruídos. Partindo dessa ideia, Shen e Castan (1992) imaginaram que o uso de um filtro com núcleo de tamanho infinito implicaria em resultados ainda melhores.

Foi considerada também uma melhor maneira de localizar os pixels da borda, e sugeriram a ação de um segundo filtro que apresentasse uma curva de distribuição com pequeno desvio padrão no centro, em vez de um filtro gaussiano.

A definição de Shen e Castan (1992) para um detector de bordas tem como base um conjunto de filtros de suavização e algumas operações de diferenciação. Eles usaram um processo de otimização com base em três fatores:

- Maximização da resposta do detector a uma borda;
- Minimização da resposta do filtro de suavização a ruídos;
- Minimização da resposta do detector ao ruído.

Shen e Castan (1992) sugerem a minimização do critério infinito normalizado, que em uma dimensão é:

$$C_N^2 = \frac{4 \int_0^\infty f^2(x)dx \cdot \int_0^\infty f'^2(x)dx}{f^4(0)}. \tag{2.1}$$

A Equação (2.1) indica que a função $f(x)$ que minimiza C_N é um filtro de suavização ótimo para um detector de bordas. O filtro ótimo encontrado é:

$$f(x) = \frac{p}{2} e^{-p|x|}, \tag{2.2}$$

sendo p uma constante. O Gráfico (2) representa este filtro.

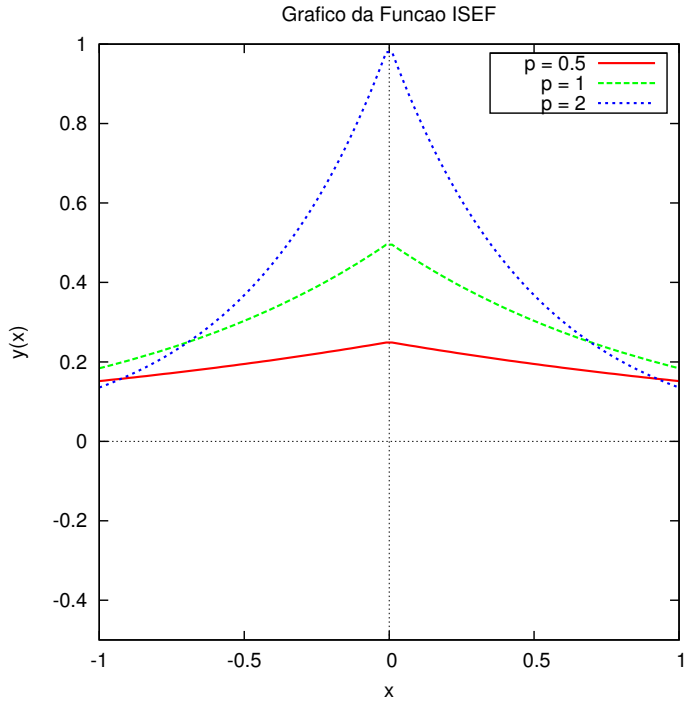


Figura 2: Gráfico da função ISEF unidimensional.

Shen e Castan (1992), afirmam que com este filtro, obtém-se um resultado mais favorável para a razão sinal - ruído e também resulta em uma melhor localização que o filtro de Canny (1986). Isto ocorre, pois Canny (1986) aproxima seu filtro ótimo pela derivada da Gaussiana, enquanto Shen e Castan (1992) usam o filtro ótimo diretamente. No entanto, Shen e Castan (1992) não levam em consideração o critério de respostas múltiplas, sendo possível que o método apresente resultados insatisfatórios para ruídos e bordas deslocadas.

A Equação (2.3) mostra o ISEF para 2 dimensões:

$$f(x, y) = a \cdot e^{-p(|x|+|y|)}, \tag{2.3}$$

sendo a uma constante e $0 < a < 1$ e $p > 0$. O valor do expoente representa a distância, em magnitude, entre x e y. Este filtro poderia ser aplicado para uma imagem, da mesma forma que o filtro Gaussiano, na direção $\vec{x}$ e depois na direção $\vec{y}$, de maneira recursiva.

2.2 Transformada de Hough

A Transformada Hough é um método matemático proposto por Paul Hough em 1962, patenteada pela IBM e reformulada computacionalmente por Duda e Hart (DUDA; HART,

1971). É um método muito eficiente na detecção de linhas e círculos em imagens cujas bordas foram previamente realçadas (MCLAUGHLIN, 1998). É uma transformada que faz uma parametrização de formas geométricas, dado sua localização, em um espaço acumulado (Espaço de Hough). De uma maneira geral, a transformada trabalha em um domínio definido pelos possíveis parâmetros da equação que descreve a forma geométrica.

Temos dois tipos de Transformada de Hough:

- Transformada Clássica: usada para identificar formas geométricas regulares, como retas, círculos, retângulos e outras formas facilmente parametrizáveis.
- Transformada Generalizada: identifica formas bi-dimensionais, com qualquer tamanho e orientação, porém com um alto custo computacional. Esta transformada é muito utilizada em imagens médicas (RODRIGUES et al., 2007).

Uma das principais vantagens de se usar a Transformada de Hough é o fato dela apresentar uma alta tolerância a ruídos. Porém, uma desvantagem desta transformada está na complexidade da curva a ser analisada, ou seja, uma forma geométrica muito complexa exige um número maior de parâmetros, consequentemente o esforço computacional exigido será bem maior.

O princípio básico da Transformada de Hough, consiste em aplicar na imagem uma transformação tal que todos os pontos pertencentes a uma mesma curva sejam mapeados em um único ponto do novo espaço, que no caso seria o *Espaço de Hough* (Fig. 3).

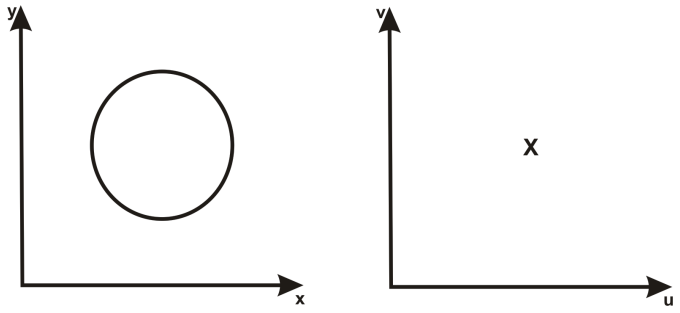


Figura 3: Uma curva qualquer parametrizada no espaço (x,y) , sofre a Transformada de Hough e passa a ser mapeada em um único ponto.

2.2.1 Transformada de Hough Clássica para Detecção de Círculos

Considerando o círculo $\Delta 1$ a seguir, de raio R e centro de coordenadas (xc, yc) (Fig. 4).

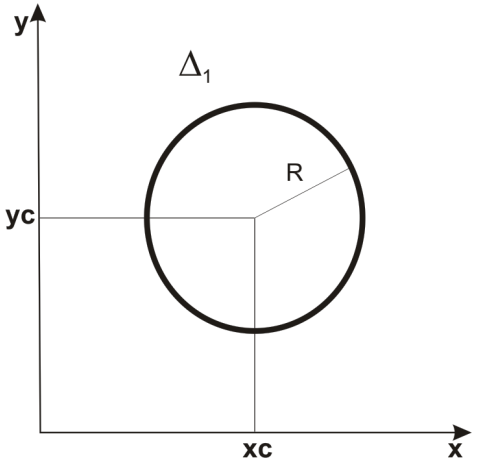


Figura 4: Um círculo $\Delta 1$ de raio R e centro de coordenadas (xc, yc).

Traçando uma tangente t ao círculo $\Delta 1$ e um vetor Gradiente G perpendicular à t, temos (Fig. 5):

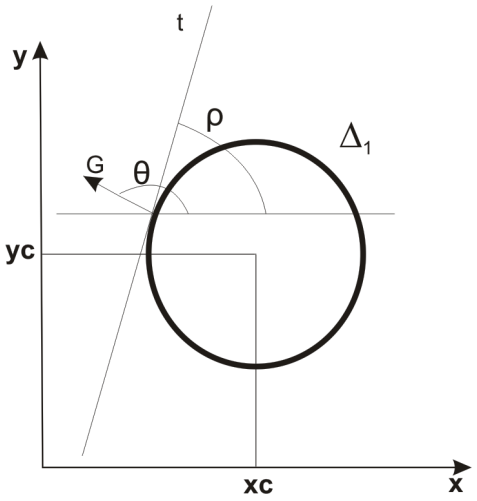


Figura 5: São traçados ao círculo $\Delta 1$ uma tangente t e um vetor gradiente G, perpendicular à t.

Onde θ é o ângulo formado entre o vetor gradiente G e o eixo horizontal $\vec{x}$ e ρ é o ângulo formado entre a tangente t e o eixo horizontal $\vec{x}$.

Da geometria elementar temos que a equação paramétrica do círculo é:

$$(x - xc)^2 + (y - yc)^2 \quad = \quad R^2. \tag{2.4}$$

Aplicando a derivada de primeira ordem, relativa ao gradiente, em (2.4), chegamos a:

$$2(x - xc) + 2(y - yc)\frac{dy}{dx} = 0, \tag{2.5}$$

onde temos que:

$$\frac{dy}{dx} = \tan\rho,$$

e

$$\theta = \frac{\pi}{2} + \rho,$$

ou seja,

$$\rho = \theta - \frac{\pi}{2}.$$

Substituindo temos que:

$$\frac{dy}{dx} = \tan(\theta - \frac{\pi}{2});$$

$$\frac{dy}{dx} = -\frac{1}{\tan\theta}. \tag{2.6}$$

Substituindo (2.6) em (2.5), obtemos:

$$2(x - xc) + 2(y - yc) \cdot (-\frac{1}{\tan\theta}) = 0.$$

Simplificando:

$$(x - xc) - \frac{(y - yc)}{\tan\theta} = 0.$$

Logo:

$$(y - yc) = (x - xc)\tan\theta. \tag{2.7}$$

Substituindo a Equação (2.7) em (2.4), temos:

$$(x - xc)^2 + (x - xc)^2\tan^2\theta = R^2.$$

Isolando $(x - xc)^2$, chegamos a:

$$(x - xc)^2 + [1 + \tan^2 \theta] \quad = \quad R^2. \tag{2.8}$$

Como $\tan^2 \theta = \sin^2 \theta / \cos^2 \theta$, podemos substituir em (2.8), ficando com:

$$(x - xc)^2 \cdot \left[1 + \frac{\sin^2 \theta}{\cos^2 \theta}\right] \quad = \quad R^2;$$

$$(x - xc)^2 \cdot \left[\frac{\cos^2 \theta}{\cos^2 \theta} \cdot 1 + \frac{\sin^2 \theta}{\cos^2 \theta}\right] \quad = \quad R^2;$$

$$(x - xc)^2 \cdot \left[\frac{\cos^2 \theta + \sin^2 \theta}{\cos^2 \theta}\right] \quad = \quad R^2.$$

Como $\cos^2 \theta + \sin^2 \theta = 1$, ficamos com:

$$(x - xc)^2 \cdot \left[\frac{1}{\cos^2 \theta}\right] \quad = \quad R^2;$$

$$(x - xc)^2 = \quad R^2 \cdot \cos^2 \theta.$$

Simplificando:

$$(x - xc) \quad = \quad R \cdot \cos \theta. \tag{2.9}$$

Substituindo (2.9) em (2.7), encontramos:

$$(y - yc) \quad = \quad R \cdot \cos \theta \cdot \tan \theta.$$

Como $\tan \theta = \sin \theta / \cos \theta$, concluímos que:

$$(y - yc) \quad = \quad R \cdot \sin \theta, \tag{2.10}$$

onde x e y definem as coordenadas do ponto no novo espaço de parametrização do círculo, xc e yc definem as coordenadas do centro e R é o raio do mesmo.

Com o *Espaço de Hough* gerado, detectar os possíveis centros xc e yc, com um raio fixo R, se resume a encontrar os pontos de máximos relativos, os pontos mais votados, neste novo espaço.

3 Modelo Computacional

Uma partida de futebol de robôs é realizada a partir de uma câmera posicionada acima do campo, responsável por enviar ao sistema de detecção as imagens capturadas (Fig. 6).

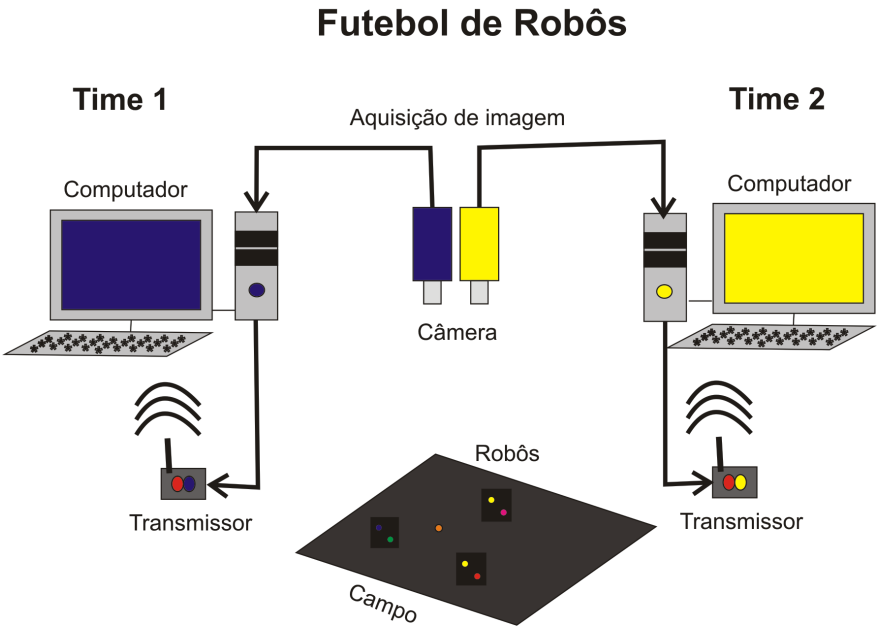


Figura 6: Posicionamento dos equipamentos para a competição.

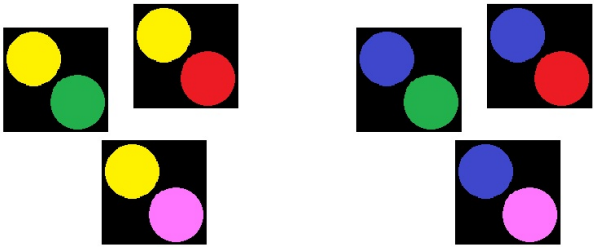
As imagens, chamadas quadros, são capturadas a uma taxa de 120 qps (cento e vinte quadros por segundo) e possuem dimensão de 640 x 480 pixels. Para obtenção das imagens foi utilizada uma câmera modelo BASLER SCA640 - 120fc e uma placa de captura de vídeo IEEE 1394B. O computador utilizado foi um Intel Core 2 Quad Q8300 de 2.50GHZ com 4.00GB de RAM.

O sistema utiliza a biblioteca GCGLIB desenvolvida pelo Grupo de Computação Gráfica, Imagem e Visão da UFJF. Esta biblioteca possui as funções necessárias para a captura e tratamento dos quadros.

O campo utilizado é baseado nas medidas oficiais da *FIRA*, pintado de preto com

linhas brancas. Tanto as medidas, quanto as marcações devem ser reconhecidas pelo sistema de detecção.

A bola utilizada para as partidas deverá ser uma bola de golfe na cor laranja e a dimensão dos robôs deverá ser limitada a 7,5cm x 7,5cm x 7,5cm. São colocadas etiquetas no topo de cada robô, uma de cor azul ou amarela, para identificação do time, e outra de cor verde, magenta ou vermelho para identificação de cada robô. A etiqueta utilizada apresenta a forma de um círculo de raio fixado previamente.



(a) (b)

Figura 7: Modelo de robôs, onde o círculo azul ou círculo amarelo define a cor do time e os círculos verde, magenta e vermelho definem os robôs daquele time.

A partir das imagens capturadas pela câmera, identifica-se os objetos de interesse, tais como, os robôs, a bola e as coordenadas do campo. Após a identificação, é enviado comandos aos robôs, de acordo com a estratégia calculada pelo sistema de IA, para que a partida possa ser realizada (Fig. 8).

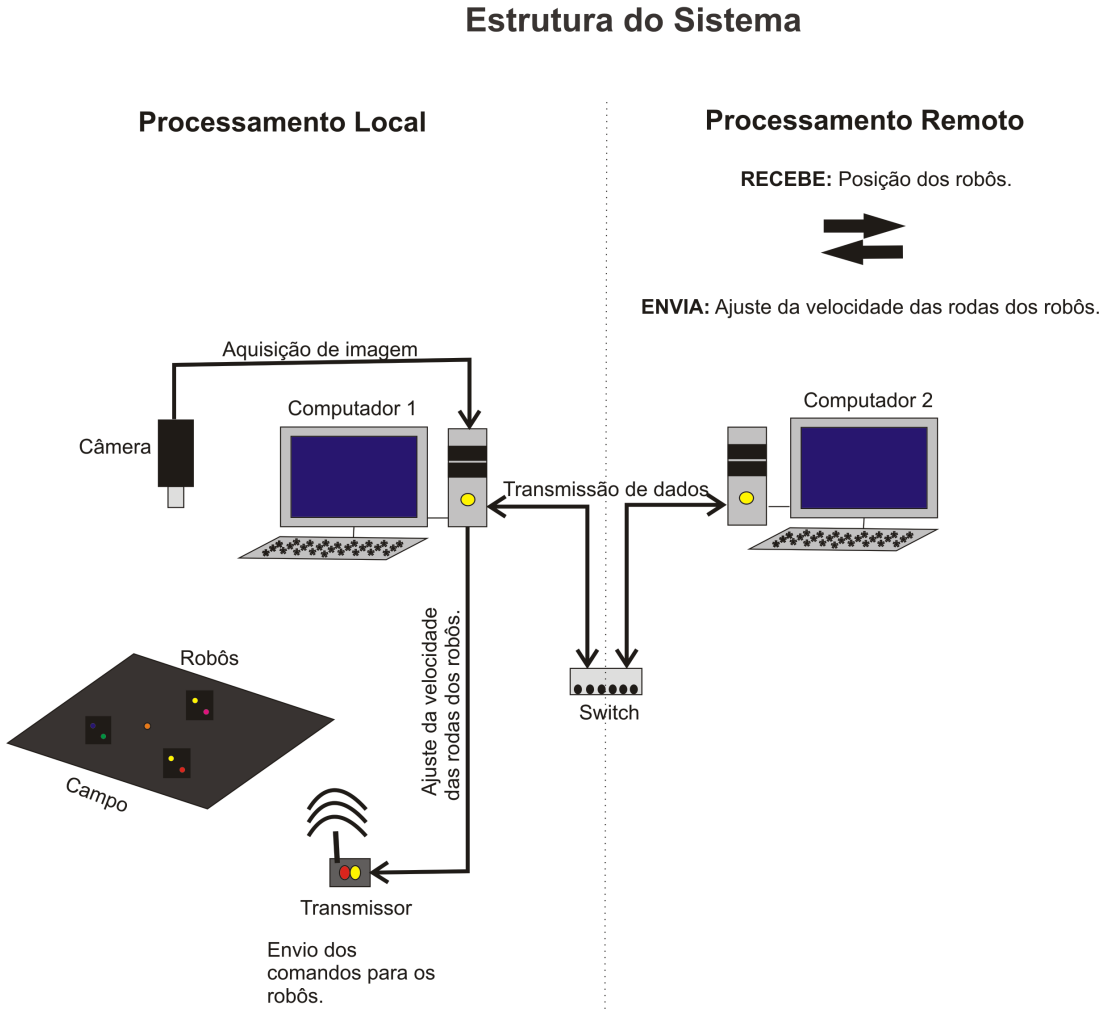


Figura 8: Estrutura do sistema.

Para que o objetivo desse trabalho seja atingido, é necessário o desenvolvimento de um time completo de Futebol de Robôs da categoria Mirosot. A seguir, serão descritas as etapas deste sistema.

3.1 Parametrização e interface com usuário

Antes do início de uma partida, existe uma etapa de inicialização do sistema na qual é criado um objeto gcgVIDEO que vai receber os quadros vindos da câmera e entregá-los ao sistema de detecção.

A partir de uma interface gráfica pode-se configurar o sistema que será utilizado para detectar e controlar os robôs. Nesta interface ajustam-se inicialmente as configurações da câmera, como por exemplo, o tamanho da janela de captura, que no caso é de 640 x 480 pixels e a taxa de captura dessas imagens (120 quadros por segundo). Depois de iniciar a

câmera, é feita uma classificação das cores que serão detectadas durante a partida (a cor da bola e as cores das etiquetas dos robôs). Essas cores também servem para calcular a matriz de calibração de cores, usada para realçar as cores desejadas, diminuindo o ruído gerado pela variação da iluminação. É permitido também mudar as dimensões do campo, caso o usuário não tenha o campo com as medidas oficiais.

Depois de feito os ajustes, carrega-se a *Dynamic Link Library*(DLL) de estratégia usada em cada time. É definido se será usado processamento remoto ou local para o cálculo da estratégia de *IA*, então inicia-se o jogo.

3.2 Pipeline

Na década de 70 surge uma nova técnica chamada pipeline, onde a execução das instruções é dividida em várias fases consecutivas.

Nesta técnica, quando uma instrução termina a execução em estágio, uma próxima instrução pode ser iniciada no mesmo estágio, resultando assim em uma concorrência temporal.

Pode-se definir que pipeline é uma técnica de projeto onde mais de uma instrução de cada vez é processada, sem a necessidade de esperar que uma instrução termine antes de outra começar. Esta técnica é semelhante a uma linha de produção de fábrica, onde cada tarefa passa por diversas fases até sua execução.

Neste trabalho é utilizada esta técnica, para que se possa ter um sistema eficiente, robusto e em tempo real.

Como alguns dos computadores de hoje apresentam 4 núcleos de processamento, o sistema foi dividido em 4 etapas de uma *Pipeline* (Fig. 9).

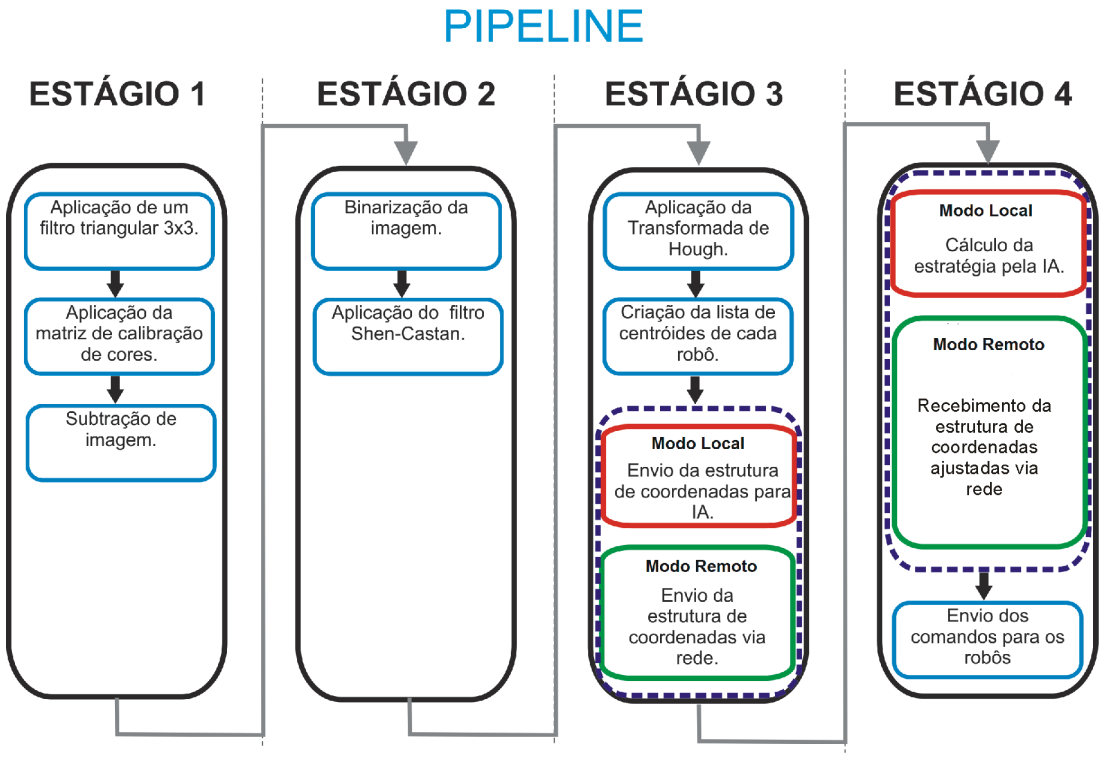


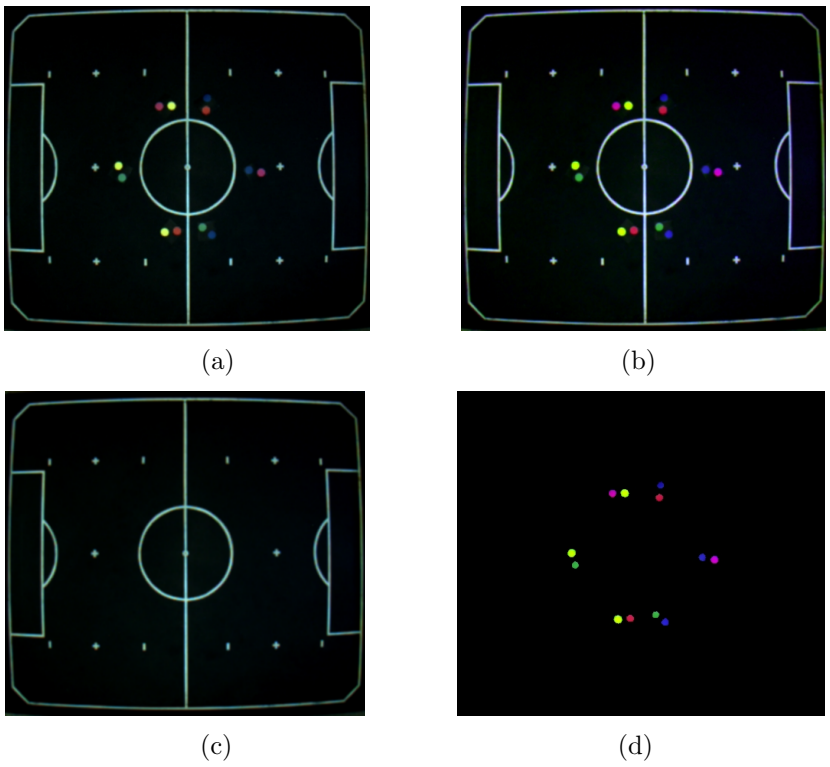
Figura 9: Pipeline do sistema.

3.2.1 Primeira Etapa da Pipeline

Após a captura de um quadro pela câmera, é iniciado uma fase de processamento de imagens. Para isso é aplicado um filtro de passa-baixa triângular 3x3 para suavizar a imagem, diminuindo os ruídos gerados pela câmera. Depois de suavizar a imagem, é necessário fazer o reconhecimento dos robôs e da bola. Inicialmente, aplica-se uma matriz de calibração de cores na imagem suavizada, transformando um conjunto de cores sensíveis à câmera para um conjunto de cores conhecido. Para encontrar a matriz de calibração de cores, é utilizada a matriz pseudo-inversa. A aplicação dessa matriz nas imagens possibilita diminuir a variação de brilho nas cores, aumentando a capacidade de detecção das mesmas (PEREZ, 2009).

Após o realce das cores de interesse, é feito uma subtração da imagem do campo com os robôs, da imagem do campo sem os robôs. É feito isso, para obter apenas uma imagem com os objetos de interesse, que são os robôs e a bola (Fig. 10).

No final desta etapa, a imagem resultante da subtração é passada para a próxima etapa da *Pipeline*.



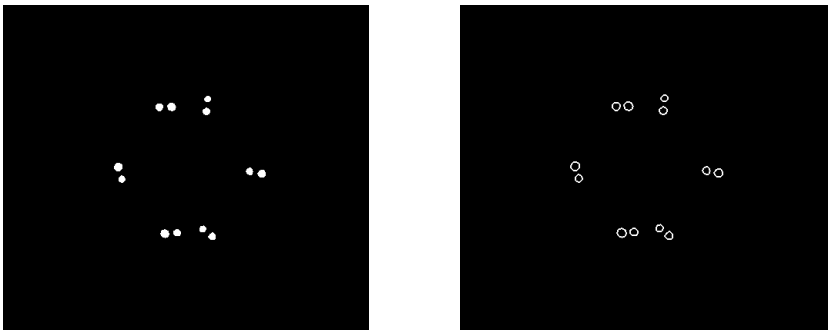
(a) (b)

(c) (d)

Figura 10: Processamento das imagens, onde (a) é a imagem original, (b) é a imagem com as cores realçadas, (c) é a imagem de fundo (d) é o resultado da subtração da imagem de cores realçadas da imagem de fundo.

3.2.2 Segunda Etapa da Pipeline

Nesta etapa, é feita uma binarização na imagem recebida pela etapa anterior. Feito isso, aplica-se um filtro de passa-alta, para extrair as bordas desta imagem. O filtro utilizado é o de Shen-Castan (1992) (Fig. 11).



(a) (b)

Figura 11: (a) Imagem binarizada, (b) é o resultado da aplicação do filtro de Shen-Castan.

3.2.2.1 Aplicação do Filtro de Shen-Castan

Para extrair as bordas da imagem resultante, aplica-se o filtro ótimo (ISEF) de Shen-Castan (1992). Primeiramente, deve-se discretizar a Equação (3.1) do filtro, que em duas dimensões é:

$$f(x,y) = ae^{-p(|x|+|y|)}, \tag{3.1}$$

Reescrevendo a Equação (3.1) na forma discreta temos:

$$f[i,j] = \frac{(1-b)b^{|i|+|j|}}{1+b}, \tag{3.2}$$

onde b é um parâmetro do filtro gerado pela discretização.

Este filtro é convoluído na imagem recursivamente na direção $\vec{x}$, gerando $r[i,j]$

$$y_1[i,j] = \frac{1-b}{1+b} \cdot I[i,j] + by_1[i,j-1]; j = 1 \cdots N; i = 1 \cdots M$$

$$y_2[i,j] = b \cdot \frac{1-b}{1+b} \cdot I[i,j] + by_1[i,j+1]; j = 1 \cdots N; i = 1 \cdots M$$

$$r[i,j] = y_1[i,j] + y_2[i,j+1], \tag{3.3}$$

onde y_1 e y_2 são a imagem resultante da convolução na direção $\vec{x}$.

As condições de fronteira são dadas por:

$$I[i,0] = 0;$$

$$y_1[i,0] = 0; \tag{3.4}$$

$$y_2[i,M+1] = 0.$$

Fazendo a convolução na imagem, desta vez na direção de $\vec{y}$, obtém-se o resultado final do filtro em $y[i,j]$:

$$y_1[i,j] = \frac{1-b}{1+b} \cdot I[i,j] + by_1[i-1,j]; j = 1 \cdots N; i = 1 \cdots M$$

$$y_2[i,j] = b \cdot \frac{1-b}{1+b} \cdot I[i,j] + by_1[i+1,j]; j = 1 \cdots N; i = 1 \cdots M$$

$$y[i,j] = y_1[i,j] + y_2[i+1,j], \tag{3.5}$$

onde y_1 e y_2 são a imagem resultante da convolução na direção $\vec{y}$.

As condições de fronteira são dadas por:

$$I[0,j] = 0;$$

$$y_1[0,j] = 0; \tag{3.6}$$

$$y_2[N+1,j] = 0.$$

O filtro é aplicado em todos os pontos da imagem. Suas bordas serão localizadas pelo zero do laplaciano, processo usado no algoritmo de Marr e Hildreth (MARR; HILDRETH, 1980). Uma aproximação do laplaciano pode ser obtida através da subtração da imagem filtrada (S) da imagem original (I):

$$S[i,j] - I[i,j] \approx \frac{1}{4a^2} I[i,j] * \nabla^2 f(i,j), \tag{3.7}$$

onde a é uma constante e $B = S - I$ é o Laplaciano limitado por banda. A partir desta imagem, uma *imagem binária do Laplaciano* (IBL) é obtida atribuindo o valor 1 para todos os pixels com valores positivo e 0 para os demais. Os pixels candidatos à borda estão na região de fronteira de IBL, que correspondem aos zeros da função.

Uma outra maneira de localizar as bordas da imagem é pelos máximos locais do gradiente, através do cálculo das derivadas de primeira ordem na direção de x e de y.

O detector de Canny usa a derivada da função gaussiana para computar a primeira derivada de uma imagem. Ele é um operador de primeira derivada que suaviza os ruídos e detecta as bordas. É possível combinar os estágios de detecção e suavização em uma simples convolução em 1-D, cada convolução com a primeira derivada da função de Gauss e considerando os picos, ou com a segunda derivada considerando os cruzamentos dos zeros.

Shen-Castan utilizam a mesma idéia, porem utilizam a derivada do filtro ISEF (Fig. 12) em x e em y ao invés da derivada da gaussiana da seguinte maneira:

- Produz derivada 1-D do ISEF para x (ISEFx) e y (ISEFy)
- Faz uma convolução de I com ISEF, gerando Ix e Iy
- Faz uma convolução de Ix com ISEFx e Iy com ISEFy,produzindo I'x e I'y
- Calcula-se a magnitude (M) dessas imagens derivadas:

$$M = \sqrt{I'x^2 + I'y^2}. \tag{3.8}$$

Tem-se como resultado final, uma imagem com suas bordas estraídas.

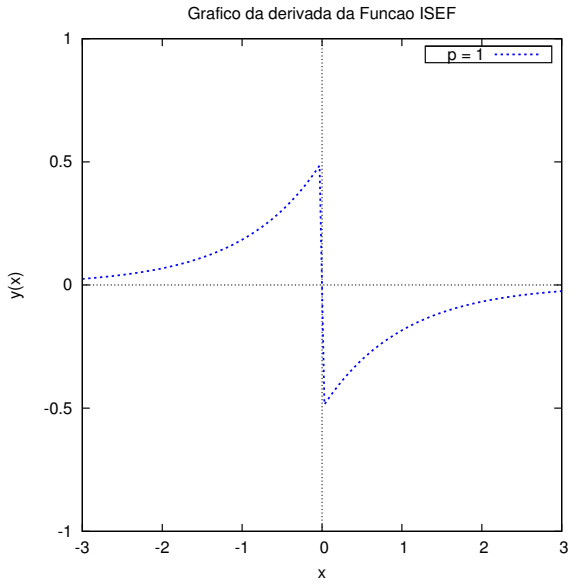
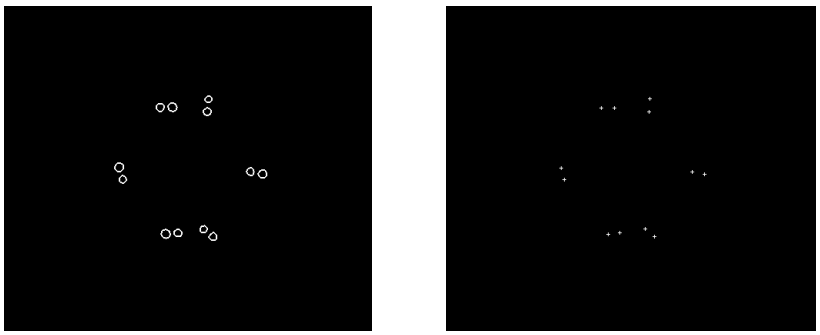


Figura 12: Derivada de primeira ordem da função ISEF.

3.2.3 Terceira Etapa da Pipeline

Após ter recebido a imagem de bordas, resultado da aplicação do filtro de Shen-Castan (1992), aplica-se a transformada de Hough nesta imagem, obtendo assim o centro de cada círculo (Fig. 13).



(a) (b)

Figura 13: (a) Imagem após a aplicação do filtro de Shen-Castan e (b) imagem com os centros dos círculos encontrado pela transformada de Hough.

Tendo calculado o centro de cada círculo, é montado um vetor de centróides de cada robô. Este vetor será passado para o sistema de IA, caso se use processamento local, ou então será enviado via rede, usando um protocolo de comunicação, para uma máquina remota, onde será calculada a IA do sistema.

3.2.3.1 Aplicação da Transformada de Hough

Detectar círculos de raio fixo em uma imagem que contém apenas pontos de borda de coordenadas (x, y) utilizando a Transformada de Hough, consiste em determinar quais desses pontos pertencem à borda do círculo com centro em (xc ,yc) e raio r. O algoritmo determina para cada ponto de borda (x, y) da imagem um conjunto de possíveis centros (xc, yc) no espaço de Hough, conjunto o qual será definido iterativamente pela variação do ângulo θ, de acordo com as Equações 2.9 e 2.10.

A Figura 14 ilustra três círculos de raio r desenhados no espaço de Hough, a partir de três pontos (x, y) de borda do círculo da imagem. É notável que eles se intersectam em apenas um ponto, justamente o ponto central de mesmas coordenadas (xc, yc) do círculo presente na imagem.

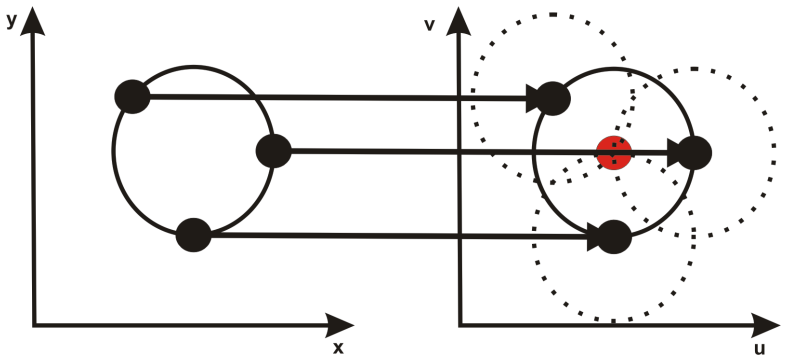


Figura 14: Geração de pontos no espaço de Hough.

A partir da imagem binária resultante do filtro de Shen-Castan (1992) pode-se gerar o espaço de Hough.

Com o espaço de Hough gerado, detectar possíveis centros (xc, yc) de círculos de raio fixo r em uma imagem se resume a encontrar os pontos de máximos locais, os mais votados nesse novo espaço.

Essa etapa é a que apresenta maiores problemas na detecção de círculos, pois como todo e qualquer ponto de borda gera um conjunto de pontos (círculo) que são votados no espaço de Hough, pode haver votos falsos, ocasionando em máximos locais falsos. Um ponto recebe um voto quando alguma circunferência gerada por um ponto de borda passa por ele.

Outro problema é determinar o número de votos mínimo para que se possa considerar tal ponto como sendo o centro de um círculo. Para resolver o problema do número de votos mínimos, após o espaço de Hough ter sido gerado, é verificado se o ponto votado

ultrapassou um determinado número de votos mínimos. Se tal ponto ultrapassou esse limiar, ele é armazenado, inserido em um vetor.

Para garantir que todos os pontos que representam centros reais de círculos na imagem sejam inseridos no vetor, é definido um baixo número de votos mínimo. Porém, pelo fato de se ter um baixo número de votos mínimo, haverá pontos referentes a falsos máximos locais presentes nesse vetor.

O primeiro passo para separar, os falsos máximos locais, de pontos de centros de círculo reais, é a ordenação dos pontos pelo número de votos recebidos com o algoritmo *quicksort*. Após a ordenação, o ponto máximo global, presente na primeira posição do vetor, é considerado um centro de círculo. O ponto que possui o segundo valor máximo é comparado com o ponto anterior. A comparação é feita calculando a distância euclidiana entre os pontos, onde essa distância deve ser maior ou igual $2R$.

Cada ponto que representa um centro de círculo é inserido em um novo vetor, o vetor de centros. O importante é ter um número de votos mínimo, capaz de garantir que todo ponto que representa um círculo real esteja presente no vetor de centros e que cada círculo seja representado por um e somente um ponto presente nesse vetor.

3.2.3.2 Protocolo de Comunicação

Na ciência da computação, um protocolo é um padrão que controla e possibilita uma conexão, comunicação ou transferência de dados entre dois sistemas. Um protocolo pode ser definido como as regras que governam a sintaxe, semântica e sincronização da comunicação.

Após a estrutura que armazena as posições dos robôs ter sido montada, esta é enviada para uma outra máquina, via rede, para que se possa calcular a estratégia de IA.

O protocolo servidor pode ser divido em três etapas:

- Conexão de novo cliente: O servidor fica aguardando em uma *thread* separada, por uma conexão da máquina cliente. A máquina cliente deve informar qual time está se conectando ao servidor. Para isso ela envia uma mensagem dizendo o time que esta se conectando e espera uma resposta de conexão estabelecida. Caso o time esteja disponível, o conteúdo do *peer* cliente é movido para novo *peer* do servidor.
- Envio de nova posição: Após a conexão estabelecida, o *peer* servidor envia as posições dos robôs para a máquina cliente. É necessário verificar se a conexão está

pronta ou não para enviar mensagens. Para isso, o *peer* é bloqueado até que um evento ocorra ou o *timeout* seja estourado. Se a conexão estiver pronta para enviar a mensagem, e o buffer da máquina cliente está vazio para armazenar as posições dos robôs, está é enviada, caso contrário ela é descartada pelo servidor, que fica à espera de novas posições. Essa verificação é feita durante um tempo máximo de 8ms. Esse tempo foi definido por ser um valor razoável de espera para uma mensagem e outra, pois caso este seja maior, poderá afetar no desempenho do resto do sistema, acarretando em um atraso na *pipeline*. Por isso é necessário descartar a mensagem caso o *timeout* ultrapasse o limite.

- Recebimento do ajuste de velocidade: Depois das posições terem sido enviadas com sucesso, o servidor envia uma mensagem de requisição da mensagem que contém os ajustes nas velocidades das rodas dos robôs e fica aguardando o recebimento dessa.

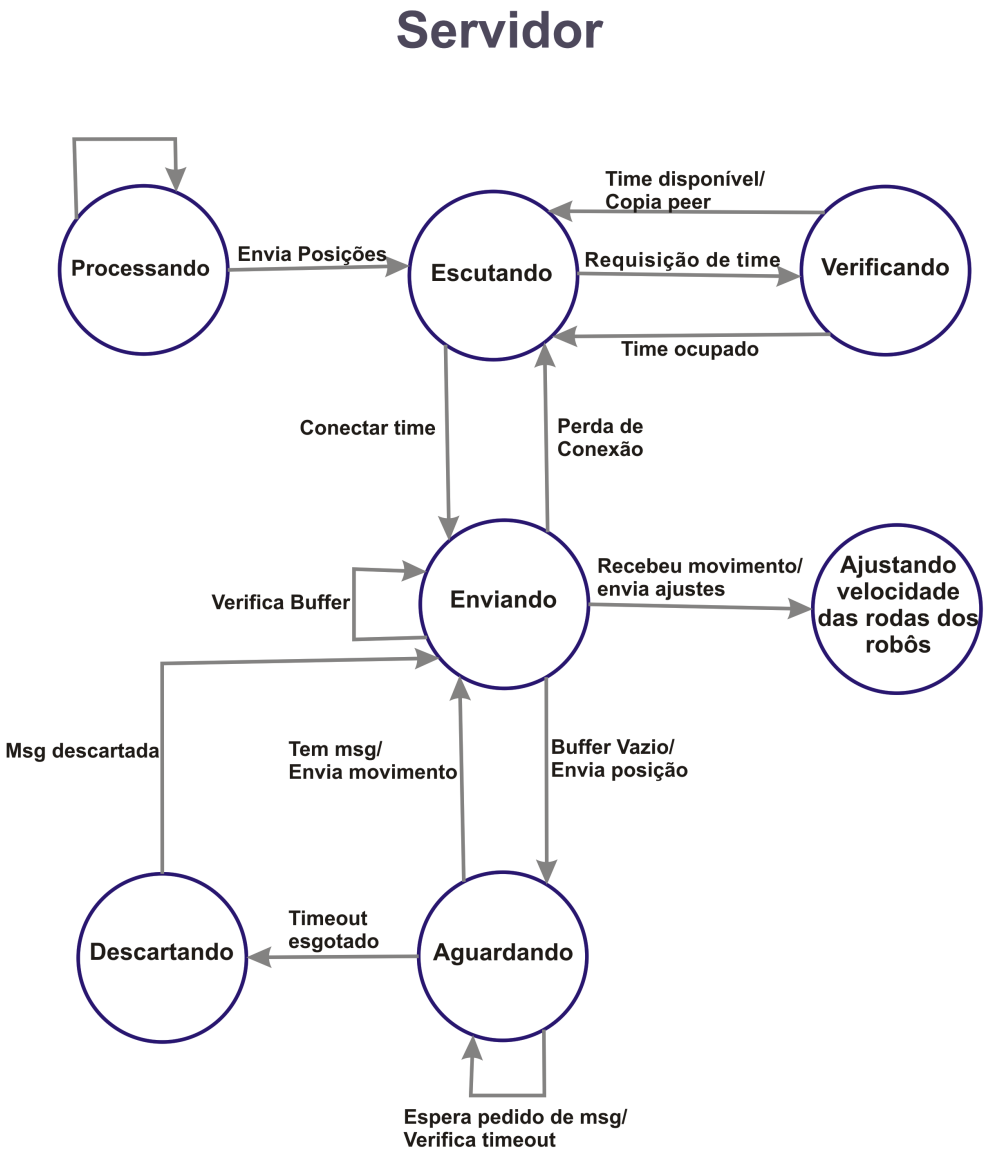


Figura 15: Protocolo Servidor.

O protocolo cliente também foi divido em três etapas:

- Conexão de novo cliente: A máquina cliente inicialmente informa ao servidor qual time está tentando se conectar e espera uma resposta do servidor dizendo se está disponível a conexão. Caso o time esteja disponível, o conteúdo do *peer* cliente é movido para o *peer* do servidor.

- Recebimento de nova posição: Após a conexão estabelecida, o *peer* cliente recebe as posições dos robôs. É necessário verificar se a conexão está pronta ou não para receber mensagens. Para isso, o *peer* cliente é bloqueado até que um evento ocorra ou o *timeout* seja estourado. Se a conexão estiver pronta para receber a mensagem, e o buffer da máquina cliente estiver vazio, as posições dos robôs são recebidas, caso

contrário elas são descartadas pelo cliente, que fica a espera de novas posições. Essa verificação é feita durante um tempo máximo de 8ms pelo mesmo motivo explicado no protocolo servidor.

- Envio do ajuste de velocidade: Depois das posições terem sido recebidas com sucesso, é calculada a estratégia de IA. Depois de calculada, o cliente espera o servidor enviar uma mensagem de requisição do ajuste das velocidades das rodas dos robôs. O cliente após receber essa mensagem verifica se a conexão está pronta ou não para enviar mensagens. Para isso, o *peer* cliente é bloqueado até que um evento ocorra ou o *timeout* seja estourado. Se a conexão estiver pronta para enviar a mensagem e o buffer da máquina servidor está vazio para armazenar essa mensagem, os ajustes das velocidades dos robôs são enviadas, caso contrário ela é descartada pelo cliente, que fica a espera de novas posições.

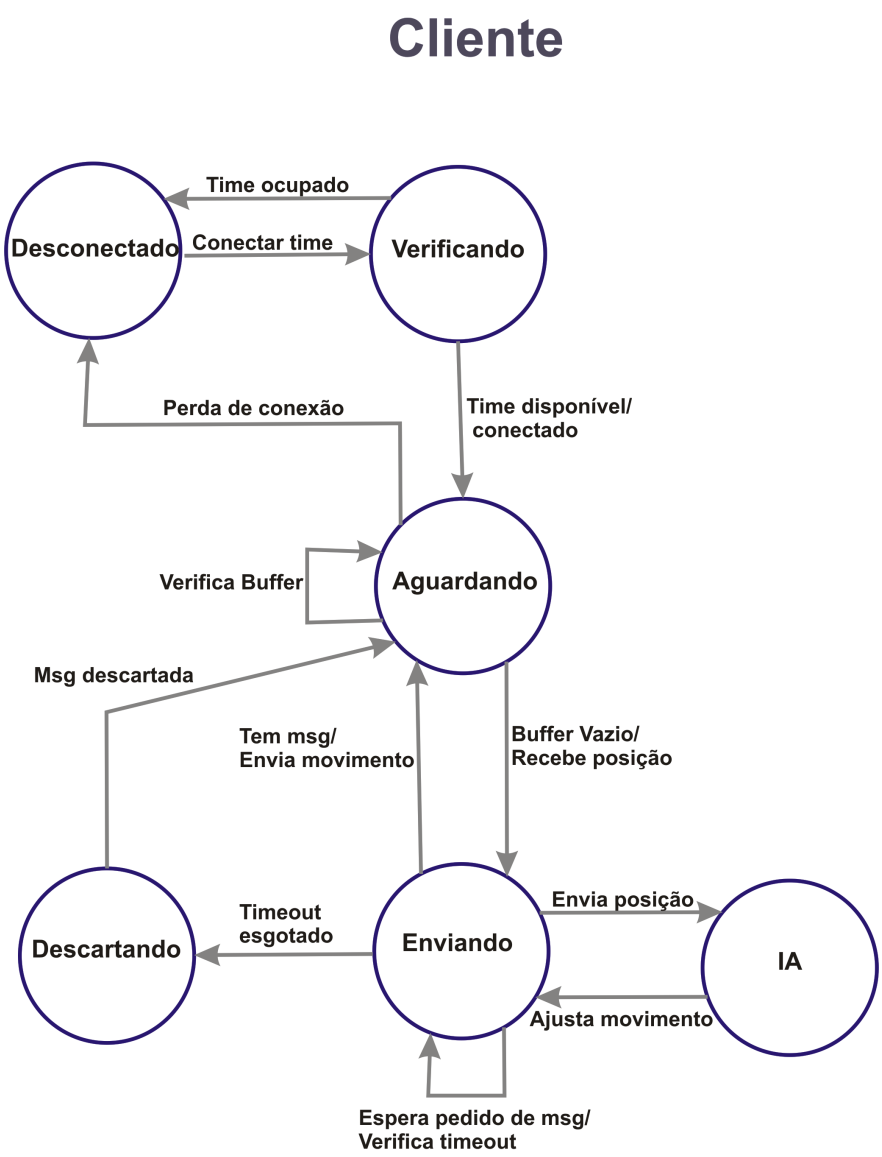


Figura 16: Protocolo Cliente.

3.2.4 Quarta Etapa da Pipeline

Nesta etapa da pipeline, é recebida uma estrutura que contém as posições de cada robô e da bola. Com essas informações, pode-se calcular a estratégia de IA, que é previamente definida pelo usuário. Em seguida, uma outra estrutura é preenchida com os ajustes necessários da velocidade da roda de cada robô. Esta nova estrutura será enviada de volta para a máquina principal, utilizando o mesmo protocolo de comunicação, caso o processamento tenha sido feito remotamente.

Com a estrutura montada, resta apenas enviar para os robôs os comandos necessários para ajustar a velocidade de cada roda, para que ele possa traçar uma trajetória de acordo com a IA definida.

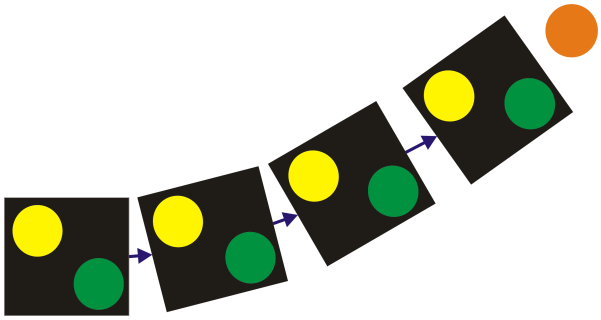


Figura 17: Velocidade de cada roda do robô sendo ajustada para ir atrás da bola.

4 Resultados

Neste capítulo são mostrados os resultados obtidos com o sistema desenvolvido neste trabalho. Foi utilizado para os testes, uma bola de tênis de mesa na cor laranja e dois times de robôs, desenvolvidos pelos alunos de Engenharia Elétrica da UFJF.

Será representada abaixo, uma sequência de figuras representando as etapas do processamento de imagens até o resultado final do sistema. Na primeira etapa foi feito uma suavização da imagem capturada pela câmera com um filtro triangular 3x3 (Fig. 19).

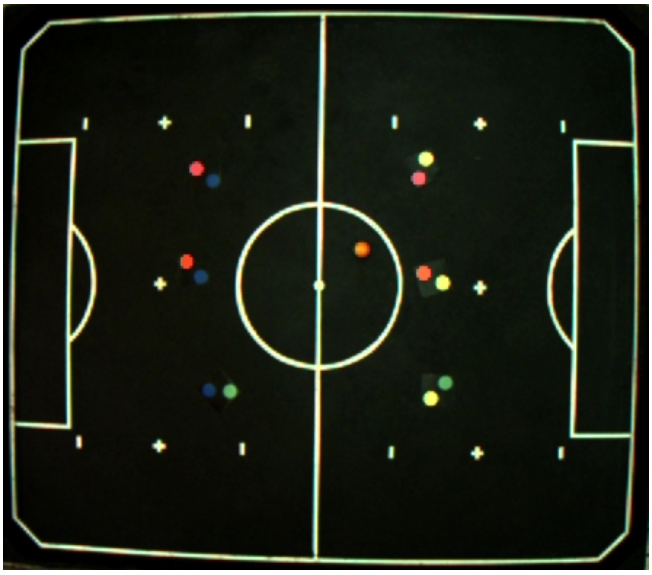


Figura 18: Imagem original.

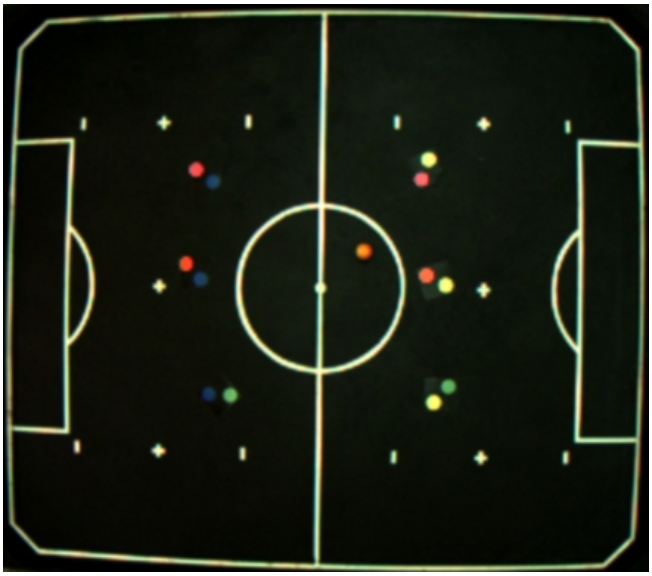


Figura 19: Imagem suavizada.

Em seguida, é aplicada a matriz de calibração de cores na imagem suavizada (Fig. 20).

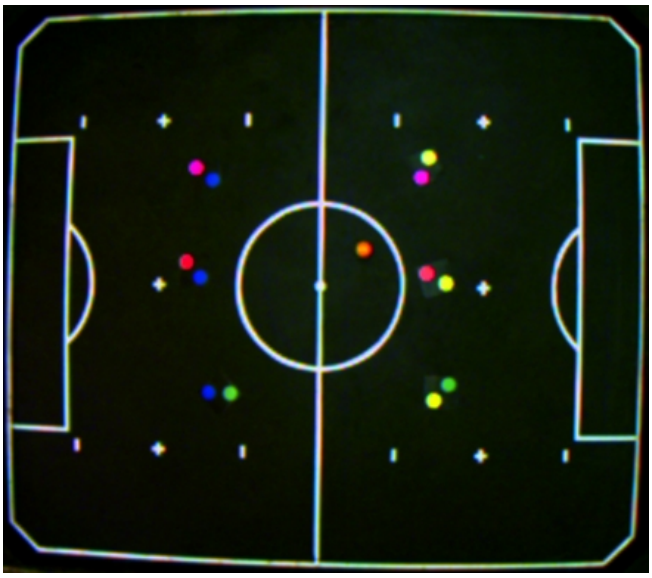


Figura 20: Imagem com as cores realçadas.

Depois de aplicado a matriz de calibração de cores, é feito uma subtração da imagem resultante da imagem do campo sem os robôs e bola (Fig. 21).

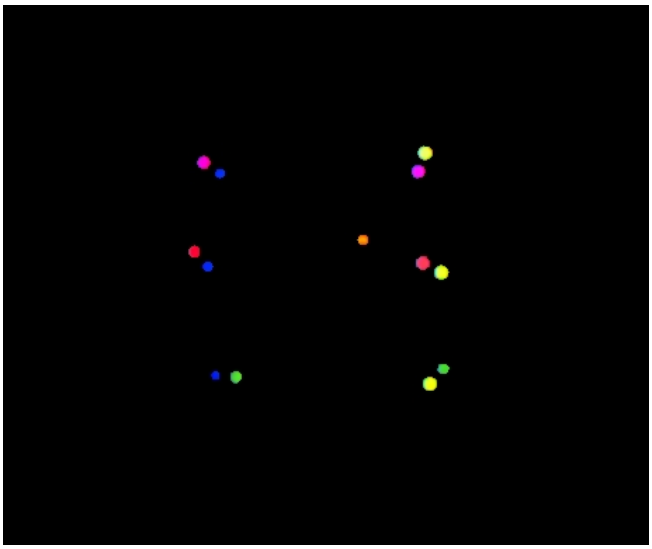


Figura 21: Imagem com os robôs e bola.

Com a imagem apenas dos robôs e da bola, é feito uma binarização da imagem (Fig. 22).

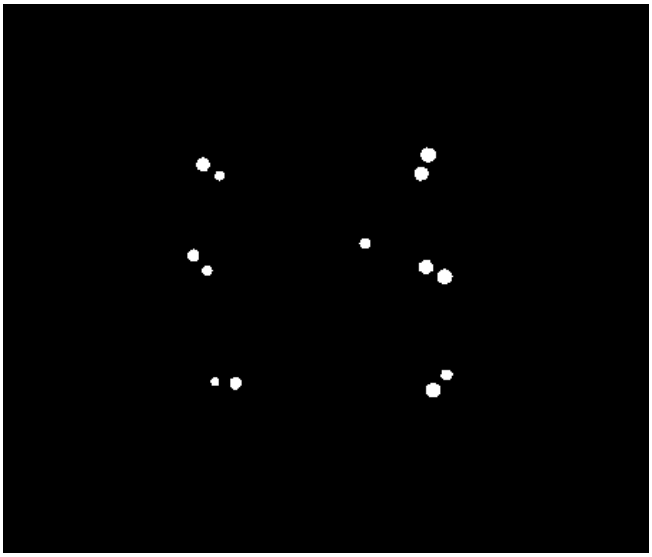


Figura 22: Imagem binarizada.

Para extrair as bordas dessa imagem binarizada, utilizamos o filtro de Shen-Castan (1992) (Fig. 23).

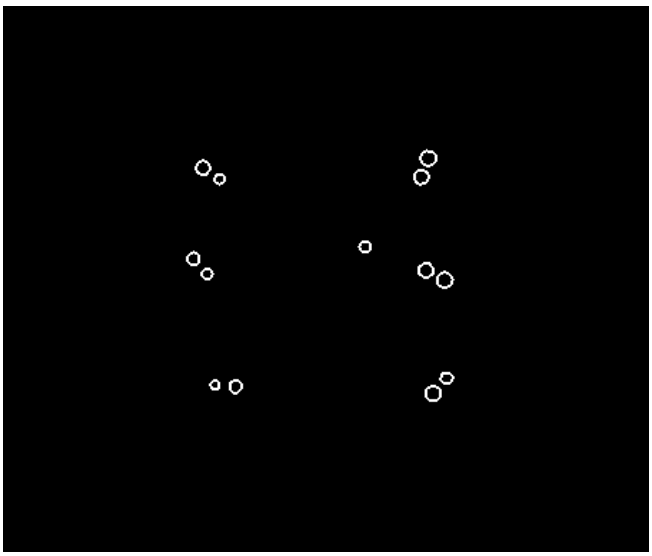


Figura 23: Imagem com as bordas dos círculos.

Após ter extraído as bordas da imagem, é feito uma transformada de Hough para detectar os centros de cada círculo (Fig. 24).

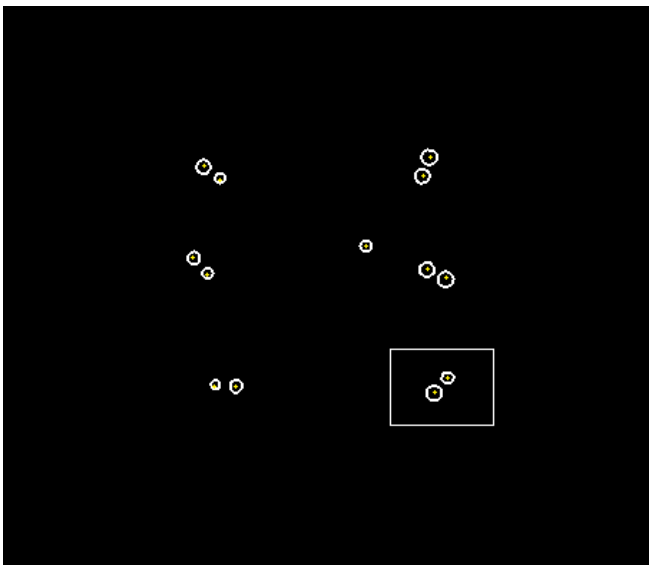


Figura 24: Imagem com os centros de cada círculo.

Pode-se notar que mesmo com uma falha na detecção de alguns círculos, devido a variação na iluminação do campo, a transformada de Hough se mostra eficiente para conseguir achar o centro do círculo defeituoso (Fig. 25).

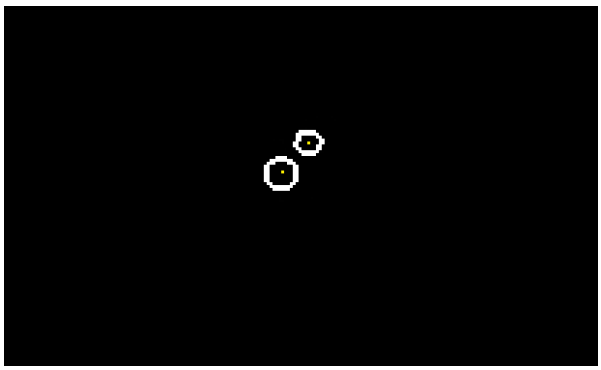


Figura 25: Imagem com círculo defeituoso.

Feito isso, resta apenas montar o vetor de centróides de cada robô, para transferir as posições de cada um deles para que o sistema de IA possa fazer os ajustes necessários de velocidade em cada roda.

A Tabela (1) mostra o tempo de processamento de algumas dessas etapas.

Tabela 1: *Tempo de Processamento*

Processamento	Média	Desvio Padrão
Detecção do Campo	112ms	11,331
Subtração de imagem	5ms	1,146
Transformações de Cores	6ms	2,505
Binarização da Imagem	1ms	0,548
Filtro Shen-Castan	122ms	13,299
Transformada de Hough	9ms	5,295
Cálculo do Centróide	8ms	4,348
Etapa 1	23ms	2,139
Etapa 2	124ms	17,334
Etapa 3	25ms	2,451
Etapa 4	10ms	3,671

Foi medida a latência da rede para a comunicação com o sistema remoto, no momento em que é feito o transporte da estrutura que armazena as posições dos robôs (Fig. 26). O tamanho da estrutura é de 520 bytes.

O tempo medido no servidor, desde o envio do pacote para a máquina cliente até o recebimento da mensagem foi de 14ms em média, com desvio padrão de $5,021$. Já na máquina cliente, o tempo gasto da chegada da estrutura das posições até o envio da nova estrutura foi de 15ms em média, com desvio padrão de $3,397$. Esse teste foi feito usando

uma rede wireless 802.11n.

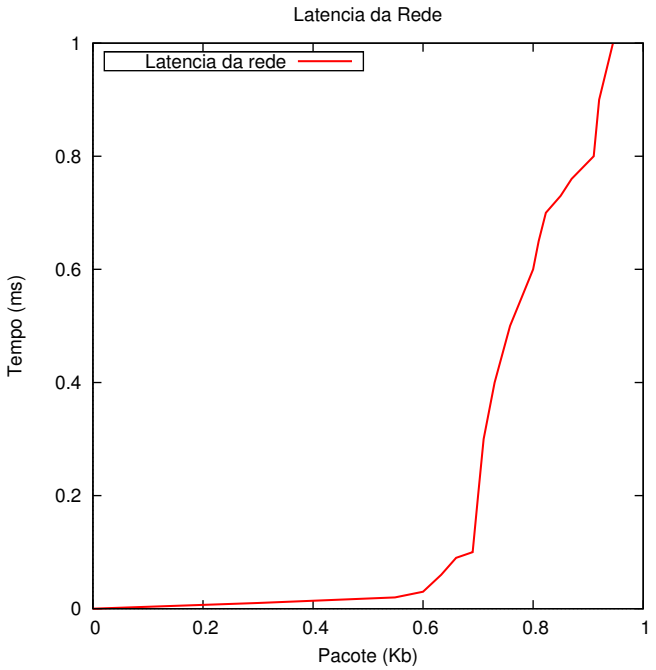


Figura 26: Latência da rede.

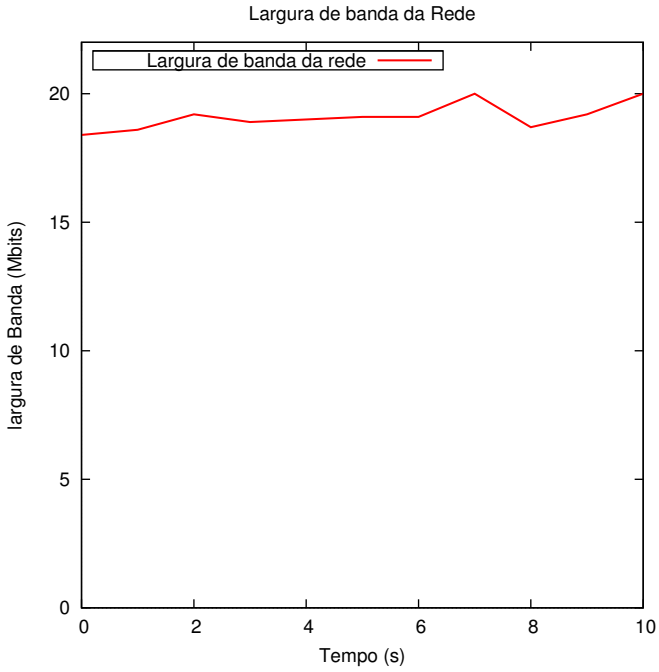


Figura 27: Largura de banda da rede.

5 Conclusões

Foi apresentado neste trabalho, técnicas necessárias para realização de uma partida de futebol de robôs autônomos, partindo da captura das imagens pela câmera, passando por uma comunicação remota entre duas máquinas, até o controle em tempo real dos robôs.

Este trabalho teve foco principalmente no processamento das imagens, apresentando os princípios para operação de filtros de passa-alta, que permite a passagem das altas frequências, porém reduz a amplitude das baixas frequências, possibilitando extrair as bordas de uma imagem. Foi apresentada a transformada de Hough clássica, método robusto para achar os centros dos círculos, e consequentemente as posições de cada robô. Por último, foi feito um protocolo de comunicação para que a IA fosse processada remotamente, tentando diminuir o tempo de processamento de todo o sistema.

É notável que o filtro utilizado (ISEF) é a operação mais custosa de todo o sistema, tendo seu tempo de processamento em torno de 122ms, tempo muito alto se comparado ao outros processos. Esse tempo impossibilita que o sistema seja de tempo real, pois só seria possível processar 8 imagens por segundo, desperdiçando a capacidade de entrega de quadros pela câmera (120 qps). Foi possível notar também que a resposta deste filtro prejudica a etapa posterior, pois o filtro tem como resultado uma borda de espessura grossa, fazendo com que a transformada de Hough tenha que analisar pixels a mais.

Em relação à transformada de Hough, observou-se que mesmo em imagens ruidosas, onde os círculos estão incompletos ou deformados, ela conseguiu de forma precisa detectar o centro desses círculos.

Neste trabalho, foi criado um protocolo de comunicação entre duas máquinas para que fosse calculada a inteligência artificial remotamente, para evitar um atraso na execução da *pipeline*.

Por último foram apresentados os resultados obtidos através de um teste realizado com dois times de robôs e uma bola. Esse teste mostrou uma boa detecção dos robôs e

bola, porém foi possível notar que houve uma variação dos centros dos círculos, devido a resposta múltipla do filtro de detecção de bordas utilizado.

Para trabalhos futuros pretende-se corrigir o problema do filtro para detecção de bordas utilizado (Shen-Castan), dividindo o seu processamento em etapas através da *pipeline*, já que o filtro de Shen-Castan é separável. Caso a solução não seja viável, será implementado outro filtro, abordando uma maneira diferente de localizar as bordas.

Por último, será feito a inclusão de técnicas de inteligência artificial mais robustas, para que se tenha um sistema competitivo.

Referências

CANNY, F. J. A Computational Approach to Edge Detection. v. 8, n. 6, p. 679–698, 1986.

DUDA, O.; HART, P. E. *Use of the Hough transformation to detect lines and curves in pictures*. 1971.

FIRA. *Regras*. 2010. Disponível em: http://www.fira.net/?mid=mirosot. Acesso em: 12 de Dezembro de 2010.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. [S.l.]: Addison Wesley, 1993.

MARR, D.; HILDRETH, E. Theory of edge detection. *Proceedings of the Royal Society of London Series B*, v. 207, p. 187–217, 1980.

MCLAUGHLIN, R. A. *Randomized Hough Transform: Improved Ellipse Detection with Comparison*. [S.l.], 1998.

PEREZ Éder de A. *Detecção e controle de robôs para futebol autônomo*. Monografia (M) — Faculdade de Ciência da Computação, Universidade Federal de Juiz de Fora, Juiz de Fora, 2009.

RODRIGUES, D. L. et al. *An Application of Hough Transform to Identify Breast Cancer in Images*. 2007.

SHEN, J.; CASTAN, S. An optimal linear operator for step edge detection. *CVGIP: Graphical Model and Image Processing*, v. 54, n. 2, p. 112–133, 1992.

ZIOU, D.; TABBONE, S. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, v. 8, p. 537–559, 1998.