Eder de Almeida Perez

Detecção e controle de robôs para futebol autônomo

Juiz de Fora

Eder de Almeida Perez

Detecção e controle de robôs para futebol autônomo

Orientador:
Marcelo Bernardes Vieira

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

Juiz de Fora

Monografia submetida ao corpo docente do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como parte integrante dos requisitos necessários para obtenção do grau de bacharel em Ciência da Computação

Prof. Marcelo Bernardes Vieira, D. Sc.
Orientador

Prof. Marcelo Lobosco, D. Sc.

Prof. André Marcato, D. Sc.

Sumário

Lista de Figuras

Resumo

1 Introdução p. 8

1.1 Definição do Problema . . . . p. 8

1.2 Objetivos . . . . p. 9

1.3 Visão Geral . . . . p. 9

2 Fundamentos p. 11

2.1 Processamento de imagens . . . . p. 11

2.1.1 Filtros Lineares Espacialmente Invariantes . . . . p. 11

2.1.1.1 Filtro passa-baixa . . . . p. 13

2.1.1.2 Filtro laplaciano . . . . p. 15

2.2 Calibração de cor . . . . p. 16

2.3 Filtro de Kalman . . . . p. 17

2.3.1 O algoritmo do filtro de Kalman . . . . p. 18

2.4 Controle PID . . . . p. 19

2.4.1 O termo proporcional no controle . . . . p. 20

2.4.2 O termo integral no controle . . . . p. 20

2.4.3 O termo derivada no controle . . . . p. 20

2.5 Reconhecimento de padrões . . . . p. 21

3 Detecção e controle de robôs p. 23

3.1 Ambiente . . . . p. 23

3.2 Estrutura geral . . . . p. 24

3.2.1 Inicialização do sistema . . . . p. 24

3.2.2 Classificação das cores . . . . p. 29

3.3 Detecção dos agentes da cena . . . . p. 32

3.4 Controle em tempo real dos robôs . . . . p. 34

4 Resultados p. 37

5 Conclusão p. 42

Apêndice A – Máscaras e padrões utilizados p. 44

Referências p. 48

Lista de Figuras

1 Modelo de etiquetas utilizada nos robôs. A etiqueta amarela identifica o time, enquanto que a etiqueta vermelha identifica o robô. . . . . p. 10

2 Variações de brilho pelo campo. . . . . p. 10

3 Aplicação de um filtro w de tamanho 3x3 em uma imagem de entrada f. No detalhe, a sobreposição do filtro nos pixels correspondentes da imagem. p. 12

4 Função pulso. . . . . p. 13

5 (a) filtro box de tamanho 3x3. (b) imagem original. (c) imagem suavizada pelo filtro. . . . . p. 13

6 Núcleo do filtro triangular. . . . . p. 14

7 (a) filtro triangular 5x5. (b) imagem original. (c) imagem suavizada pelo filtro. . . . . p. 14

8 (a) filtro gaussiano 5x5. (b) imagem original. (c) imagem suavizada pelo filtro. . . . . p. 15

9 (a) imagem original. (b) imagem filtrada por um filtro gaussiano 15x15. (c) imagem das altas frequências. . . . . p. 16

10 Diagrama da operação do filtro de Kalman (WELCH; BISHOP, 2001). . . p. 19

11 (a) subimagem p com o padrão. (b) marca de pênalti do campo. (c) detalhe da área que será varrida por p. (d) coordenada central do padrão encontrado. . . . . p. 22

12 Dimensões e marcações do campo. P refere-se à marcação de pênalti e LL à marcação de lance livre (as barras ao lado são as posições onde dois robôs deverão ficar durante um lance livre). . . . . p. 23

13 (a) imagem antes da calibração de cores. (b) imagem depois da calibração de cores. . . . . p. 25

14 Detecção das altas frequências do campo. . . . . p. 25

15 Reconhecimento das interseções da linha central do campo com o círculo central. . . . . p. 26

16 Fluxograma da aplicação do reconhecimento de padrões no campo. . . . p. 27

17 (a) Detecção das marcações do campo. (b) detalhe da marca do pênalti direito, o ponto destacado é a coordenada estimada pelo filtro de kalman. p. 27

18 Fluxograma do funcionamento das threads. . . . . p. 28

19 Vetores de classes de cores (KLANCAR et al., 2005). Cada classe de cor possui um bit que a representa dentre os 32 bits de cada posição nos vetores. . . . . p. 30

20 Classificação de um pixel de cor R = 250, G = 125, B = 15. O oitavo bit da direita para esquerda representa a classe de cor laranja. . . . . . p. 31

21 (a) imagem antes da classificação dos pixels. (b) imagem depois da classificação dos pixels. . . . . p. 31

22 Problemas de associação de regiões para formar agentes. . . . . p. 33

23 Formação de ciclo entre as regiões. p. 34

24 Erro entre o ângulo e posição atuais do robô e o ângulo e posição desejados. p. 35

25 Robô do time azul perseguindo a bola no simulador da FIRA. . . . . . p. 36

26 Subtração da imagem original pela imagem filtrada com passa-baixa. . p. 37

27 Detecção dos marcadores do campo. A imagem foi invertida e os pontos realçados para melhor visualização. . . . . p. 38

28 Imagem sem processamento entregue por um objeto gcgVIDEO. . . . . p. 38

29 Filtragem passa-baixa para redução de ruídos da imagem. . . . . p. 39

30 Transformação de cores pela matriz de calibração. . . . . p. 39

31 Classificação de pixels e detecção das regiões de interesse. . . . . p. 40

32 Cálculo do centróide de cada região. A imagem foi invertida e os pontos realçados para melhor visualização. . . . . p. 40

Resumo

Este trabalho trata do problema de rastreamento e controle em tempo real em partidas de futebol de robôs. São apresentados os fundamentos matemáticos de filtragem de imagens, calibração de cores, filtros de Kalman, controle em tempo real e reconhecimento de padrões. Cada uma destas abordagens é implementada para obtenção de um sistema capaz de reconhecer os objetos em campo e controlar os robôs de forma precisa.

1 Introdução

> A frase mais excitante para se ouvir em ciência, aquela que anuncia novas descobertas, não é "Eureka!"mas "Isto é divertido..."
>
> Isaac Asimov

Futebol de robôs é uma iniciativa interdisciplinar voltada à educação e pesquisa. Seu objetivo principal é o desenvolvimento de áreas como Visão Computacional, Inteligência Artificial e Robótica Inteligente. Trata-se de um problema padrão, logo, diversas teorias, algoritmos e arquiteturas podem ser avaliadas. Muitos desafios presentes no futebol de robôs são também encontrados no mundo real. Como, por exemplo, veículos autônomos, controle de tráfego aéreo e urbano, entre outros. Desta forma, o futebol de robôs torna-se uma grande área para pesquisa e ensino de diversas técnicas computacionais, microeletrônica e engenharia mecânica.

Em 1993 foi criada a RoboCup, competição internacional cujo objetivo era desenvolver robôs autônomos capazes de jogar uma partida de futebol. A competição visa promover a pesquisa e educação nas áreas de ciência da computação e engenharias. A RoboCup tem como objetivo maior ser capaz de desenvolver, em meados do século 21, um time completo de humanóides autônomos que seja capaz de ganhar uma partida de futebol, obedecendo as regras da FIFA, contra o atual campeão da Copa do Mundo

1.1 Definição do Problema

Este trabalho trata do problema de visão e controle em tempo-real em partidas de futebol de robôs. Para que o sistema de IA consiga tomar decisões estratégicas nesses jogos, é preciso que lhe seja fornecido informações precisas sobre as coordenadas do campo e os agentes da cena (neste caso, os robôs e a bola). As informações sobre os agentes da cena são a posição de cada um no campo e a orientação dos robôs.

O sistema de visão deve então ser capaz de reconhecer todos os objetos de interesse da cena, como robôs, bola e as coordenadas do campo. Esta última refere-se às coordenadas relativas ao centro do campo, área do gol, laterais e linha de fundo, além das marcas de pênalti e faltas.

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 fino dos mesmos. O controle é feito através do ajuste das velocidades das rodas dos robôs.

1.2 Objetivos

O objetivo desta monografia é pesquisar e formalizar matematicamente os métodos de:

- Calibração de cores da câmera;
- Reconhecimento de padrões do campo;
- Segmentação dos quadros em regiões de interesse e identificação dos centróides de cada região;
- Aplicação de um filtro de Kalman para correção dos ruídos;
- Identificação das coordenadas e rotação de cada robô e coordenadas da bola;
- Controle dos robôs em tempo real utilizando PID.

1.3 Visão Geral

Num jogo de futebol de robôs, as informações sobre os objetos de interesse e o campo são extraídas a partir de uma câmera posicionada a, no mínimo, $2m$ de altura do centro do campo. Cada imagem capturada por essa câmera é definida como um *quadro*.

Um robô é definido por duas etiquetas de cores diferentes. Uma etiqueta representa a cor do time (pode ser amarela ou azul) e a outra etiqueta representa a cor que individualiza cada robô de um mesmo time (Fig. 1). A bola deve ser uma bola de golfe laranja.

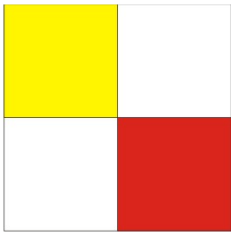


Figura 1: Modelo de etiquetas utilizada nos robôs. A etiqueta amarela identifica o time, enquanto que a etiqueta vermelha identifica o robô.

Devido às variações de iluminação por todo campo, uma mesma cor pode apresentar diferenças de brilho em diferentes partes do campo (Fig. 2). Sendo assim, as cores captadas pela câmera devem ser transformadas em um conjunto de cores conhecido. É necessário então aplicar uma calibração de cores. Em seguida, todas as regiões de cores de interesse da cena devem ser encontradas. As regiões de cores de interesse são aquelas correspondentes à bola e as etiquetas de cada robô. Uma vez reconhecidas, as regiões referentes às etiquetas dos robôs devem ser agrupadas duas a duas para que sejam calculadas as coordenadas e rotação dos mesmos.

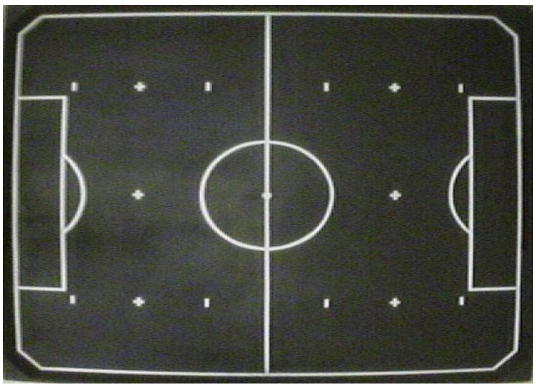


Figura 2: Variações de brilho pelo campo.

Como existem ruídos intrínsecos ao ambiente e à câmera, o cálculo destas coordenadas pode não corresponder a uma boa aproximação da realidade. Então, uma correção através de um filtro de Kalman deve ser aplicada às coordenadas encontradas.

O controle em tempo real dos robôs, responsável por fazer com que um robô execute uma determinada ação definida pelo sistema de IA, é feito através de um controlador do tipo PID (Proporcional-Integral-Derivativo). As velocidades das rodas, esquerda e direita, são ajustadas para que o robô execute esta ação.

2 Fundamentos

Neste capítulo será abordada a fundamentação teórica que dará suporte a este trabalho. Serão tratados os problemas de processamento de imagens, calibração de cores, predição e estimativa, controle em tempo real e reconhecimento de padrões.

2.1 Processamento de imagens

Um sinal se manifesta através da variação de uma grandeza física que pode ocorrer em função do espaço ou do tempo.

Em uma imagem de uma cena real, cada ponto do espaço da imagem recebe um impulso luminoso que associa uma informação de cor a esse ponto (GOMES; VELHO, 2008). Desta forma, uma imagem pode ser entendida como um sinal cuja variação da grandeza física ocorre no espaço. Podemos então definir um modelo matemático espacial para descrever uma imagem através de uma função $f : U \subset \Re^2 \rightarrow C$, no qual o conjunto U é chamado de *suporte* da imagem e o conjunto C é um espaço de cor (GOMES; VELHO, 2008).

2.1.1 Filtros Lineares Espacialmente Invariantes

O principal objetivo da aplicação de filtros em imagens é torná-las mais adequadas que a imagem original para uma aplicação específica (GONZALEZ; WOODS, 2002).

Um filtro linear espacialmente invariante (L.E.I) é uma aplicação F em um sinal f (no nosso caso, em uma imagem), cujo resultado também é um sinal e o filtro atende à condição $F(\alpha f + \beta g) = \alpha F(f) + \beta F(g)$. Este filtro opera num sinal copiando, escalando, atrasando e, por fim, somando os resultados. A resposta será um sinal de mesma frequência que o sinal de entrada. Podemos usá-lo para, por exemplo, detectar bordas em imagens ou suavizar uma imagem ruidosa.

Todo filtro L.E.I é determinado por uma *função resposta de impulso*, que representa o espalhamento da energia do impulso resultante da filtragem. A *função resposta de impulso* para um filtro L.E.I L é dada por $h(t) = L(\delta(t))$, onde $\delta(t)$ é a função delta de Dirac. Ou seja, dado um sinal f, tem-se que $L(f(t)) = f(t) * h(t)$, onde $f(t) * h(t)$ é o produto de convolução entre f e h. A função h também é conhecida como *núcleo do filtro*.

A aplicação computacional de um filtro em uma imagem amostrada se dá através de um produto de convolução 2D entre os valores de uma vizinhança de pixels de uma imagem e uma subimagem com as mesmas dimensões desta vizinhança (GONZALEZ; WOODS, 2002). Esta subimagem é também conhecida como *máscara* e seus valores são chamados *coeficientes*. O produto de convolução 2D entre uma imagem f e uma máscara w é dado pela equação,

$$r(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t), \tag{2.1}$$

onde $a = (m - 1)/2$ e $b = (n - 1)/2$ e mn é o tamanho do filtro.

O processo de aplicar um filtro em uma imagem consiste em mover o centro da máscara em cada pixel da imagem e calcular sua resposta $r(x, y)$. Esse processo é esquematizado na Figura 3.

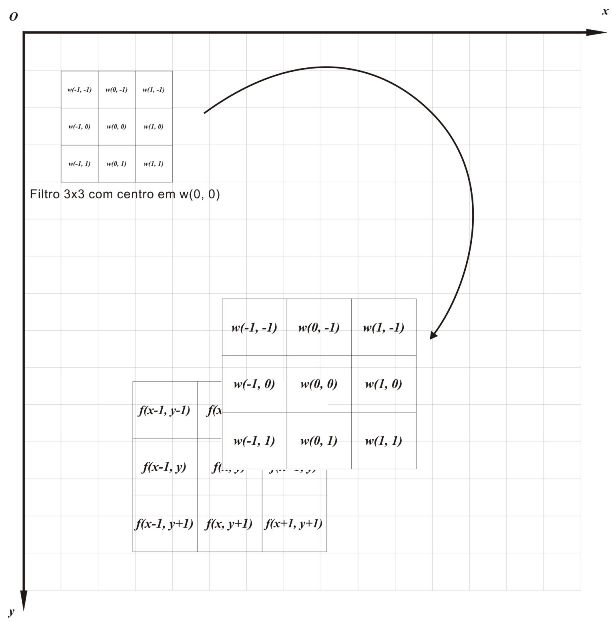


Figura 3: Aplicação de um filtro w de tamanho 3x3 em uma imagem de entrada f. No detalhe, a sobreposição do filtro nos pixels correspondentes da imagem.

2.1.1.1 Filtro passa-baixa

Filtros passa-baixa são usados especialmente para redução de ruídos. Sua resposta é dada pela média ponderada dos valores dos pixels contidos na vizinhança do filtro (GONZALEZ; WOODS, 2002), através da normalização da Equação 2.1. O maior uso destes filtros é a redução de detalhes "irrelevantes"em uma imagem. Por detalhes "irrelevantes"entende-se por regiões de pixels que são pequenas em relação ao tamanho do filtro (GONZALEZ; WOODS, 2002).

Abordaremos neste trabalho três tipos de filtros passa-baixa: filtro box, filtro triangular e filtro gaussiano.

O filtro box é um tipo de passa-baixa cujo núcleo é um pulso,

$$box_1(t) = \begin{cases} 1, & \text{se} \quad |t| \leq x \\ 0, & \text{senão.} \end{cases} \tag{2.2}$$

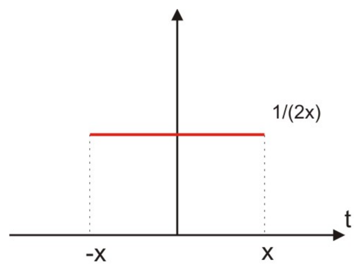
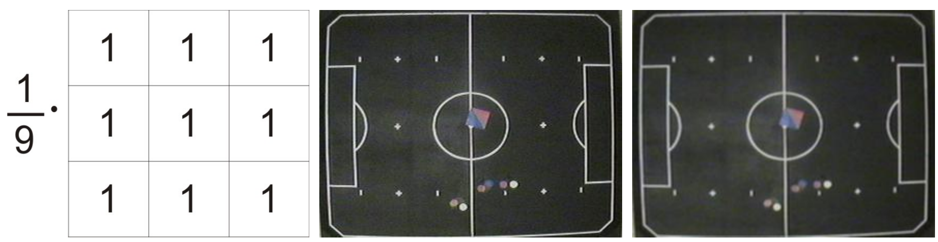


Figura 4: Função pulso.

Abaixo é mostrado o uso do filtro box no campo de futebol de robôs (Fig. 5). A resposta é a média de todos os pixels sob a máscara.



(a) (b) (c)

Figura 5: (a) filtro box de tamanho 3x3. (b) imagem original. (c) imagem suavizada pelo filtro.

O filtro triangular ou de Bartlet possui núcleo dado por

$$triang_1(t) = \begin{cases} \frac{1-|t|}{x}, & \text{se} \quad |t| \leq x \\ 0, & \text{senão.} \end{cases} \tag{2.3}$$

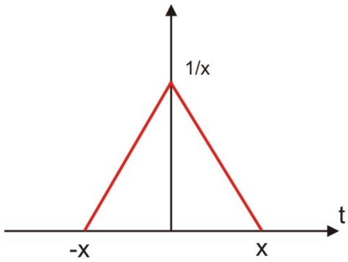
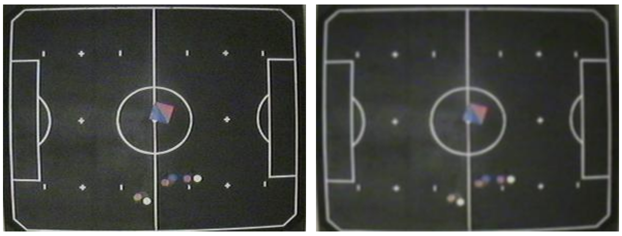


Figura 6: Núcleo do filtro triangular.

Um exemplo do uso de um filtro triangular é mostrado na Figura 7.

$$\frac{1}{81}\cdot\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$



(a) (b) (c)

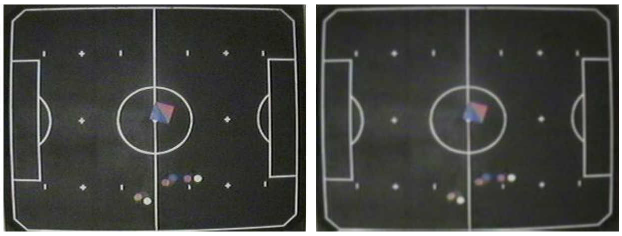
Figura 7: (a) filtro triangular 5x5. (b) imagem original. (c) imagem suavizada pelo filtro.

No filtro gaussiano o núcleo é uma gaussiana

$$G_\sigma(t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2\sigma^2}} \tag{2.4}$$

Um exemplo do uso de um filtro gaussiano é mostrado na Figura 8.

$$\frac{1}{256}\cdot\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$



(a) (b) (c)

Figura 8: (a) filtro gaussiano 5x5. (b) imagem original. (c) imagem suavizada pelo filtro.

2.1.1.2 Filtro laplaciano

O filtro passa-alta tem como principal objetivo detectar bordas ou destacar detalhes em uma imagem. Basicamente, um filtro passa-alta trabalha com diferenciação. O gradiente na direção $\vec{i}$ e $\vec{j}$ de um sinal bidimensional $f(x, y)$ é o vetor

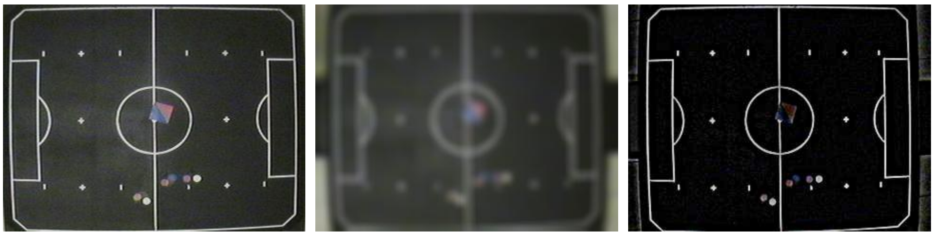
$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right).$$

Este vetor aponta para direção de máxima variação do sinal. É nessas regiões de máxima variação que ocorrem as bordas de uma imagem.

Existem várias formas de se implementar um filtro passa-alta. Este trabalho abordará apenas o filtro laplaciano, que trabalha com a segunda derivada. Sendo assim, o filtro laplaciano de uma função bidimensional f duas vezes diferenciável é definido como

$$\Delta f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \nabla^2 f \tag{2.5}$$

O resultado da passagem de um filtro gaussiano em uma imagem seguido de uma subtração desta imagem pela filtrada é uma aproximação de um filtro laplaciano. Na Figura 9c é mostrado o resultado da subtração da imagem original (Fig. 9a) pela imagem filtrada por um filtro gaussiano 15x15 (Fig. 9b).



(a) (b) (c)

Figura 9: (a) imagem original. (b) imagem filtrada por um filtro gaussiano 15x15. (c) imagem das altas frequências.

2.2 Calibração de cor

As cores dos objetos captadas pela câmera, podem sofrer variações de brilho ao longo da cena devido à diferenças de iluminação. Porém, devemos detectar uma cor de maneira única, independente das variações que ela possa sofrer. Sendo assim, faz-se necessário um método que associe um conjunto de cores captado em um conjunto de cores conhecido.

Seja C a matriz formada por m cores[1] captadas pela câmera (suponha as cores no formato RGB).

$$C = \begin{pmatrix} c_{11}^r & c_{12}^r & c_{13}^r & \dots & c_{1m}^r \\ c_{21}^g & c_{22}^g & c_{23}^g & \dots & c_{2m}^g \\ c_{31}^b & c_{32}^b & c_{33}^b & \dots & c_{3m}^b \end{pmatrix}$$

Queremos encontrar, para cada uma dessas cores, uma cor correspondente num espaço de cores conhecido. Seja K a matriz formada por m cores conhecidas.

$$K = \begin{pmatrix} k_{11}^r & k_{12}^r & k_{13}^r & \dots & k_{1m}^r \\ k_{21}^g & k_{22}^g & k_{23}^g & \dots & k_{2m}^g \\ k_{31}^b & k_{32}^b & k_{33}^b & \dots & k_{3m}^b \end{pmatrix}$$

Se encontrarmos uma matriz de transformação P tal que $CP = K$, então podemos associar uma cor da cena ao espaço de cores que conhecemos. Uma maneira natural de pensarmos, é encontrar a matriz inversa C^{-1} e calcular $C^{-1}CP = KC^{-1} \Leftrightarrow P = KC^{-1}$. Porém, a matriz C não é necessariamente quadrada e não podemos então usar o conceito de matriz inversa. Mas podemos usar o conceito de *pseudo-inversa*, utilizado em regressão.

[1]São usadas as cores das etiquetas dos robôs, da bola, o preto e o branco

A pseudo-inversa de uma matriz não quadrada X é a matriz X^+ tal que $XX^+ \approx I$. Essa matriz é calculada como $(X^t X)^{-1} X^t$.

Sendo assim, podemos agora calcular C^+ e obter $C^+CP = KC^+ \Leftrightarrow P \approx KC^+$.

A aplicação da matriz P é feita em todos os pixels da imagem e nada mais é do que uma transformação linear do espaço de cores RGB. Outras transformações poderiam ser usadas para o mesmo fim, mas esta em questão é suficiente para o nosso problema.

2.3 Filtro de Kalman

O filtro de Kalman é uma ferramenta utilizada para estimação estocástica do estado $x \in \Re^n$ de um sistema governado pela equação diferencial de tempo-discreto (WELCH; BISHOP, 2001),

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{2.6}$$

com uma medição $z \in \Re^m$ definida por:

$$z_k = Hx_k + v_k \tag{2.7}$$

As variáveis aleatórias w_k e v_k representam o ruído do processo e da medição, respectivamente. É assumido que os ruídos são independentes, do tipo branco e com distribuição de probabilidade normal:

$$p(w) \approx N(0, Q), \tag{2.8}$$

$$p(v) \approx N(0, R). \tag{2.9}$$

A matriz $A_{n \times n}$ na Equação 2.6 representa a transformação que relaciona o estado x_{k-1} no tempo $k-1$ ao estado x_k no tempo k, sem levar em conta o ruído do processo. A matriz $B_{n \times l}$ relaciona a entrada de controle (esta pode ser opcional) $u \in \Re^l$ ao estado x. A matriz $H_{m \times n}$ na Equação 2.7 relaciona o estado x_k à medição z_k.

2.3.1 O algoritmo do filtro de Kalman

Este trabalho apresentará o filtro de Kalman sob a forma abordada em (WELCH; BISHOP, 2001).

O estado estimado por um filtro de Kalman é feito através de dois passos: *predição*, através das equações de atualização de tempo e *correção*, através das equações de atualização de medição. A *predição* é responsável por projetar no tempo o estado atual e a covariância do erro a fim de obter uma estimativa *a priori* para o próximo passo. A *correção* é responsável por incorporar uma nova medição na estimativa *a priori* e obter uma estimativa melhorada *a posteriori*.

As equações para atualização de tempo são apresentadas abaixo:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{2.10}$$

$$P_k^- = AP_{k-1}A^T + Q, \tag{2.11}$$

onde A e B são da Equação 2.6 enquanto que Q é da Equação 2.8. O estado estimado *a priori* é representado por $\hat{x}_k^-$. P_k^- é a covariância do erro estimado *a priori* e P_{k-1} a covariância do erro estimado *a posteriori*.

As equações para atualização de medição são dadas por:

$$K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \tag{2.12}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H\hat{x}_k^-\right) \tag{2.13}$$

$$P_k = \left(I - K_k H\right) P_k^-, \tag{2.14}$$

onde H vem da Equação 2.7, R da Equação 2.9 e K_k é o *ganho de Kalman* que minimiza a covariância do erro *a posteriori*.

As equações de atualização de tempo simplesmente projetam as estimativas de estado e covariância do tempo $k-1$ para o tempo k. Nas equações de atualização de medição a primeira tarefa é calcular o *ganho de Kalman*, K_k. O próximo passo é medir o processo para obter z_k e então gerar a estimativa *a posteriori* incorporando essa medição como na Equação 2.13. O passo final é obter a covariância do erro estimado *a posteriori* pela Equação 2.14.

Após cada passo da atualização de tempo e medição, o processo é repetido com as estimativas *a posteriori* usadas para projetar ou prever as estimativas *a priori*. Essa natureza recursiva do filtro de Kalman o torna atrativo para implementação computacional. A Figura 10 mostra a operação completa do filtro de Kalman:

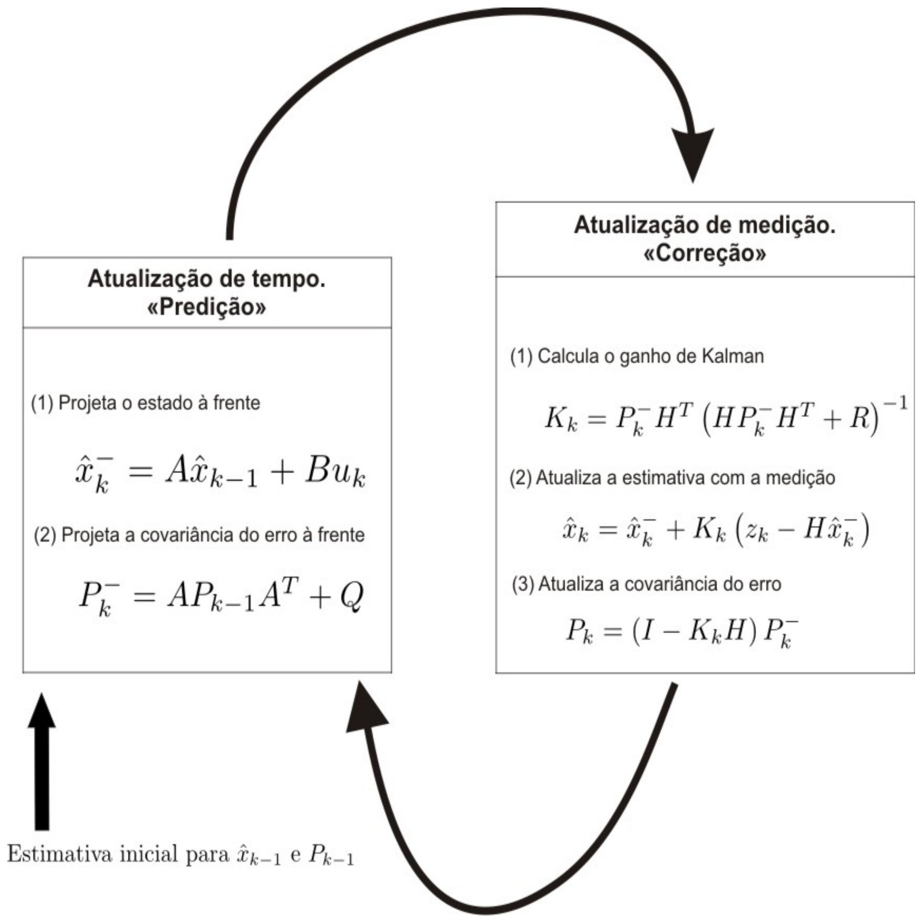


Figura 10: Diagrama da operação do filtro de Kalman (WELCH; BISHOP, 2001).

2.4 Controle PID

O controlador PID é um laço de controle usado amplamente no controle de sistemas industriais. Esta técnica consiste em medir a saída do sistema e comparar com a entrada desejada e, dessa forma, tomar uma ação de correção baseada no erro obtido (GUPTA et al., 2004).

O cálculo do controlador envolve três parâmetros[2]: *proporcional*, *integral* e *derivada*;

[2]Existem ainda diversos parâmetros para o ajuste fino do PID que fogem do escopo deste trabalho.

como mostrado na Equação 2.15.

$$CO = K_p e(t) + K_i \int_0^t e(t) + K_d \frac{de(t)}{dt}, \qquad (2.15)$$

onde o erro $e(t)$ em um determinado instante de tempo t é tomado pela diferença entre o valor de saída desejado do sistema (chamado de *ponto de referência*) e o valor medido. A função do controlador é manter o valor do erro igual a zero ($e = 0$).

Após o cálculo do controle e tomada a ação, a saída é medida novamente e uma nova ação de correção é tomada. Esse tipo de controle é conhecido como *closed-loop*.

2.4.1 O termo proporcional no controle

O termo proporcional $K_p e(t)$ no controlador PID muda a saída do sistema de maneira proporcional ao erro $e(t)$. O valor K_p é conhecido como *ganho proporcional*. Quanto maior o valor de K_p maior a resposta, porém o controle do sistema pode se tornar instável. Por outro lado, quanto menor o ganho K_p maior a estabilidade do controle mas o tempo para o sistema convergir é maior ou pode mesmo não ocorrer.

2.4.2 O termo integral no controle

O simples uso do termo proporcional no controlador PID pode não alcançar o *ponto de referência* desejado, causando o que chamamos de *deslocamento*. Quando isso ocorre, o sistema se estabiliza em um ponto abaixo do valor do *ponto de referência*. A ação do termo integral $K_i \int_0^t e(t)$, permite ao controlador eliminar esse *deslocamento*. Enquanto que o termo proporcional considera apenas o erro num instante t, o termo integral considera o histórico do erro durante todo o processo. Sendo assim, mesmo o sistema se estabilizando com um *deslocamento*, o termo integral continuará agindo no controle para eliminá-lo.

2.4.3 O termo derivada no controle

O termo derivada $K_d \frac{de(t)}{dt}$, calcula a taxa de variação do erro no tempo multiplicado por um *ganho* K_d. Como o histórico de erro calculado pelo termo integral pode persistir mesmo quando o valor medido está próximo do *ponto de referência*. Isso faz com que o

Logo, esta seção abordará apenas os aspectos teóricos desse tipo de controlador. Para mais informações consulte http://www.controlguru.com termo integral continue atuando no controlador de forma intensa. Sendo assim, o termo derivada tem a função de diminuir essa taxa de variação, principalmente quando o valor medido está próximo do *ponto de referência*.

2.5 Reconhecimento de padrões

Reconhecimento de padrões tem por objetivo classificar um conjunto de dados a partir de um padrão ou de informações estatísticas. No nosso caso, existem determinados pontos no campo de futebol que devem bem ser conhecidos pela aplicação, como as marcas de pênalti, o meio do campo, entre outros. Essas áreas seguem um padrão específico e podem ser classificadas através de técnicas de reconhecimento de padrões. Abordaremos aqui o uso do SAD (*Sum of Absolute Differences*), ou soma de diferenças absolutas.

A aplicação do SAD em uma imagem é similar à aplicação dos filtros L.E.I vistos acima. Uma subimagem que contém o padrão específico para um tipo é comparada com uma região da imagem de mesmo tamanho. Nesta comparação, é tomado o valor absoluto da diferença entre cada pixel da imagem e da subimagem. Esses valores são somados (Eq. 2.16) e essa soma é usada como métrica de similaridade. Quanto menor a soma maior a similaridade entre as subimagens, ou seja, a região da imagem estará mais próxima do padrão que se quer encontrar.

$$S = \sum_{s=-a}^{a} \sum_{t=-b}^{b} |p(s,t) - f(x+s, y+t)| \tag{2.16}$$

Como exemplo, considere uma subimagem p (Fig. 11a) contendo um padrão relativo ao mostrado na Figura 11b. Varrendo p sobre a imagem da Figura 11c (na qual ou um pixel é preto ou é branco), podemos comparar os valores obtidos em cada aplicação do SAD. É fácil perceber que este valor será 0 quando o pixel central de p estiver sob o pixel central da figura geométrica em destaque em (c). Sendo 0 o menor valor possível dentre todos os valores obtidos pelo SAD aplicado na imagem, o padrão desejado foi então encontrado e o pixel central da figura geométrica é usado como coordenada para referenciá-la (Fig. 11d).

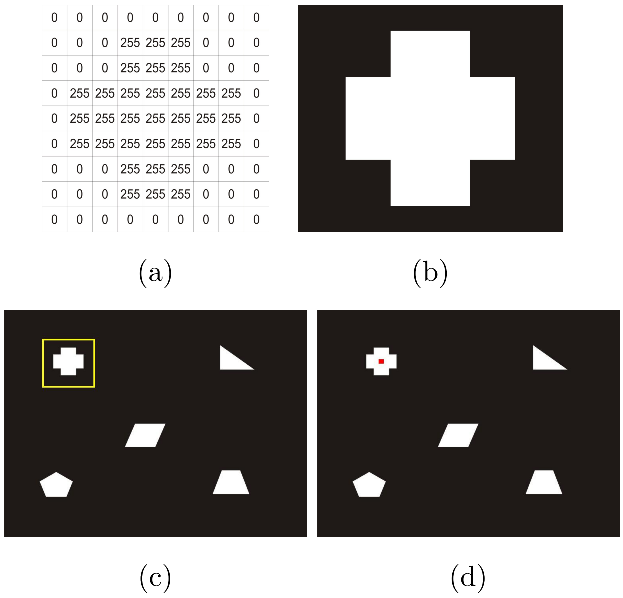


Figura 11: (a) subimagem p com o padrão. (b) marca de pênalti do campo. (c) detalhe da área que será varrida por p. (d) coordenada central do padrão encontrado.

3 Detecção e controle de robôs

Este capítulo abordará o ambiente em que ocorrem as partidas de futebol de robôs e a estrutura geral do fluxo de processamento do sistema de detecção e controle desenvolvido neste trabalho.

3.1 Ambiente

A partida de futebol de robôs acontece num campo pintado de preto e com linhas brancas. Sua textura é similar a de um campo de tênis de mesa e com dimensão de $150cm \times 130cm$. As marcações do campo bem como as medidas e dimensões do campo são mostradas na Figura 12. Essas dimensões devem ser reconhecidas pelo sistema de detecção.

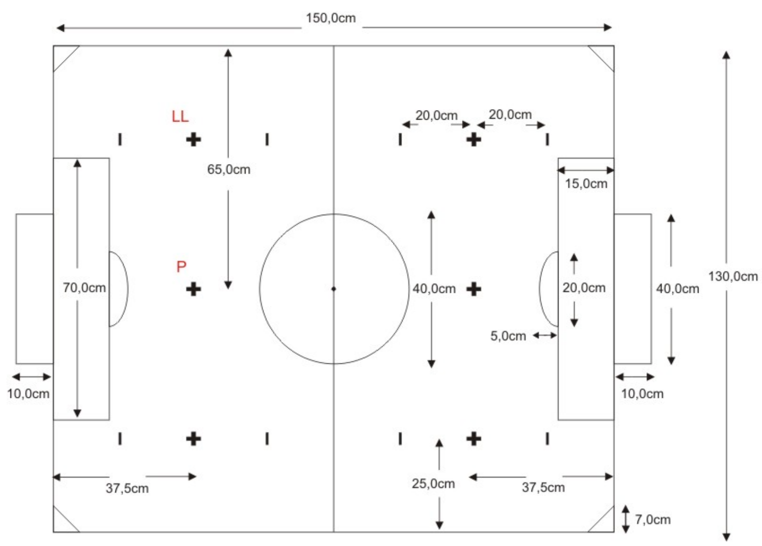


Figura 12: Dimensões e marcações do campo. P refere-se à marcação de pênalti e LL à marcação de lance livre (as barras ao lado são as posições onde dois robôs deverão ficar durante um lance livre).

A bola utilizada para as partidas deverá ser uma bola de golfe na cor laranja e a dimensão dos robôs deverá ser limitada a $7,5cm \times 7,5cm \times 7,5cm$ excluindo-se a antena. No topo de cada robô é colocada uma etiqueta de cor azul ou amarela, para identificação do time, e uma etiqueta de cor diferente das cores azul, amarelo, branco, preto ou laranja para identificação de cada robô. Essas etiquetas podem ser de qualquer forma, desde que possam conter um círculo de $4cm$ de diâmetro ou um quadrado de $3,5cm$ de lado. A luminosidade do ambiente deve ser fixada em aproximadamente $1000lx$ e uma câmera é posicionada perpendicularmente ao plano do campo a $2m$ acima do centro. Outros detalhes sobre as regras podem ser encontrados no site do ramo IEEE da UFJF no endereço *http://www.ieee.ufjf.br/reserved/archive/regras_futebol_autonomo.pdf.*

3.2 Estrutura geral

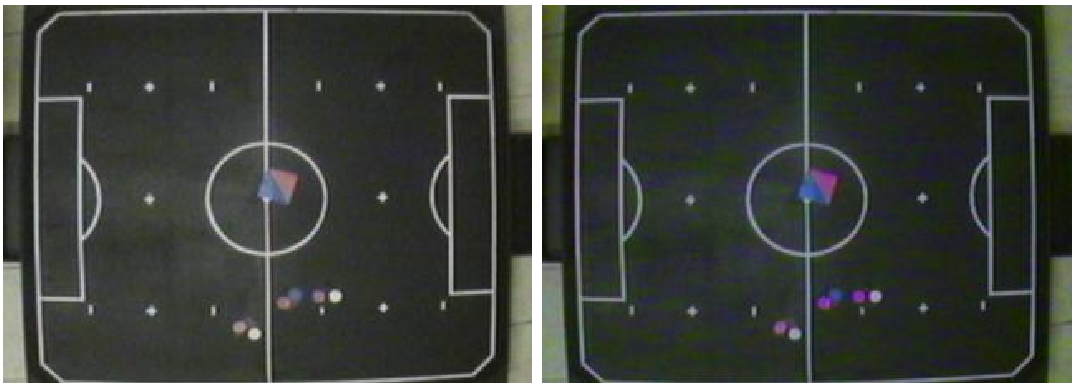
Uma câmera posicionada acima do campo é responsável por enviar ao sistema de detecção as imagens captadas. Essas imagens, chamadas *quadros*, são captadas a uma taxa de 30qps (trinta quadros por segundo) e possuem dimensão de 640×480 pixels. Para obtenção das imagens foi utilizada uma câmera modelo SH-312 CCD 1/3" SONY e uma placa de captura de vídeo XCapture da PixelView. O computador utilizado foi um Intel Xeon QuadCore 2.00GHz com 2.00GB de RAM.

O sistema usa 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.

A detecção dos robôs e bola é feita em três fases: inicialização do sistema, classificação das cores e detecção dos agentes da cena.

3.2.1 Inicialização do sistema

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 que mostra a visualização da câmera, um usuário deve selecionar as cores de interesse da cena que serão detectadas durante a partida (a cor da bola e as cores das etiquetas dos robôs). Essas cores também serão usadas para calcular a matriz de calibração de cores através de um objeto gcgColorCalibration. A Figura 13 mostra um quadro antes e depois de passar pelo processo de calibração de cores.



(a) (b)

Figura 13: (a) imagem antes da calibração de cores. (b) imagem depois da calibração de cores.

As coordenadas das marcações do campo também são detectadas na inicialização através de um objeto gcgFieldDetector. Como a câmera permanece estática durante toda a partida, essas marcações são detectadas somente uma vez. Primeiramente, é aplicado um filtro gaussiano 15×15 e o resultado é subtraído da imagem original (obtendo-se uma aproximação de uma filtro laplaciano), como mostra a Figura 14.

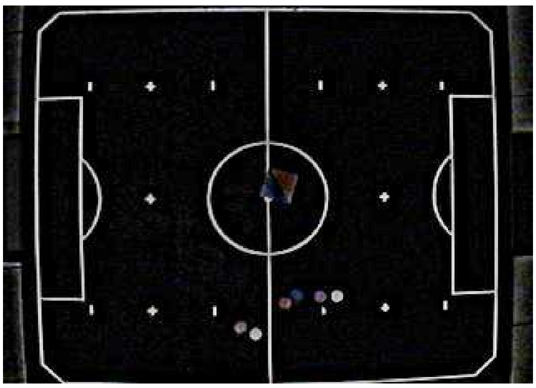


Figura 14: Detecção das altas frequências do campo.

Em seguida a função setFieldParameters() do objeto gcgFieldDetector utiliza esta imagem para aplicar o algoritmo de reconhecimento de padrões e detectar as coordenadas das marcações do campo. Esse algoritmo é executado um número determinado de vezes e a cada iteração as coordenadas são filtradas por um filtro de Kalman. Essa iteração é necessária para que o filtro possa se corrigir e obter uma estimativa melhor das coordenadas. Para que não seja preciso varrer toda a imagem, o que causaria um custo adicional de processamento, cada padrão é procurado em uma região da imagem mais provável de se encontrá-lo. As duas coordenadas da interseção do círculo central do campo com a linha central são encontradas primeiro, varrendo-se a região central da imagem (Fig. 15). A partir destas duas coordenadas, é encontrada a relação de pixels por centímetros, pois se conhece a distância entre essas coordenadas tanto em pixels quanto em centímetros. Com essa relação e as medidas conhecidas do campo, podemos calcular a coordenada aproximada de uma determinada marcação e definir ali uma janela de varredura. O padrão dessa marcação será então procurado nesta janela. A Figura 16 mostra o diagrama de fluxos desta operação. Para cada tipo de marcação no campo existe um padrão específico. A relação dos padrões usados no código se encontra no Apêndice A.

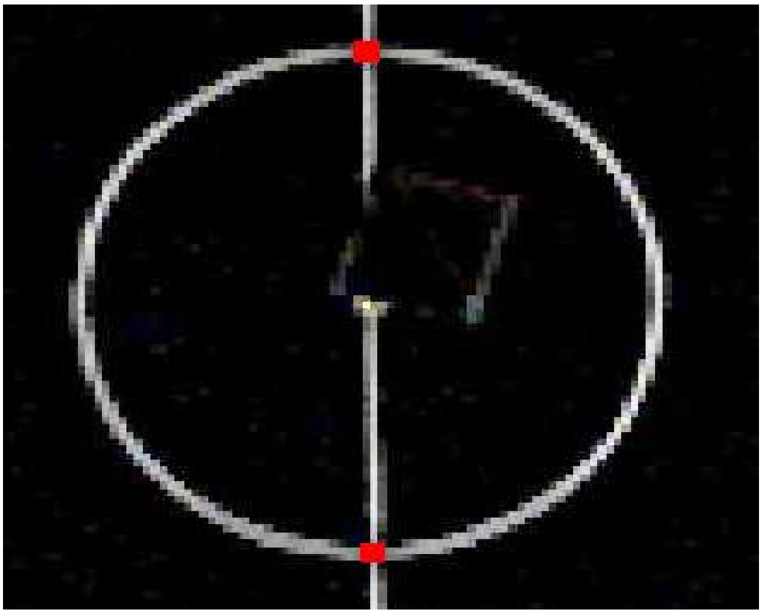


Figura 15: Reconhecimento das interseções da linha central do campo com o círculo central.

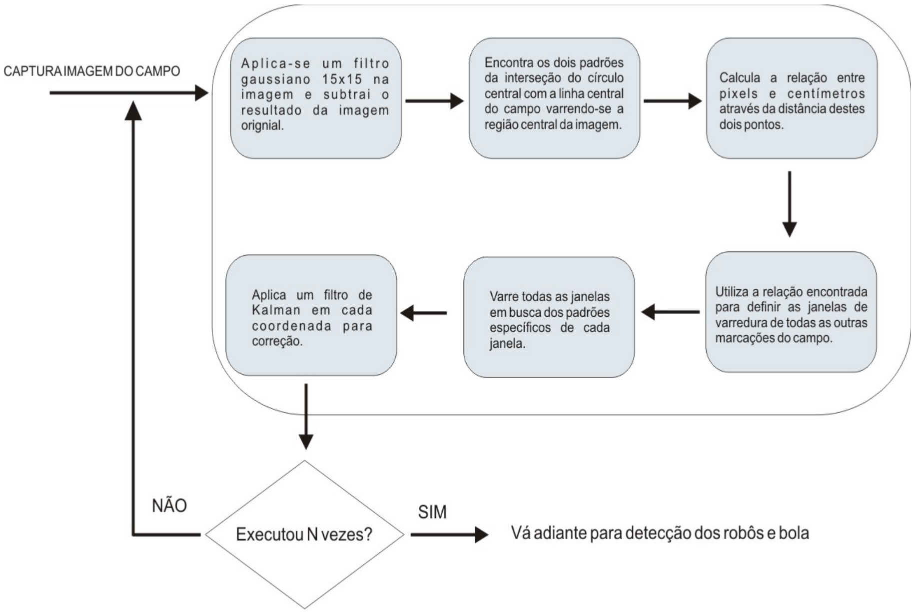
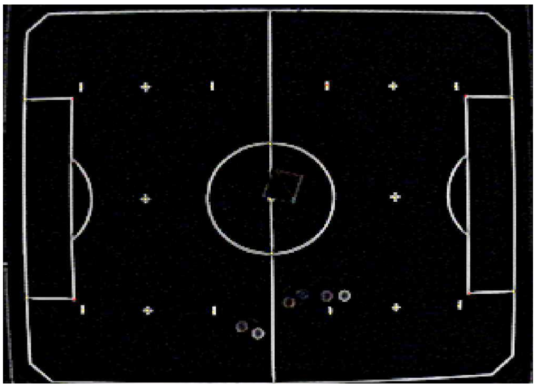
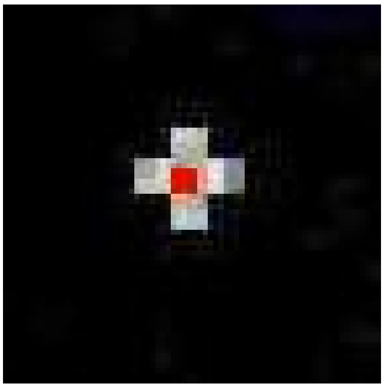


Figura 16: Fluxograma da aplicação do reconhecimento de padrões no campo.

Abaixo, a Figura 17 mostra as coordenadas dos padrões encontrados. Devido aos ruídos intrínsecos à câmera e ao ambiente, essa detecção é feita utilizando-se um filtro de Kalman.



(a)



(b)

Figura 17: (a) Detecção das marcações do campo. (b) detalhe da marca do pênalti direito, o ponto destacado é a coordenada estimada pelo filtro de kalman.

São criadas ainda nesta etapa, quatro threads para o processamento paralelo do sistema. A criação destas threads aumenta o desempenho geral do sistema quando a máquina tem quatro núcleos (que é o caso da máquina usada para este trabalho). Este ganho é mostrado na Tabela 1 no capítulo de resultados. Uma thread ficará responsável pela aplicação de um filtro triangular 3×3, para redução de ruídos, pela aplicação da matriz de calibração de cores em todos os pixels de um quadro e a classificação de cores dos mesmos. Uma thread será responsável por calcular as coordenadas e rotação dos agentes da cena e enviá-las para as threads dos times. As duas últimas threads são usadas pelos times, cada um com uma thread, para rodar seus algoritmos de estratégia e enviar os comandos para seus robôs através do objeto gcgCommunication. O fluxograma da Figura 18 mostra o funcionamento das threads.

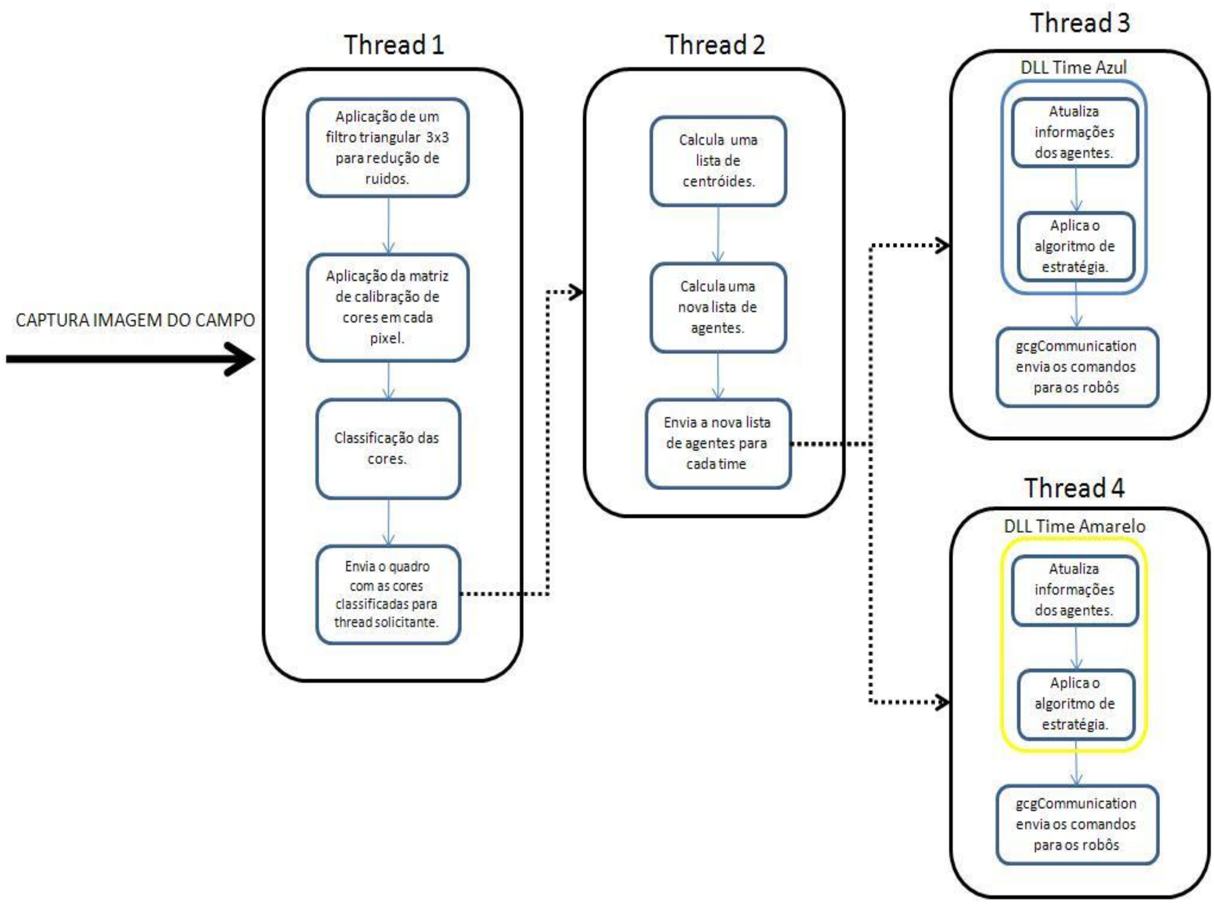


Figura 18: Fluxograma do funcionamento das threads.

A classe gcgCommunication, responsável por transmitir os comandos para os robôs, utiliza a API do Windows para enviar dados de oito bits para a porta serial especificada. Um transmissor de rádio ligado a esta porta enviará os dados para os robôs. Na construção de um objeto gcgCommunication devem ser passadas informações como a porta serial utilizada e a taxa de transmissão dos dados. Os dados são enviados através da função sendData(const char data) que tem como parâmetro uma cadeia de 8 bits. Os comandos enviados dependem da implementação dos robôs mas, geralmente, é enviada uma mensagem com o identificador do robô seguida de uma outra mensagem com a velocidade a ser aplicada. Devem ser criados dois objetos gcgCommunication, um para cada time.

3.2.2 Classificação das cores

Para que cada cor seja detectada e com isso sejam criadas regiões de cores associadas a um agente da cena, cada pixel deve ser associado a uma cor.

Nesta etapa, cada pixel é classificado segundo um conjunto de cores pré-determinadas, chamados *classe de cores*. Ele pode ser um pixel relativo à bola, a uma cor de etiqueta ou a nenhuma outra cor de interesse (sendo então descartado). Essa classificação possibilita subdividir regiões de pixels da imagem. Um objeto gcgColorSpace é responsável por essa etapa de classificação.

A ideia básica é classificar cada pixel de acordo com limiares de cores (KLANCAR et al., 2005). Três vetores (Fig. 19) de 256 posições de 32 bits são alocados, nos quais cada posição representa um nível de cor e cada vetor representa um canal de cor (R, G ou B). Uma classe de cor é representada por um bit em uma determinada posição da cadeia de 32 bits (KLANCAR et al., 2005). Podemos então classificar até 32 cores. Isso é mais do que suficiente para o nosso caso em que são usados, no máximo, seis robôs e uma bola, dando um total de até nove cores. Cada valor RGB de uma classe de cor representa um índice nos vetores. Em cada uma destas posições o bit correspondente a essa cor recebe o valor 1 e o restante recebe o valor 0. O mesmo é feito para as posições que estão dentro de um limiar especificado para cada canal desta cor, ou seja, uma classe de cor se estende para um conjunto de cores próximas a ela. Quando um pixel é analisado, verifica-se o seu valor RGB nos vetores e um AND lógico é feito com os valores das três posições dos vetores. O resultado é um valor de 32 bits no qual um bit numa posição n com valor 1 significa que o pixel foi classificado com a classe de cor correspondente à posição n do bit. O pixel então recebe o valor daquela classe de cor.

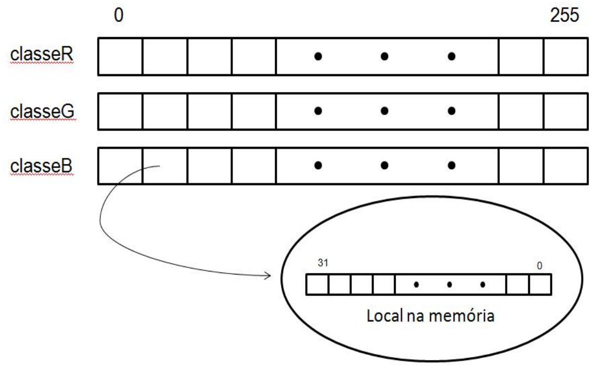


Figura 19: Vetores de classes de cores (KLANCAR et al., 2005). Cada classe de cor possui um bit que a representa dentre os 32 bits de cada posição nos vetores.

Como exemplo, suponha que queremos classificar os pixels com a classe de cor laranja $(R = 255, G = 127, B = 0)$ com seus valores sendo estendidos pelo seguinte intervalo:

$$235 \leq R \leq 255$$

$$115 \leq G \leq 145$$

$$0 \leq B \leq 25$$

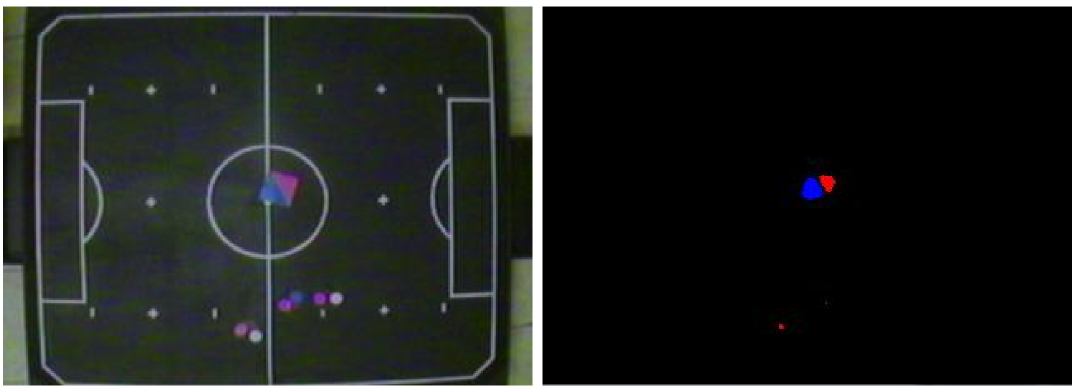
Suponha também que esta cor está associada ao bit 8 de cada posição do vetor. Neste caso, os valores que estão no intervalo $[235, 255]$ do vetor classeR, $[115, 145]$ do vetor classeG e $[0, 25]$ do vetor classeB tem seu valor no oitavo bit igual a 1. Quando um pixel for examinado, os índices desses vetores nas posições correspondentes a R, G e B do pixel são avaliados e se o resultado for 1 para a posição 8 do bit então o pixel é classificado como sendo laranja. A Figura 20 ilustra a operação em um pixel de cor R = 250, G = 125, B = 15. Em seguida, é apresentado o algoritmo de classificação.

```
classeR[250] = 00000000000110000000000100000000
                              AND
classeG[125] = 00000001001000001000000100000001
                              AND
classeB[15]  = 00001000000000001000000110000000
                               =
   Resultado = 00000000000000000000000100000000
```

Figura 20: Classificação de um pixel de cor R = 250, G = 125, B = 15. O oitavo bit da direita para esquerda representa a classe de cor laranja.

```
Para cada pixel da imagem
  classe = classeR[pixel.R] & classeG[pixel.G] & classeB[pixel.B];
  se classe & bit_classe_laranja então
    pixel.cor = laranja
  senão
    pixel.cor = preto;
  fim-se
fim-Para
```

A Figura 21 mostra a classificação dos pixels em duas classes de cores, azul e vermelho. A Figura 21a mostra a imagem antes da classificação e a Figura 21b mostra a imagem depois da classificação.



(a) (b)

Figura 21: (a) imagem antes da classificação dos pixels. (b) imagem depois da classificação dos pixels.

Após a classificação de todos os pixels é criada uma lista de regiões de pixels, que são aglomerações representando um objeto de interesse na cena (a bola ou uma etiqueta). Estas regiões são obtidas através de um algoritmo de região de crescimento nos pixels classificados. Para que uma região seja considerada válida é necessário que ela possua um número mínimo de pixels. Este número mínimo é pré-determinado e sua função é evitar que sejam criadas regiões que correspondam a ruídos da imagem. Na Figura 21, por exemplo, existem duas regiões de cores válidas, no centro da imagem, e uma região muito pequena logo embaixo que será descartada. Cada região encontrada é inserida numa lista de regiões que possui a informação de cor da região e as coordenadas de seu centróide, computado pela Equação 3.1. Essa lista será usada posteriormente para o cálculo das informações dos agentes.

$$C_r = \frac{1}{N} \cdot \sum_{i=1}^{N} (x_i, y_i), \tag{3.1}$$

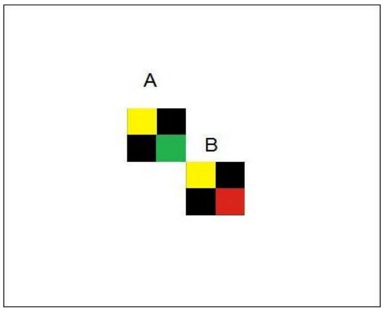
onde (x_i, y_i) é a coordenada do pixel i da região e N é o número total de pixels da região.

Apesar do procedimento para definir a classe de um pixel ser rápido, pois necessita apenas de um AND lógico e uma comparação com cada classe de cor, torna-se custoso computacionalmente ter que varrer toda a imagem a cada quadro. Como a mudança de posição de cada agente de um quadro para o outro é relativamente pequena, o desempenho pode ser melhorado varrendo apenas as áreas da imagem próximas a uma região que foi detectada no quadro anterior.

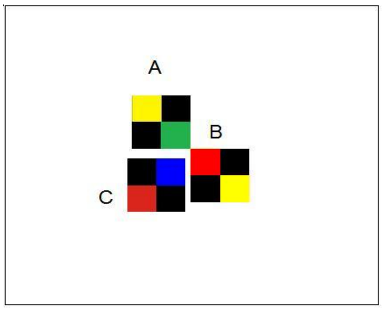
3.3 Detecção dos agentes da cena

A lista de centróides das regiões computadas na etapa anterior será usada nesta etapa para calcular as coordenadas e rotação dos agentes. Para as coordenadas da bola o processo é trivial, basta pegar as coordenadas do centróide da região correspondente à cor da bola. Para os robôs é necessário associar pares de regiões de forma que uma região referente a uma etiqueta de time se associe com uma região referente a uma etiqueta de robô. A grande questão é: quais pares de regiões correspondem a um mesmo robô?

A distância d entre os centróides de duas etiquetas de um mesmo robô é fixa e pode ser calculada na etapa de inicialização. Um agente é identificado quando temos duas regiões de tipos distintos (uma referente a uma etiqueta de time e outra a uma etiqueta de robô) e a distância d_{r1r2} entre seus centróides é $d - \delta \leq d_{r1r2} \leq d + \delta$, onde δ é um parâmetro que representa o desvio máximo que a distância calculada pode ter da distância d. Porém, pode haver regiões que não pertencem a um mesmo robô e ainda assim satisfazem à condição anterior. Por exemplo, considere as duas situações apresentadas na Figura 22. Na Figura 22a, a etiqueta verde do robô A e a etiqueta amarela do robô B satisfazem à condição de distância, possibilitando a identificação de um agente inválido. O mesmo pode ser observado na Figura 22b com as etiquetas azul do robô C e as etiquetas verde e vermelha dos robôs A e B, respectivamente.



(a)



(b)

Figura 22: Problemas de associação de regiões para formar agentes.

Para solucionar este problema fazendo com que cada par de regiões seja associado corretamente, é criado um grafo não-direcionado $G = (V, E)$ no qual os vértices são os centróides que representam as etiquetas dos robôs e as arestas representam a distância entre dois centróides. Dois vértices V_a e V_b possuem aresta E_{ab} se, e somente se, satisfazem a condição de distância citada acima. Caso algum vértice não possua nenhuma aresta, então ele é descartado pois, provavelmente, somente uma etiqueta do robô foi identificada pelo classificador de cores sendo impossível então determinar este agente no quadro corrente. Uma vez montado o grafo, o algoritmo para formação dos agentes remove do grafo o primeiro vértice que possuir apenas uma aresta e o seu adjacente. Esses vértices formam então um agente e o algoritmo continua até que não seja retirado nenhum par ou o grafo fique vazio. Se não for retirado nenhum par do grafo durante uma execução do laço, então pode ter havido ciclo. A solução apresentada neste trabalho para resolver o problema de ciclo é escolher um par de vértices adjacentes cuja distância estiver mais próxima de d, formar um agente com esse par e continuar com o laço. Um exemplo de formação de ciclos é ilustrado na Figura 23 no qual cada centróide possui dois vizinhos que satisfazem à condição de distância.

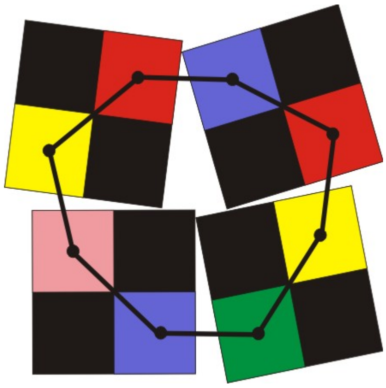


Figura 23: Formação de ciclo entre as regiões.

Após serem definidos os pares de regiões, é montada a lista de agentes. Nesta lista, cada agente carrega informação de sua coordenada, cor primária (etiqueta de time), cor secundária (etiqueta de robô) e rotação. No caso da bola, não há cor secundária e nem rotação. A coordenada da bola, conforme dito, é a própria coordenada do centróide da região que a representa. A coordenada C_R de um robô é tomada pelo ponto médio entre as coordenadas dos centróides que o representa:

$$C_R = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right) \tag{3.2}$$

A rotação do robô é calculada por (KLANCAR et al., 2005):

$$\varphi_r = \tan^{-1} \left(\frac{y_1 - y_2}{x_1 - x_2} \right) - \frac{\pi}{4}, \tag{3.3}$$

onde (x_1, y_1) são as coordenadas do centróide da etiqueta do time e (x_2, y_2) as coordenadas do centróide da etiqueta do robô.

3.4 Controle em tempo real dos robôs

O controle em tempo real dos robôs foi implementado utilizando o PID através da classe gcgCONTROL. Este controle é feito ajustando-se o valor das velocidades de suas rodas esquerda e direita de forma independente e em qualquer sentido. Para que um robô possa ir de sua posição e orientação atual a um ponto qualquer do campo ou girar um determinado ângulo, é necessário ajustar as velocidades de suas rodas. Para isso são utilizados dois objetos gcgCONTROL, um para a correção da posição e um para correção da rotação. Na Figura 24, para que o robô chegue até a bola é necessário corrigir seu ângulo e sua posição. O ângulo final do robô deve ser α e para isso ele deve girar um ângulo de θ_e na direção de α. A distâcia que o robô deve percorrer para que ele chegue até a bola é $\Delta d = \sqrt{d_x^2 + d_y^2}$ (GUPTA et al., 2004). É passado então para o controlador do ângulo o valor do erro angular θ_e e para o controlador da posição o valor do erro da distância Δd. As velocidades das rodas são calculadas por cada controlador, somadas e passadas para o sistema de comunicação. No próximo quadro o cálculo é refeito e o procedimento se repete até que ambos os erros sejam próximos de zero.

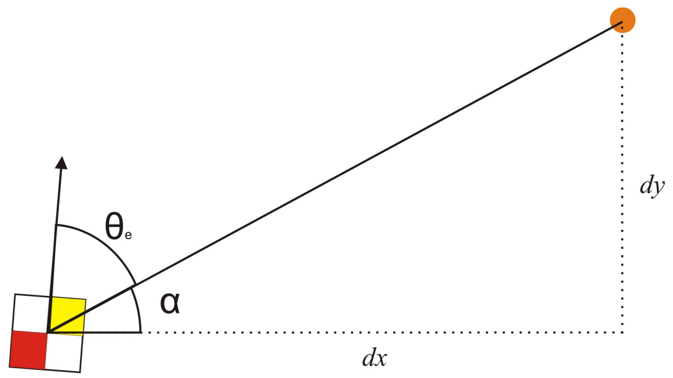


Figura 24: Erro entre o ângulo e posição atuais do robô e o ângulo e posição desejados.

Os robôs desenvolvidos pelos alunos de Engenharia Elétrica da UFJF foram especificados para serem controlados por operadores humanos através de joysticks, sendo que não é possível ter o controle independente das velocidades de suas rodas de acordo com as regras do futebol de robôs autônomo. Por conta disso, a classe de controle em tempo real foi testada no simulador RobotSoccer v1.5a da FIRA. Neste simulador é recriado o ambiente de uma partida de futebol de robôs e consiste de um servidor com o ambiente (campo, robôs, bola, placar, etc.) e dois clientes que representam as estratégias dos times. A sequência de imagens da Figura 25 mostra um robô do time azul perseguindo a bola em movimento.

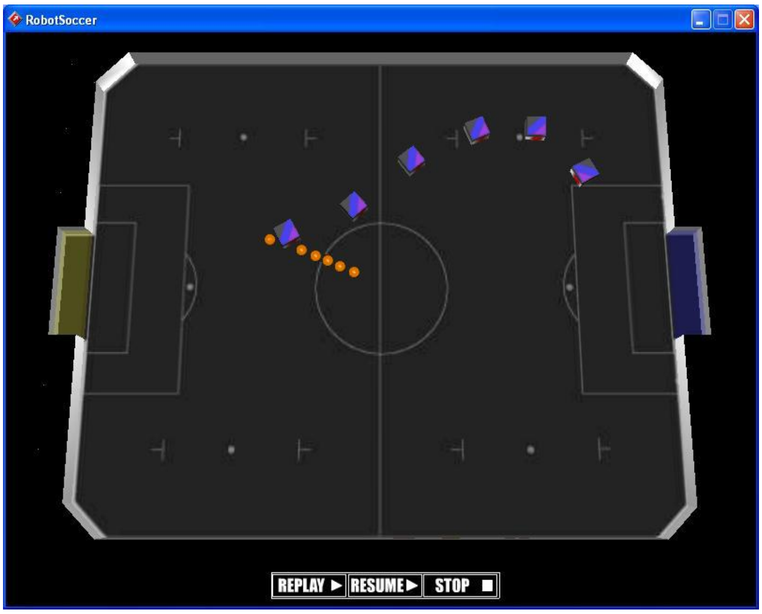


Figura 25: Robô do time azul perseguindo a bola no simulador da FIRA.

4 Resultados

Neste capítulo são mostrados os resultados obtidos com o programa desenvolvido durante este trabalho. Foi utilizado para os testes uma bola de tênis de mesa na cor verde e um robô desenvolvido pelos alunos de Engenharia Elétrica da UFJF com uma etiqueta de time azul e uma etiqueta de robô vermelha.

É apresentado abaixo a sequência de figuras representando as etapas do processamento de imagens. A primeira etapa trata da inicialização do sistema no qual são detectados os marcadores do campo. Esta etapa é executada apenas uma vez.

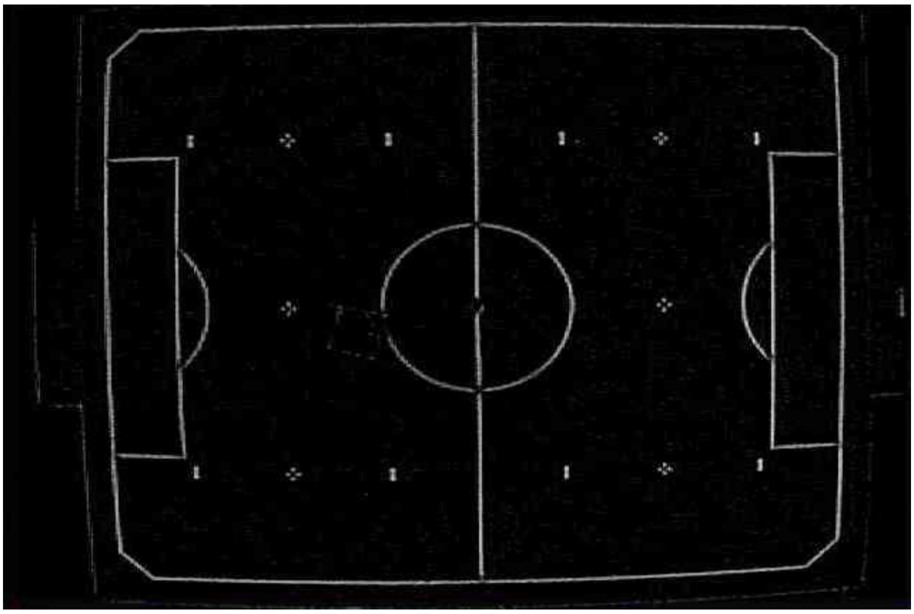


Figura 26: Subtração da imagem original pela imagem filtrada com passa-baixa.

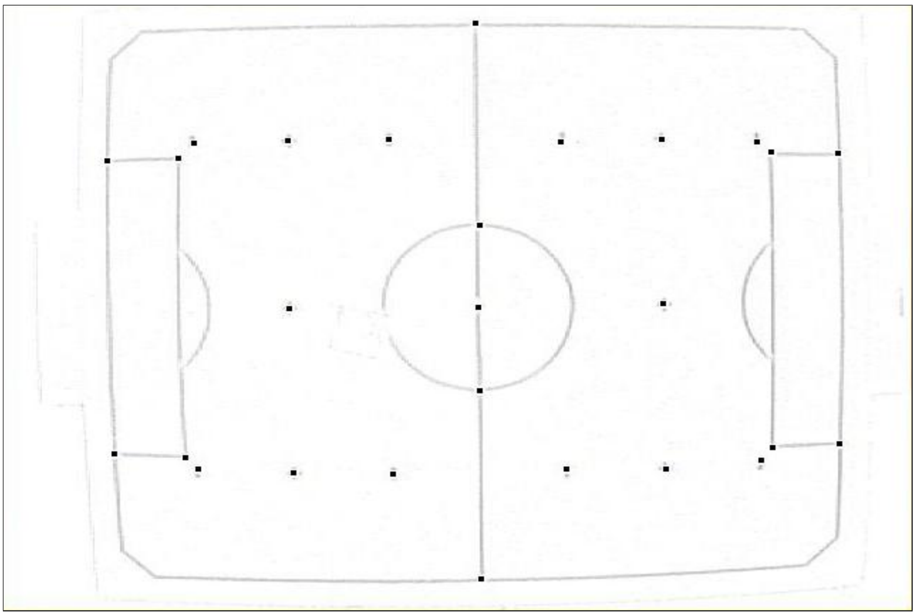


Figura 27: Detecção dos marcadores do campo. A imagem foi invertida e os pontos realçados para melhor visualização.

As etapas seguintes são executadas durante todo o tempo e tratam os quadros vindos da câmera para detecção dos robôs e da bola.

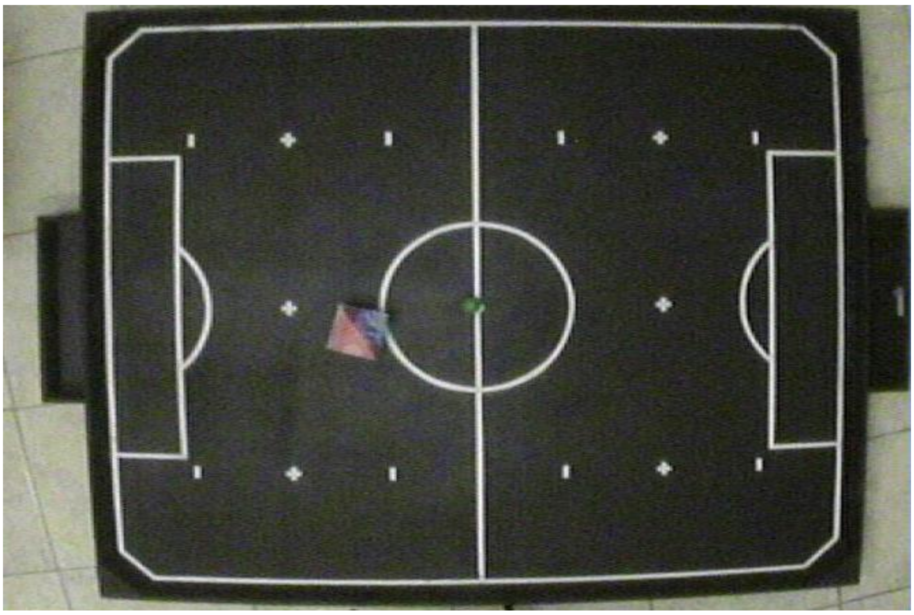


Figura 28: Imagem sem processamento entregue por um objeto gcgVIDEO.

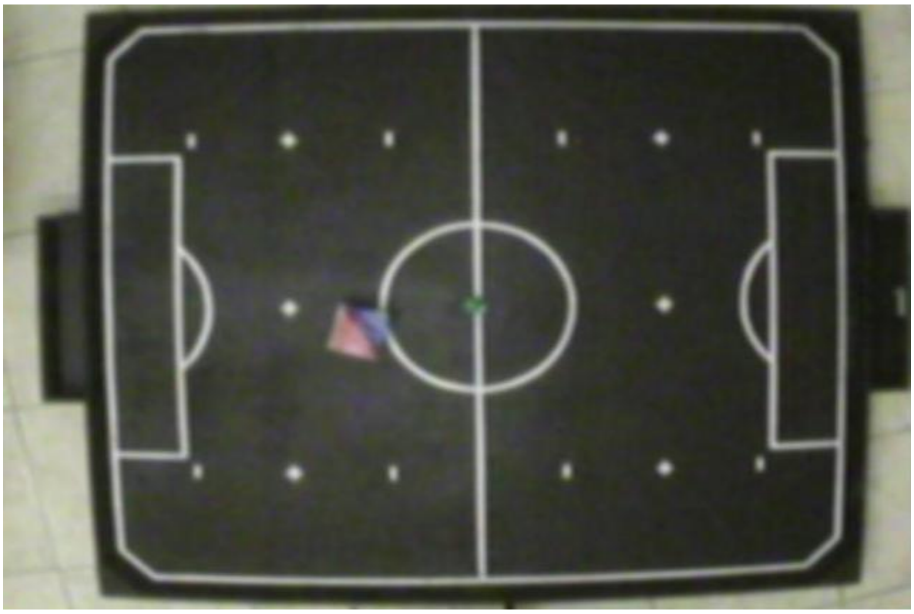


Figura 29: Filtragem passa-baixa para redução de ruídos da imagem.

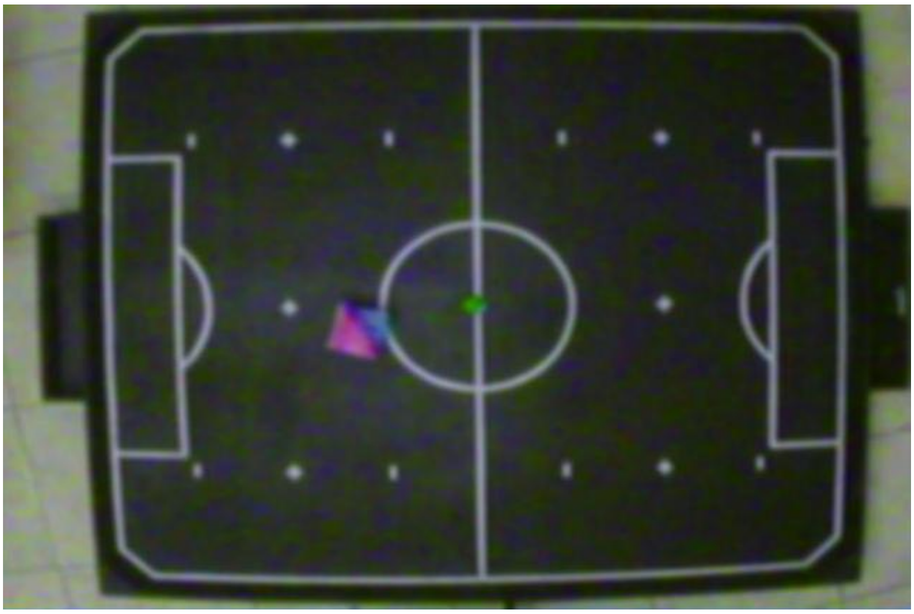


Figura 30: Transformação de cores pela matriz de calibração.

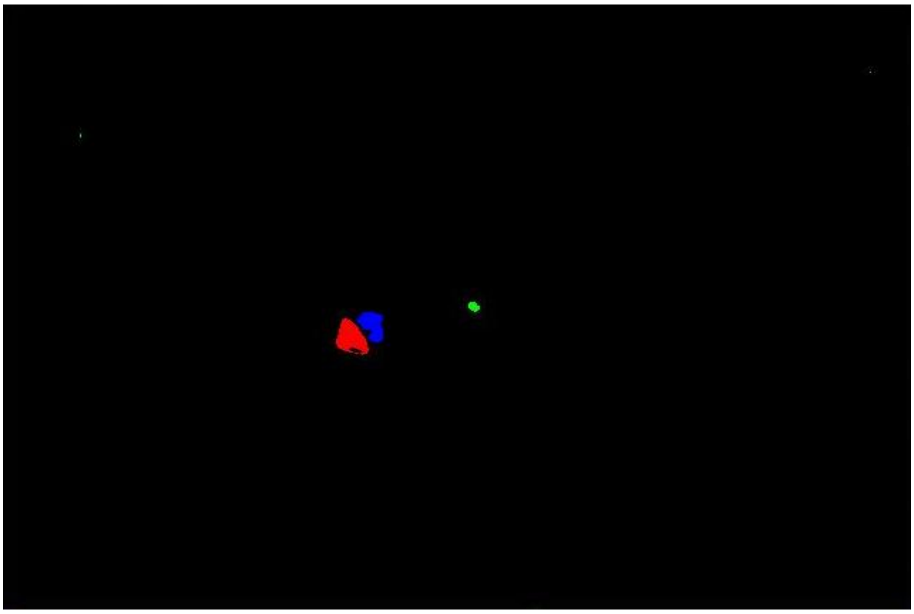


Figura 31: Classificação de pixels e detecção das regiões de interesse.

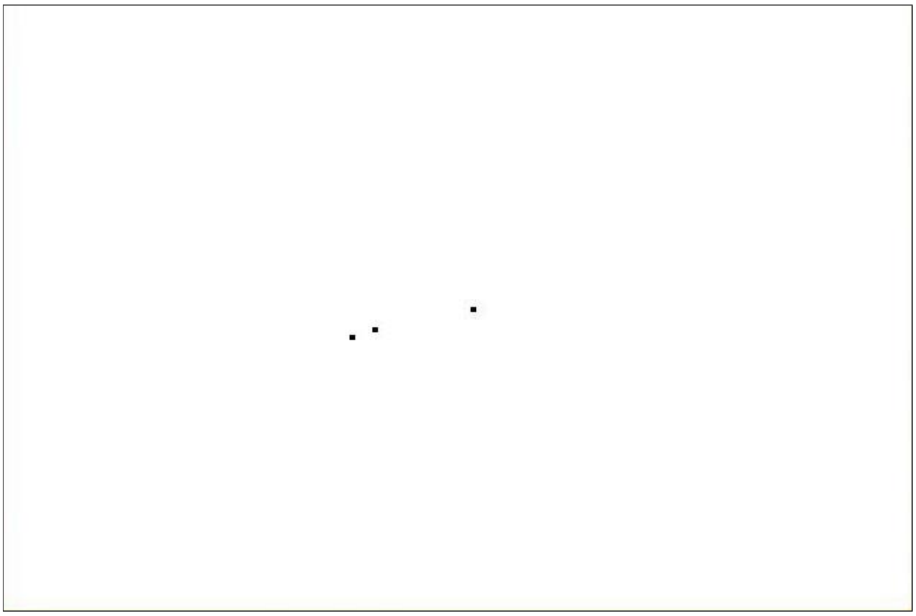


Figura 32: Cálculo do centróide de cada região. A imagem foi invertida e os pontos realçados para melhor visualização.

A tabela abaixo mostra o tempo médio de execução de cada etapa separadamente. Também é apresentado o tempo de execução sequencial e paralelo do processo.

Processamento	**Média**	**Desvio Padrão**
Detecção do campo	124ms	$4,765988509$
Transformação das cores	54ms	$7,851886475$
Classificação das cores	28ms	$6,647730683$
Cálculo dos centróides	19ms	$6,517071599$
Thread 1	83ms	$7,196653263$
Thread 2	20ms	$7,019050556$
Execução sequencial	101ms	$7,825779724$

Tabela 1: Tempo médio de execução de cada etapa e de todo o processo.

O tempo total da execução do processo em paralelo pode ser dado pela Thread 1 (pois é a thread com maior tempo). Sendo assim, houve um ganho de tempo de 18ms em relação à execução sequencial. Esse tempo pode ser melhorado ainda mais através de um melhor balanceamento entre as threads 1 e 2.

Após estas etapas foram calculadas as coordenadas da bola e do robô além da rotação deste. Essas informações foram enviadas para thread do time azul que utilizou uma estratégia simples para, apenas, seguir a bola. As velocidades a serem atribuídas a cada roda foram calculadas por um controlador PID e enviadas até o robô por um transmissor de rádio ligado à porta serial. Devido às diferenças de protocolos de operação entre o robô e a estratégia desenvolvida (que seguiu o padrão da FIRA) o robô não respondeu da forma esperada, tendo perseguido de fato a bola apenas algumas vezes. Outras restrições que impediram o funcionamento eficiente do sistema foram a latência e a baixa qualidade da imagem captada. A latência promoveu um grande atraso no processamento das imagens e a baixa qualidade das imagens captadas dificultou na detecção dos agentes em algumas posições do campo e impediu a classificação de algumas cores, como o amarelo e o laranja, por exemplo.

5 Conclusão

Foram vistos os conceitos básicos necessários para realização de uma partida de futebol de robôs autônomos, partindo da captura das imagens pela câmera até o controle em tempo real dos robôs. Este trabalho permite o estudo de técnicas de diversas áreas da ciência da computação e engenharias.

Os princípios da aplicação de filtros lineares espacialmente invariantes foram apresentados e exemplificados. Estes filtros permitem a extração de ruídos das imagens e a detecção de regiões de altas frequências.

As cores sensíveis à câmera foram transformadas em um conjunto de cores conhecido através do conceito de calibração de cores. Foi mostrado como utilizar a pseudo-inversa para encontrar a matriz de calibração de cores. A aplicação dessa matriz nas imagens possibilita diminuir a variação de brilho nas cores, aumentando a capacidade de detecção das mesmas.

Para filtragem e estimação dos ruídos intrínsecos à câmera e ao ambiente, foi apresentado o filtro de Kalman e mostrado o funcionamento de seu algoritmo. Através deste filtro é possível minimizar o erro na estimação das coordenadas calculadas pelo sistema.

O controle dos robôs, feito através do ajuste das velocidades de suas rodas, é alcançado através de um controlador PID. Este controlador permite ajustar as velocidades das rodas de um robô a partir do erro entre o estado atual e seu estado desejado.

O reconhecimento de padrões é uma etapa importante para detecção das marcações do campo. Foi estudado o SAD (*Soma de Diferenças Absolutas*) para que, a partir de um padrão pré-definido, fossem encontradas as coordenadas de todas as marcações do campo.

A partir dos fundamentos matemáticos apresentados, foi desenvolvido o sistema de detecção e controle em tempo real dos robôs. Foi apresentado o ambiente em que ocorre uma partida de futebol de robôs, as etapas de inicialização e as etapas de detecção dos robôs. O sistema usa processamento paralelo através do uso de threads, onde cada thread é responsável por uma etapa do processo. Uma thread cuida do processamento das imagens e reconhecimento das coordenadas de cada região de cor. A segunda thread trata as coordenadas encontradas pela thread anterior para calcular as coordenadas e rotação dos robôs e as coordenadas da bola. As duas últimas threads processam a estratégia de cada time. Como foi usada uma máquina com processador de quatro núcleos, cada thread pode ser disparada em um núcleo e, desta forma, obteve-se um ganho de desempenho.

Por último foram apresentados os resultados obtidos através de um teste feito com um robô e uma bola. Esse teste mostrou alguns problemas como a latência da câmera e do processamento e a baixa qualidade das imagens captadas.

Para trabalhos futuros pretende-se corrigir os problemas de hardware encontrados para que sejam obtidos resultados mais precisos e eficientes. A inclusão de técnicas de inteligência artificial para as estratégias também devem ser inseridas a fim de que ocorra uma partida de futebol com todos os robôs e de acordo com as regras oficiais. Também deve ser implementado um filtro de Kalman estendido para predição e estimativa dos agentes da cena.

APÊNDICE A – Máscaras e padrões utilizados

Neste apêndice são mostradas as máscaras utilizadas na filtragem das imagens e os padrões para o reconhecimento.

Máscara do filtro triangular aplicado na imagem. Tamanho 3, centro 1. O filtro é aplicado primeiramente na horizontal e depois na vertical.

```
{ 1.0/4.0, 2.0/4.0, 1.0/4.0 }
```

Máscara do filtro gaussiano aplicado na imagem. Tamanho 15, centro 8. O filtro é aplicado primeiramente na horizontal e depois na vertical.

```
{ 2.0/60.0, 2.0/60.0, 3.0/60.0, 4.0/60.0, 5.0/60.0, 5.0/60.0, 6.0/60.0, 6.0/60.0, 6.0/60.0, 5.0/60.0, 5.0/60.0, 4.0/60.0, 3.0/60.0, 2.0/60.0, 2.0/60.0 }
```

Padrão da interseção do cículo central com a linha central do campo. Tamanho 13x13, centro (6, 6).

```
0.0, 0.0, 0.0, 0.0, 0.2, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.2, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.2, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.2, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0,
0.2, 0.2, 0.2, 0.2, 0.2, 0.5, 1.0, 0.5, 0.2, 0.2, 0.2, 0.2, 0.2,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.2, 0.2, 0.2, 0.2, 0.2, 0.5, 1.0, 0.5, 0.2, 0.2, 0.2, 0.2, 0.2,
0.0, 0.0, 0.0, 0.0, 0.2, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.2, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.2, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.2, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0
```

Padrão da interseção da área do gol direita com a linha de fundo e bico da área. Tamanho 13x13, centro (9, 6).

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.5, 0.5, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.8, 0.8, 0.8, 0.5, 0.5, 0.5, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.5, 0.5, 0.5, 1.0, 0.5, 0.0, 0.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.0, 0.0,
0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.5, 0.5, 0.5, 1.0, 0.5, 0.0, 0.0,
0.8, 0.8, 0.8, 0.5, 0.5, 0.5, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.5, 0.5, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0
```

Padrão da interseção da área do gol esquerda com a linha de fundo e bico da área. Tamanho 13x13, centro (3, 6).

```
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.5, 0.5,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.5, 0.8, 0.8, 0.8,
0.0, 0.0, 0.5, 1.0, 0.5, 0.5, 0.5, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8,
0.0, 0.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.0, 0.0, 0.5, 1.0, 0.5, 0.5, 0.5, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.5, 0.8, 0.8, 0.8,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.5, 0.5,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
```

Padrão da interseção da linha lateral inferior do campo com a linha central. Tamanho 13x13, centro (6, 4).

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0
```

Padrão da interseção da linha lateral superior do campo com a linha central. Tamanho 13x13, centro (6, 8).

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
```

Padrão da marca do pênalti e lance livre relativo à bola. Tamanho 15x15, centro (7, 7).

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.7, 1.0, 0.7, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
0.0, 0.0, 0.7, 0.7, 0.7, 0.7, 0.7, 1.0, 0.7, 0.7, 0.7, 0.7, 0.7, 0.0, 0.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0,
0.0, 0.0, 0.7, 0.7, 0.7, 0.7, 0.7, 1.0, 0.7, 0.7, 0.7, 0.7, 0.7, 0.0, 0.0,
0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.7, 1.0, 0.7, 0.5, 0.5, 0.5, 0.5, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
```

Padrão do lance livre relativo aos jogadores. Tamanho 11x17, centro (5, 8).

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.5, 0.7, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
```

Padrão do bico inferior da área do gol direito. Tamanho 13x13, centro (1, 3).

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
```

Padrão do bico superior da área do gol direito. Tamanho 13x13, centro (1, 9).

```
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
```

Referências

GOMES, J.; VELHO, L. *Fundamentos da Computação Gráfica*. [S.l.]: Instituto Nacional de Matemática Pura e Aplicada, 2008. ISBN 9788524402005.

GONZALEZ, R.; WOODS, R. E. *Digital Imaging Processing*. [S.l.]: Prentice Hall, 2002. ISBN 0201180758.

GUPTA, G. S.; MESSOM, G. H.; DEMIDENKO, S. Vision assisted measurement for optimization of robot motion and position control functions. *Instrumentation and Measurement Technology Conference*, 2004.

KLANCAR, G.; BREZAK, M.; MATKO, D.; PETROVIć, I. Mobile robots tracking using computer vision. In: *Proceedings of the International Conference on Electrical Drives and Power Electronics 2005*. [S.l.: s.n.], 2005.

WELCH, G.; BISHOP, G. *An Introduction to the Kalman Filter*. 2001. Curso ministrado no SIGGRAPH.