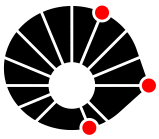

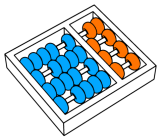

Detecção de objetos no futebol de robôs

Gabriel Magalhães *Esther Luna Colombini*

Relatório Técnico - IC-PFG-17-08
Projeto Final de Graduação
2017 - Junho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Detecção de objetos no futebol de robô

Gabriel Magalhães* Esther Colombini †

Resumo

O problema de *tracking* da posição de uma robô em uma cena é de extrema relevância para a robótica autônoma e a qualidade deste processo está instrisicamente relacionada à qualidade da extração de características e estimação da distância do robô aos elementos de interesse em um mapa conhecido. Neste contexto, este trabalho tem o objetivo de implementar um sistema de detecção de objetos – pertinentes ao domínio do futebol de robôs – capaz de identificar elementos de interessse em imagens capturadas por uma câmera e estimação da localização desses objetos em relação ao robô. Para o projeto o smiluador Webots foi utilizado assim como o modelo virtual do robô humanoide NAO. Foram estudadas e implementadas técnicas para detecção de objetos em baixo nível com base nas imagens coletadas e posteriormente foi feita a análise em mais alto nível para a classificação ou descarte de candidatos e a representação de objetos.

1 Introdução

1.1 A RoboCup

A RoboCup [1] é uma competição anual mundial de futebol disputada por robôs que serve como incentivo para o desenvolvimento e para o aprimoramento de técnicas e estudos na área de robótica e de inteligência artificial. Um objetivo final foi proposto para a competição: que até a metáde do século XXI um time de robôs humanoides totalmente autônomos fosse capaz de derrotar o time campeão da copa do mundo da FIFA, seguindo todas as regras do esporte impostas pela instituição. A competição é divida em categorias, e algumas delas serão detalhadas a seguir

No contexto da Robocup, o processamento e a análise das imagens extraídas pelas câmeras de cada robô são de extrema importância para a extração e detecção de objetos que possam ser úteis para fornecer informações a respeito da localização tanto do robô no campo quanto dos objetos em relação a ele. Essas informações são necessárias para fundamentar a tomada de decisão, seja para chutar a bola em direção ao gol ou para interceptar a ação de um oponente, por exemplo. Como a intenção final da competição é que os robôs consigam competir com humanos em campos de futebol reais seguindo as regras tradicionais do esporte, a cada ano as regras da Robocup são atualizadas e muitas vezes são removidas restrições que facilitariam a detecção de objetos por imagens mas que não

*Inst. de Computação, UNICAMP, 13083-852 Campinas, SP. ra138386@students.ic.unicamp.br
†Inst. de Computação, UNICAMP, 13083-852 Campinas, SP. esther@ic.unicamp.br

Figura 1: Exemplares do robô humanóide NAO

são compatíveis com o esporte original. Em geral objetos com cores que se destacam das demais do ambiente facilitam muito a sua detecção através de uma simples segmentação, e por isso originalmente a bola de futebol possuia a cor laranja e cada um dos gols era identificado por uma cor distinta (amarelo e azul). Havia também alguns indicadores coloridos em regiões interessantes do campo. Atualmente, essas restrições foram alteradas: os gols obrigatoriamente devem ser brancos e a bola não necessariamente tem um padrão definido (como ocorre no futebol disputado por humanos).

Com as dificuldades impostas pelas alterações nas regras da competição, muitos métodos precisam ser descartados (como os que se baseavam apenas em segmentação de cores) em favor de métodos que consigam se basear em formas geométricas, descritores ou mesmo de métodos de mais alto nível que empreguem Inteligência Artificial. Efetivamente, dois caminhos podem ser percorridos nas busca destas informações: Por um lado, há um extenso trabalho a respeito de técnicas de aprendizado de máquina que possibilitam que os robôes, com o treinamento adequado, consigam extrair significado das imagens da câmera de forma muitas vezes bastante eficiente. Do outro lado, as estratégias de baixo nível representam técnicas computacionalmente menos custosas e, por vezes, menos eficientes que as primeiras. Estas últimas serão o foco de análise deste trabalho devido as limitações computacionais impostas a certas ligas da RoboCup.

A liga **Humanoide** representa uma categoria onde os robôs possuem corpos e movimentação semelhantes aos de humanos. Nessa categoria, além dos desafios de estratégia e percepção visual há a dificuldade de movimentação, já que o robô deve ser capaz de se equilibrar, caminhar e se levantar após quedas, por exemplo. Essa categoria possui três subcategorias de acordo com o tamanho dos robôs: *Kid*, *Teen* e *Adult*.

Na categoria **Standard Platform** os dois times devem utilizar o mesmo modelo de robô, variando apenas a implementação de seu software. Atualmente o robô padrão é o NAO, da Aldebaran, que pode ser visualizado na Figura 1.

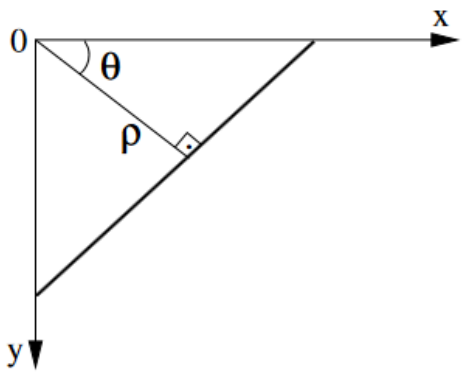

Figura 2: Representação da reta na forma polar.

1.2 Definições e conceitos

1.2.1 Transformada de Hough

Na tentativa de identificar objetos em uma imagem bidimensional, é comum ser necessária a detecção de formas geométricas predefinidas, como retas e círculos. Nesse contexto, a Transformada de Hough é uma técnica vastamente utlizada na extração de características que utiliza um algoritmo de voto para mapear instâncias muitas vezes imperfeitas das formas geométricas procuradas para uma equação paramétrica bem definida.

O exemplo mais simples de uso da Transformada de Hough é a detecção de linhas. Uma reta pode ser representada pela equação $y = mx + b$, que corresponde ao ponto $(m.b)$ no espaço de parâmetros. Todos os pontos que são colineares no plano da imagem se interceptam no mesmo ponto no espaço de parâmetros. Como retas verticais (com inclinação tendendo ao infinito) gerariam problemas nessa representação, é usada a equação de reta na forma polar (Figura 2):

$$\rho = xcos\theta + ysen\theta$$

onde ρ é a distância da origem à reta e θ é o ângulo formado entre a reta perpendicular e o eixo x.

Com a representação na forma polar, pontos colineares no espaço (x, y) correspondem a curvas senoidais que se interceptam no espaço (ρ, θ). A ideia do algoritmo de detecção de retas é discretizar o espaço (ρ, θ) e para cada ponto da imagem calcular os valores de ρ e θ que incluam o ponto, com a precisão sendo regulada pelo número de subdivisões no plano ro teta. Para cada valor de (ρ, θ) englobando o ponto é incrementado um respectivo acumulador. Os acumuladores com maior valor indicam os melhores candidatos a reta. O Algoritmo 1 sintetiza o que foi explicado.

A transformada de Hough pode ser utilizada de forma análoga para a detecção de círculos, porém o número de parâmetros necessários aumenta.

Algorithm 1 Detecção de retas pela transformada de Hough [15]

1: **procedure** HOUGHLINETRANSFORM
2: Discretizar o espaço de parâmetros (ρ, θ) em intervalos finitos. Cada célula M(ρ, θ) no espaço de parâmetros corresponde a um acumulador.
3: Inicializar todas as células do acumulador com zero.
4: Para cada ponto (x,y) no espaço da imagem, calcular os valores de ro e teta que satisfazem a equação da reta.
5: Incrementar em 1 o acumulador M(ρ, θ).
6: Após a determinação dos parâmetros de todos os pontos no espaço da imagem, os pontos de máximo da matriz de acumulação indicam forte evidência de retas na imagem.

$-\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$
0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

$-\frac{1}{3}$	0	$\frac{1}{3}$
$-\frac{1}{3}$	0	$\frac{1}{3}$
$-\frac{1}{3}$	0	$\frac{1}{3}$

Figura 3: Máscaras de Prewitt para obtenção do gradiente.

1.2.2 Detecção de bordas

No processo de análise de uma imagem, uma maneira de identificar os limites entre candidatos a objetos detectados e de reduzir a quantidade de informação a ser processada é aplicar um operador de bordas. Uma borda é um limite ou fronteira entre duas regiões com níveis de cinza suficientemente diferentes. As mudanças de níveis de cinza ao longo de uma imagem podem ser quatificadas através de derivadas, e muitas técnicas utilizam o vetor gradiente, que aponta na direção da mudança de maior intensidade e que pode ser calculado através da aplicação de certas máscaras como as de *Roberts* e de *Prewitt* [9] sobre a imagem.

Uma das técnicas mais difundidas para a detecção de bordas é a elaborada por *Canny* (1986), que procura otimizar a localização de pontos da borda na presença de ruído. O primeiro passo dessa técnica é suavizar a imagem original aplicando um filtro Gaussiano, que colabora para a redução de certos ruídos. Em seguida, a direção e o módulo do gradiente são calculados ao longo da imagem atarvés das máscaras mostradas acima. Com o cálculo do gradiente, a borda é localizada considerando os pontos cujo módulo seja localmente máximo na direção do gradiente. Esse último passo é chamado de *supressão não máxima*, que atua na redução da espessura das bordas. Nessa etapa as bordas estão detectadas, mas há a possibilidade de certos fragmentos de ruído ainda estarem presentes. O que a técnica realiza para solucionar esse problema é estipular dois limiares durante a etapa da supressão não máxima: T1 e T2, com $T2 > T1$. Os pontos da borda que possuem gradiente maior que T2 são mantidos. Outros pontos conetados a estes primeiros são considerados

Figura 4: Exemplo de resultado da aplicação do Operador de Canny.

parte da borda apenas se o módulo de seu gradiente estiver acima de T1. O resultado da aplicação da técnica é uma imagem binária com as bordas da imagem representadas pelos pixels brancos, conforme mostrado na Figura 4.

1.2.3 Operadores morfológicos

A morfologia matemática é uma metodologia para analisar imagens que permite a elaboração e utilização de operadores para a descrição de objetos em imagens. Entre as aplicações dos operadores morfológicos estão: extração de componentes conexos, busca de padrões em imagens, delimitação de fecho convexo, extração de bordas dos objtos e afinamento de bordas. A base da morfologia matemática é a teoria de conjuntos. Uma imagem binária pode ser considerada uma coleção A de coordenadas em que o valor do pixel é 1.

Um operador morfológico relaciona um conjunto A que representa uma imagem a conjunto B, denominado elemento estruturante, que é expresso com respeito a uma origem local. Os dois operadores básicos são o de dilatação e o de erosão. O operador de dilatação pode ser definido por:

$$D(A,B) = A \oplus B = \bigcup_{b \epsilon B} (A + b)$$

enquanto o de erosão é definido por:

$$\varepsilon(A,B) = A \ominus B = \bigcap_{b \epsilon B} (A + b)$$

A dilatação da imagem pode ser obtida da seguinte forma: posicione a origem do elemento estruturante em cada pixel da imagem original, e adicione todos os seus pixels à imagem final. A erosão da imagem pode ser obtida posicionando o elemento estruturante em cada pixel e, se o elemento inteiro couber na região delimitada pela imagem original, o pixel da origem é mantido na imagem final. Caso contrário o pixel da origem é retirado da imagem. A Figura 5 apresenta exemplos de operadores desta natureza.

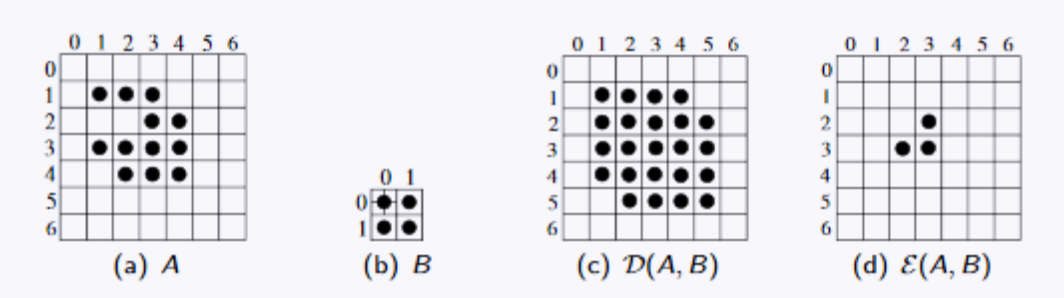

Figura 5: Exemplo de aplicação de operadores de dilatação e de erosão [15]

2 Revisão Bibliográfica

O problema da identificação de objetos através da imagem capturada na área da robótica é frequentemente abordado em publicações, especialmente no contexto da RoboCup. Grande parte da literatura sobre a competição acaba ficando rapidamente obsoleta em certo nível, já que as restrições e regras da competição são alteradas anualmente. No entanto, as estratégia utilizadas e descritas nesses trabalhos continuam sendo úteis para a referência e para o desenvolvimento de novas abordagens. Um exemplo foi a abordagem de Krishnan, Aswath, e Udupa [2], que utilizou a segmentação de cores combinada com a transformada de Hough para a detecção da bola (no período em que a bola utilizada ainda era laranja). No texto a detecção dos gols também foi orientada principalmente à distinção de cores, já que eles apresentavam cores bem definidas (azul e amarelo). As traves eram identificadas por 4 linhas detectadas pela transformada de Hough e a proporção entre os lados do retângulo que envolvia o gol detectado era utilizada para estimar a posição relativa da câmera do robô em relação ao objeto.

A abordagem de Hartl, Visser, e Rofer [3] baseia toda a detecção de objetos na segmentação por semelhança de cores. A escolha de não se utilizar valores pré-definidos torna o sistema mais robusto em relação a mudanças na luminosidade. Com a segmentação, neste trabalho a detecção é realizada pelo processo de aumento de região, em que são realizadas buscas comparando os valores das cores de cada pixel e agrupando em uma mesma região aqueles que são semelhantes de acordo com um limiar definido. Estivill-Castro e Radev [4] também combinaram a técnica de aumento de região com uma calibragem automática de cores que seria executada no início de uma partida para a definição robusta das cores que seriam utilizadas na segmentação. Neste trabalho foram empregadas estratégias de mais alto nível como o uso de Histogramas de Gradientes Orientados (HOG), que utiliza uma base de dados em conjunto com um classificador para detectar oponentes no campo de visão do robô.

No trabalho desenvolvido por Schulz e Behnke [7], focado na detecção eficiente das linhas do campo, é realizada a vetorização das linhas com base nos pontos detectados em imagens pré-processadas pela técninca de Esqueletização (que utiliza operadores morfológicas para reduzir a largura das linhas da imagem). A representação das linhas detectadas é feita

através de grafos, e a classificação dos cantos específicos do campo (escanteio, meio de campo, etc) é feita com base no grau dos nós referentes às junções no grafo.

3 Justificativa

No contexto da Robocup é imprescindível a localização e a detecção de objetos e de oponentes para a tomada de decisão, tornando interessante e relevante o estudo de técnicas de visão computacional aplicadas a esse fim. A configuração de um ambiente simulado que que comporte a visualização da imagem original capturada pela câmera do robô e suas segmentações, assim como a exibição em tempo real de objetos e formas detectadas em um computador convencional torna mais acessível o estudo e a experimentação nesse contexto. Sendo assim, além de um estudo introdutório das técnicas de visão computacional o trabalho pode oferecer uma estrutura inicial para facilitar trabalhos futuros através da configuração do simulador.

4 Objetivos

O objetivo do trabalho é compreender os desafios da competição Robocup, estudar as principais estratégias de visão computacional aplicadas a esse contexto e, com processos de baixo e alto nível, extrair e classificar objetos das imagens coletadas pela câmera do robô humanóide NAO em um ambiente simulado, em que seja possível a visualização em tempo real da detecção e da estimativa de localização dos objetos.

5 Desenvolvimento do Trabalho

5.1 Configuração do ambiente de simulação

A primeira etapa do trabalho envolveu a configuração de um ambiente que possibilitasse a execução de uma simulação coerente com as especificações descritas pelas regras da Robocup e a transmissão da imagem capturada pela câmera do robô para que o processamento fosse realizado e o resultado exibido.

O simulador escolhido foi o Webots [8], desenvolvido pela empresa Cyberbotics, que oferece uma versão pronta do robô NAO e modelos com os quais foi possível montar o cenário do campo de futebol (Figura 6). O Webots permite a criação de ambientes com propriedades físicas próximas à realidade e oferece uma grande variedade de sensores e atuadores que podem ser utilizados para coletar informações durante a simulação (como GPS e sensores de proximidade, por exemplo). O simulador é modelado em nós, conforme mostrado na Figura 7. Todos os sensores derivam da classe *Solid*, e a classe *Device* é abstrata e serve apenas para agrupar campos e funções utilizados na simulação. A classe *Solid* engloba todos os objetos com propriedades físicas, como dimensões e massa. Dessa forma, o nó *Robot* é derivado de *Solid* e para este projeto gera uma instância específica para o robô NAO cuja câmera é utilizada (Player) e um tipo especial de robô, o *Supervisor*, que consegue coletar dados globais da simulação, controlá-la e realizar operações como movimentação de outros

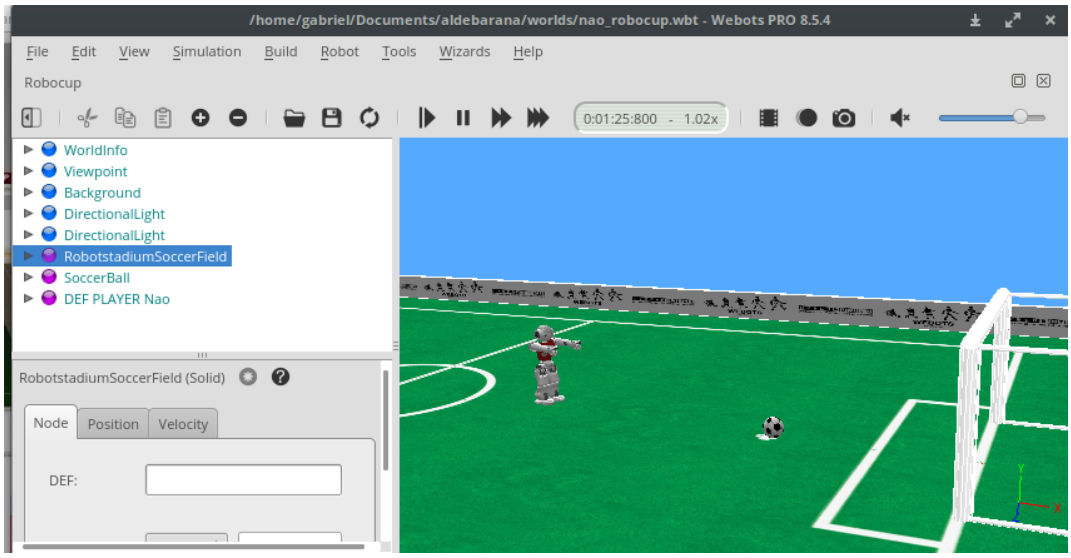

Figura 6: Exemplo de imagem capturada pela câmera do NAO no simulador

objetos (sejam eles robôs ou sólidos quaisquer). Para o projeto, o *Supervisor* foi utilizado nos experimentos de validação para posicionar o robôs em diversas localizações e coletar os dados de referência para serem comparados com os resultados obtidos pelo processamento da imagem da câmera. Para cada robô pode ser atribuído um controlador: um código que será executado junto ao início da simulação e que será encarregado de toda lógica executada pelo NAO simulado e pelo *Supervisor*.

Para o controlador do robô foi escolhido a linguagem Python. Com o uso da biblioteca de sockets, foi possível implementar a transmissão dos dados coletados pela chamada da função *getCameraMatrix()* e pelo sensor GPS do robô para um cliente externo que rodando a biblioteca OpenCV executa todo o processamento de imagem e os algoritmos de reconhecimento e classificação, exibindo em tempo real os objetos detectados. Para que a simulação fosse executada em tempo real com todo o processamento de imagens implementado, foi necessário reduzir a resolução de captura da câmera do robô para 160x120 (Figura 8).

5.2 OpenCV

A OpenCV (Open Source Computer Vision Library) é uma biblioteca lançada em 1999 sob a licensa BSD voltada para a área de visão computacional e de processamento de imagens [13]. Ela é escrita em C/C++ mas possui interfaces em outras linguagens, como Python, que foi a escolha para esse projeto. Suas aplicações envolvem reconhecimento facial, segmentação de imagens, realidade aumentada, reconhecimento de gestos, entre outras. Parte da biblioteca é voltada para o aprendizado de máquina, contendo protótipos e interfaces para a utlização de redes neurais, máquina de vetores de suporte e classificador de Bayes, por exemplo.

A biblioteca foi escolhida pois possui implementações prontas de diversos algoritmos como a detecção de formas por transformada de Hough, de diversos detectores de bordas,

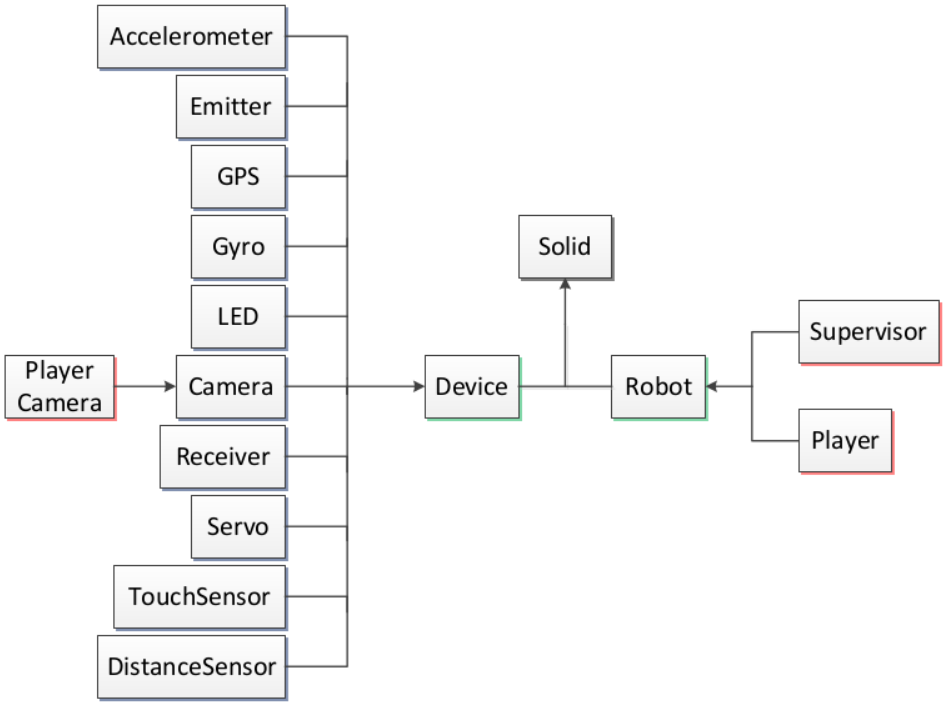

Figura 7: Estrutura dos nós do simulador Webots [12]

Figura 8: Exemplo de imagem capturada pela câmera do NAO no simulador

como o já mencionado elaborado por Canny e também ferramentas para exibição e manipulação de imagens, possibilitando a visualização e a anotação das imagens capturadas e processadas em tempo real.

5.3 Arquitetura do Sistema

Com a configuração do simulador e da estrutura cliente/servidor para obtenção e processamento de imagens, foi definida a arquitetura do sistema, resumida no fluxograma da Figura 9. A implementação realizada do lado do cliente (controlador do Webots) foi bastante simples: no controlador do robô NAO apenas realizamos a coleta da matriz de imagem da câmera do robô e dos dados de posicionamento do sensor de GPS e os enviamos para o servidor através da biblioteca de sockets padrão do Python. No controlador do Supervisor realizamos algumas iterações de reposicionamento do robô através do método *robot.getField('translation').setSFVec3f()* e do método *robot.getField('rotation').setSFRotation()*.

Todo o processamento de imagem é realizado no lado do servidor da aplicação, com uso da biblioteca OpenCV. O primeiro passo do processamento é a conversão da imagem gerada pelo câmera do robô simulado de RGB para HSV, o que facilita a segmentação por cores. Segmentando a imagem, quase sempre encontramos ruídos pontuais que dificultam bastante a detecção dos objetos, e por isso nessa etapa utilizamos suavizadores para eliminar parte deles e para eliminar um pouco dos efeitos de distorção que a baixa resolução da imagem acaba provocando nas formas capturadas. Nos casos de detecção de linhas do campo e dos gols, foi necessário aplicar operadores morfológicos para remover regiões que acarretavam na detecção de falso positivos e para conectar regiões que foram desconectadas devido a baixa resolução da captura.

As próximas etapas da detecção são específicas para cada tipo de objeto procurado, produzindo os valores de coordenadas e de dimensões que serão utilizados pelo último componente da aplicação, que baseado nas características da câmera realiza uma estimativa de localização dos objetos em relação à câmera. Essa estimativa posteriormente é comparada com os valores de referência e podemos assim avaliar o desempenho da implementação.

5.4 Escolha do modelo de cores e segmentação

A imagem extraída da câmera do robô no simulador utiliza o espaço de cores RGB que, apesar de ser o espaço mais utilizado na representação de imagens em computadores no cotidiano, complica o processo de segmentação pois é bastante sucetível a variações de luminosidade. A escolha para o projeto foi converter as imagens para o espaço HSV, abreviatura em inglês para *hue*(matiz) *saturation*(saturação) e *value* valor.

O HSV é uma representação cilíndrica do espaço de cores (Figura 10). Matiz define o tipo de cor, independentemente das condições de iluminação, variando do vermelho ao magenta. Esse valor é bastante informativo para distinguir cores e facilitou a segmentação (citar gudi2013). A conversão foi feita utilizando o método *cvtColor* do OpenCV.

Com base na experimentação de valores, foi possível montar imagens binárias segmentando áreas verdes e brancas para isolar as linhas do campo e os gols, como pode ser visto na Figura 11.

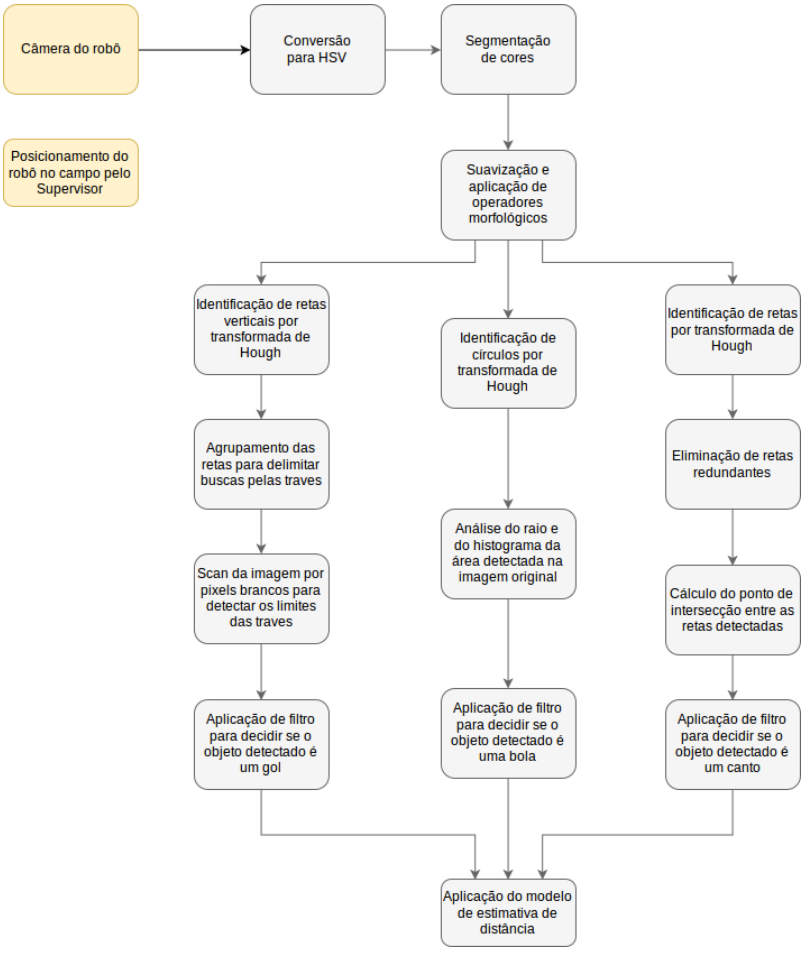

Figura 9: Fluxograma com a arquitetura do sistema.

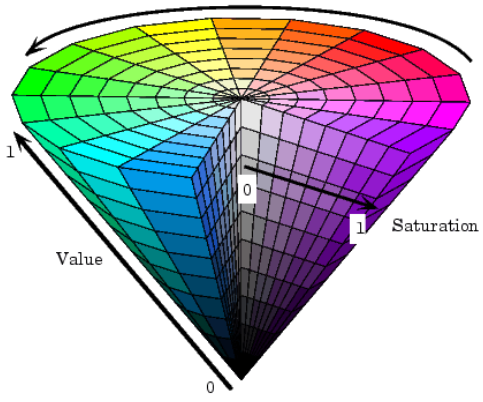

Figura 10: Representação cilíndrica do espaço HSV.

Figura 11: Exemplos do resultado da segmentação.

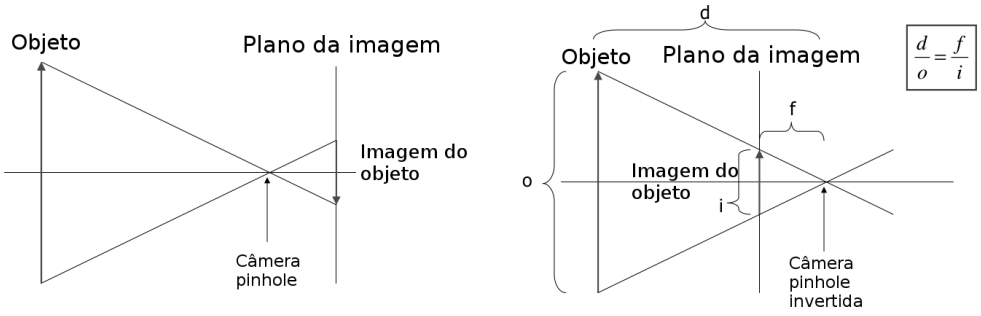

Figura 12: Modelo de câmera *pinhole*. [14]

5.5 Estimativa de distância dos objetos

A base para o modelo de estimativa de distância é a chamada câmera *pinhole*, um modelo que descreve o relacionamento matemático entre as coordenadas 3D do objeto no mundo real e a sua projeção no plano da imagem de uma câmera com a menor abertura de obturador possível (correspondente a um ponto). Com esse modelo, conhecendo a dimensão real de um objeto e a distância focal da lente da câmera utilizada é fácil determinar a distância do objeto em relação à câmera apenas pelo tamanho em pixels ocupado por ele. A Figura 12 apresenta esta configuração.

A câmera presente no robô NAO utilizado no projeto é equivalente a uma câmera com lente retilínea, isto é, ela preserva o aspecto das linhas que delimitam objetos retos como prédios e paredes, em vez de distorcê-los em formas curvas [11]. Da configuração da câmera, temos o campo de visão horizontal (hFoV) que pode ser usado para calcular a distância focal pela seguinte relação [10]:

$$hFoV = 2 * atan(\frac{largura}{2 * dist.focal})$$

Para obtenção do campo de visão vertical utilizamos a relação:

$$vFoV = 2 * atan(tan(\frac{hFoV}{2}) * \frac{altura}{largura})$$

O modelo de câmera pinhole deve ser ligeiramente modificado para reproduzir com maior exatidão as condições da câmera utilizada, já que a relação descrita na Figura 15 só vale no caso de a objeto estar exatamente no centro da imagem. Se não for o caso, a distância será um pouco maior pois devemos levar em conta o ângulo de deslocamento horizontal (AH), dado por:

$$AH = (\frac{x}{largura} - 0.5) * hFoV$$

onde x é a coordenada no eixo x do ponto analisado na imagem, e também o ângulo de elevação (AE), dado por:

$$AE = -(\frac{y}{altura} - 0.5) * vFoV$$

onde y é a coordenada no eixo y do ponto analisado.

5.6 Detecção dos Gols

Como para este projeto ainda não foram consideradas variações na iluminação, foi possível obter um intervalo de cores em que os dois gols brancos se sobressaíam levemente em relação às linhas do campo, possibilitando que os pixels referentes às partes do gol fossem isolados. Devido às características da câmera utilizada, podemos assumir que no momento em que o robô estiver em repouso as duas traves de um gol representarão segmentos de reta aproximadamente verticais.

Para identificar os limites do gol inicialmente foram realizadas varreduras por todos os pixels da imagem binarizada, mas essa abordagem se mostrou pouco eficiente e tornou o processamento bastante lento. A alternativa encontrada foi delimitar um espaço reduzido para essas varreduras através da detecção de linhas por Transformada de Hough. Configurando os parâmetros para detectar apenas segmentos aproximadamente verticais esperamos a detecção de 4 retas, referentes às duas bordas de cada trave. Realizando uma comparação entre as distâncias foi possível atribuir as retas à trave esquerda ou à direita.

As retas encontradas delimitam o espaço de varredura, que procura os pixel branco de maior e menor coordenadas y na imagem para cada trave, identificando o início e o fim de cada uma delas. A estrutura traseira do gol representou um problema na detecção, já que a transformada de Hough acabava traçando retas adicionais em cada lado de acordo com o ângulo de rotação do robô. Como a largura dessa estruturas era menor que a das traves procuradas, a aplicação do operador morfológico de erosão foi suficiente para manter apenas os objetos procurados na imagem. Para a classificação do objeto encontrado como o gol aplicamos um filtro [14] conforme as seguintes equações:

$$o = e^{-(\frac{value}{0.6})^2/2}$$

onde:

$$value = \frac{\frac{ratio}{2} \times (leftPoleLength + rightPoleLength) - topPoleLength}{\frac{ratio}{2} \times (leftPoleLength + rightPoleLength) + topPoleLength}$$

e

$$ratio = \frac{1.6}{0.85}$$

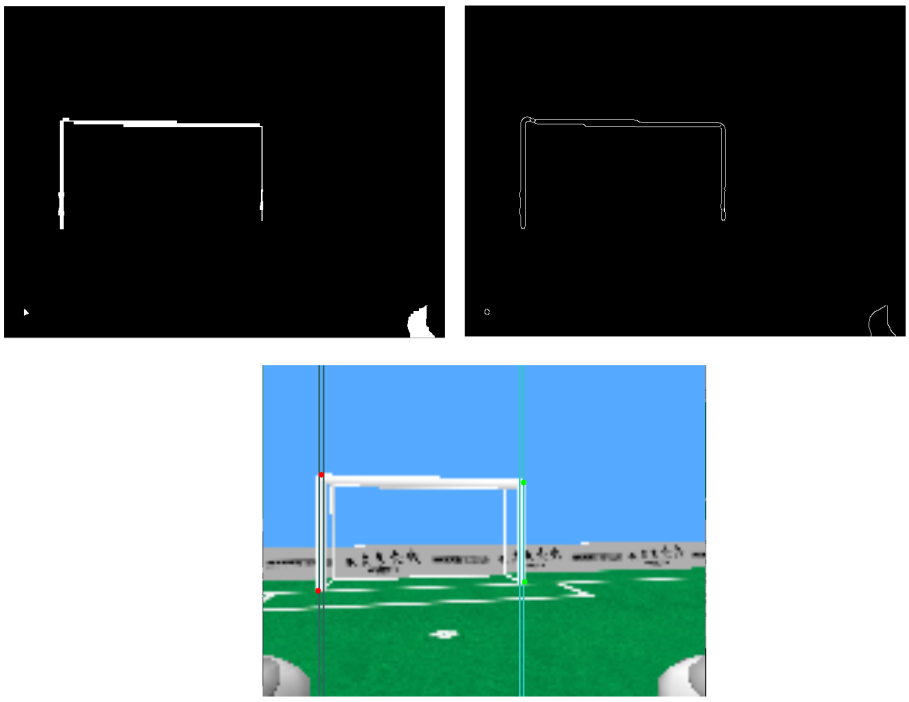

Figura 13: Etapas da detecção de gols.

representando a proporção entre o comprimento do travessão e o das traves. Se o valor resultante do filtro fosse maior que 0.5, o objeto era considerado um gol e a distância era calculada.

5.7 Detecção das linhas do Campo

De forma semelhante à segmentação realizada na procura dos gols, foi possível isolar de forma razoável as formas referentes às linhas de campo. Nessa etapa a baixa resolução de captura gerou um problema: as linhas do campo muitas vezes eram representadas por segmentos desconexos de pixels brancos intercalados por regiões verdes. Para tornar a detecção de reta mais eficiente foi aplicado o operador morfológico de dilatação com o seguinte elemento estruturante

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

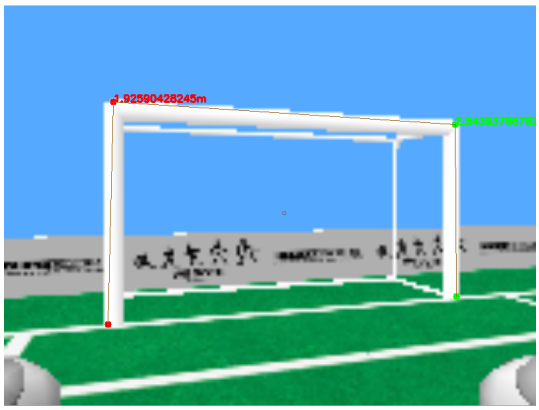

Figura 14: Gol detectado com a estimativa de distância.

seguido da aplicação do operador de erosão com o seguinte elemento estruturante:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

O resultado da aplicação dos operadores pode ser visto na figura 15. Com os segmentos reconectados, foi aplicada novamente a transformada de Hough para detecção de retas. Como a imagem resultante da aplicação do operador de Canny gera duas retas para cada faixa do campo (bordas inicial e final), algumas retas redundantes eram detectadas pelo algoritmo. Para resolver isso, consideramos uma diferença mínima do ângulo das retas para as adicionarmos ou não no conjunto de análise.

A estratégia utilizada para estimar a distância foi a baseada no ângulo de elevação do ponto em relação ao centro da imagem, já que apenas uma reta era atribuida a cada linha do campo e não foi estimada a largura da mesma para a utlzação do modelo de câmera *pinhole* baseada nessa medida.

5.8 Detecção da Bola

A imagem da bola capturada pela câmera apresenta uma característica que facilita a sua detecção: independentemente da sua posição e dos efeitos da perspectiva a sua imagem será representada por um círculo (em casos sem oclusão). Com a imagem segmentada, aplicamos a transformada de Hough para encontrar circunferências na imagem. Como a resolução da imagem é bastante reduzida, a forma circular da imagem da bola sofre algumas distorções e nem sempre é detectada com precisão pelo método. A ocorrência de falsos positivos também é elevada. Um exemplo de detecção correto pode ser visualizado na figura 16.

Para distinguir a bola real dos falsos positivos foi definido um intervalo de raio válido para o objeto e foi analisado o histograma de níveis de cinza da imagem original na região

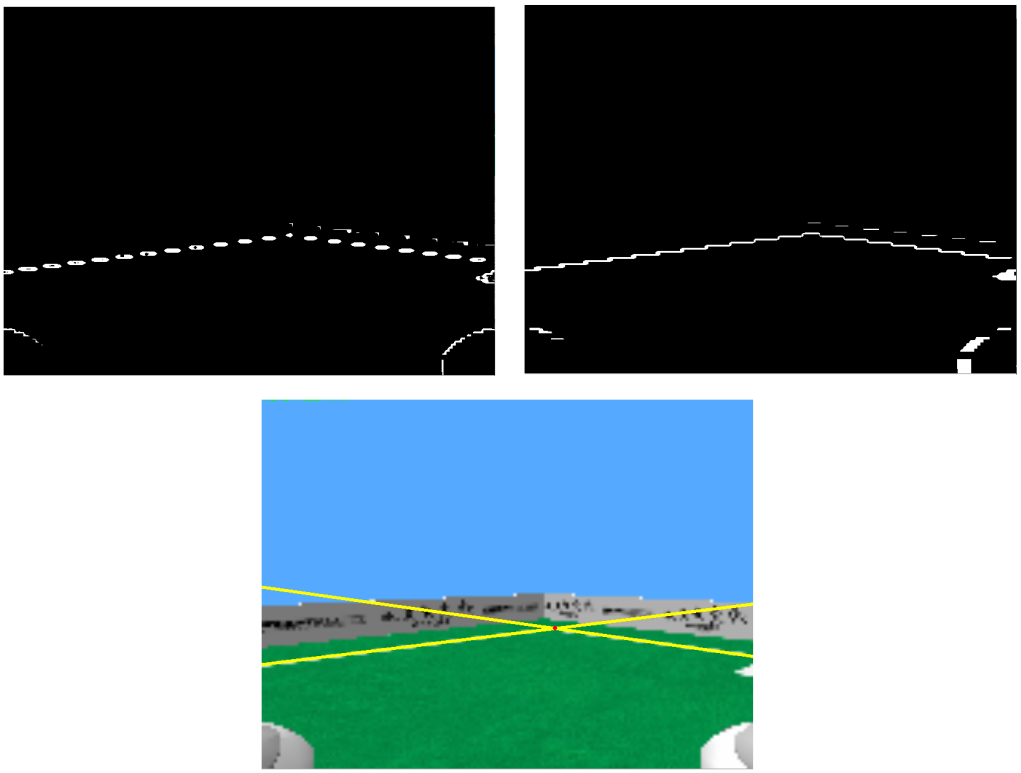

Figura 15: Aplicação de operadores morfológicos e detecção de canto.

Figura 16: Detecção da bola de futebol.

detectada. A presença de pixels pretos indica fortemente a presença da bola, que nos experimentos é quadriculada preta e branca. Devido a baixa resolução da imagem capturada, a forma que a representação da bola na imagem assumia nem sempre era perfeitamente circular, o que causava elevada variação do raio detectado pela transformada de Hough. As tentativas de estimar a distância da bola em relação a câmera baseadas primariamente no raio detectado não tiveram resultados satisfatórios. Dessa forma optou-se pela estratégia de levar em consideração o ângulo de elevação da bola em relação ao centro da imagem e a projeção do segmento de reta que intercepta o solo na direção do centro de massa da bola (dessa forma a variação do raio impactava de forma reduzida no resultado final).

5.9 Experimentos e validação

Os experimentos foram realizados com o uso do nó *Supervisor* da simulação. A ideia foi definir pontos específicos do campo nos quais o robô seria inserido e também um ângulo de rotação em relação às coordenadas globais. Para estes experimentos não foram considerados movimentos nos eixos do robô, todo o posicionamento foi realizado externamente pelo *Supervisor*, simplificando cálculos de ângulação e distância.

Para o caso do gol, foram feitos experimentos com dois ângulos diferentes em relação ao eixo X do campo: 0 graus e 45 graus. Para cada um deles foram definidas coordenadas que eram atualizadas após tempo suficiente para o servidor realizar o processamento e escrever em um arquivo de log as estimativas de distância e de ângulo em relação à câmera. Para a bola, definimos um ângulo arbitrário de 11 graus em relação ao eixo X do campo em uma posição que minimizasse as chances de linhas do campo interferirem na detecção do objeto. Também foram definidas as coordenadas de teste e computados os valores estimados pelo sistema de detecção. O canto do campo escolhido para o teste de detecção foi o escanteio, definindo o ângulo do robô em 45 graus e testando diversas combinações de coordenadas e registrando os valores gerados quando houvesse detecção.

Com os valores coletados, foi criada uma planilha no software *LibreOffice Calc* para computar a distância e a angulação esperadas para cada passo dos experimentos através das posições conhecidas dos objetos analisados e da leitura do sensor de posição do robô impressa juntamente aos dados gerados pela aplicação. Com os valores (ve) esperados e os obtidos (vo) calculamos o erro na medida pela simples fórmula

$$1 - \frac{ve}{vo}$$

6 Resultados

Nesta seção serão exibidas versões resumidas dos dados gerados e erros calculados após a execução dos experimentos. Cada tabela exibe os valores das coordenadas X e Y do robô no campo, a distância esperada calculada com base nessas coordenadas e as posições dos objetos analisados, a distância calculada pela aplicação e o erro calculado entre as duas. Os resultados de detecção de gol foram divididos em 4 tabelas, levando em consideração separadamente as traves esquerda e direita e as angulações utilizadas. As tabelas 5 e 6 mostram o resultado dos experimentos para detecção de bola e de escanteio, respectivamente.

Tabela 1: Resultado dos experimentos para trave esquerda com rotação de 0 graus

X (m)	**Y (m)**	**Dist. Esperada (m)**	**Dist. Calculada(m)**	**Erro**
0.75	-2	3.937	3.794	-0.03777
0.75	-1.5	3.815	3.644	-0.04686
0.75	-1	3.755	3.6	-0.04314
0.75	-0.5	3.762	3.6	-0.04499
0.75	0	3.834	3.656	-0.0487
0.75	0.5	3.969	3.777	-0.05081

Tabela 2: Resultado dos experimentos para trave esquerda com rotação de 45 graus

X (m)	**Y (m)**	**Dist. Esperada (m)**	**Dist. Calculada(m)**	**Erro**
1	-2	3.700	3.669	-0.00844
1.5	-2	3.231	3.209	-0.00688
2	-2	2.773	2.75	-0.00839
2.5	-2	2.332	2.273	-0.02612
3	-2	1.921	1.88	-0.02177

Tabela 3: Resultado dos experimentos para trave direita com rotação de 0 graus

X (m)	**Y (m)**	**Dist. Esperada (m)**	**Dist. Calculada(m)**	**Erro**
0.75	-2	4.6800	4.306	-0.08674
0.75	-1.5	4.3991	4.178	-0.05293
0.75	-1	4.1596	4.077	-0.02026
0.75	-0.5	3.9689	3.886	-0.02134
0.75	0	3.8344	3.8	-0.00904
0.75	0.5	3.7620	3.697	-0.01757

Tabela 4: Resultado dos experimentos para trave direita com rotação de 45 graus

X (m)	**Y (m)**	**Dist. Esperada (m)**	**Dist. Calculada(m)**	**Erro**
1	-2	4.4822	4.394	-0.02006
1.5	-2	4.1037	4.0822	-0.00525
2	-2	3.7537	3.735	-0.00499
2.5	-2	3.4409	3.4	-0.01203
3	-2	3.1765	3.086	-0.02931

Tabela 5: Resultado dos experimentos para a detecção de bola

X (m)	**Y (m)**	**Dist. Esperada (m)**	**Dist. Calculada(m)**	**Erro**
1	-2	2.45552	2.758	0.10967
1.5	-2	1.95591	2.12	0.07739
2	-2	1.45657	1.501	0.02959
2.25	-2.15	1.22702	1.193	-0.02852
1.46	-2.33	2.03705	2.237	0.08938

Tabela 6: Resultado dos experimentos para a detecção de canto do campo

X (m)	**Y (m)**	**Dist. Esperada (m)**	**Dist. Calculada(m)**	**Erro**
2	-0.84	3.3038	3.91	0.15501
2.3	-0.538	3.30173	4.17	0.20821
2.647	-1.592	2.32724	2.67	0.12837
1.275	-0.17	4.29063	5.04	0.14868
1.95	-0.88	3.31615	4.05	0.181

O código-fonte do trabalho e as planilhas mais detalhadas do experimentos podem ser encontradas no repositório: https://github.com/gabrielborgesmagalhaes/Computer-Vision-Nao-Robot

7 Conclusão e trabalhos futuros

Com este projeto foi possível analisar o estado atual das pesquisas sobre visão computacional no contexto da RoboCup e compreender os rumos da competição, assim como a sua importância no desenvolvimento e na melhoria de técnicas, algoritmos e sistemas de robótica e outras áreas. A revisão bibilográfica tornou claros os conceitos teóricos e as técnicas mais utilizadas no contexto de detcção de objetos, servindo de guia para as escolhas realizadas no projeto.

As limitações de processamento impostas pelo computador utilizado e pelo simulador forçou o uso de uma imagem de baixa resolução, que dificultou certos aspectos da detecção mas serviu como uma restrição realista, já que o hardware de muitos robôs não consegue realizar de forma rápida processamentos muito complexos. Os experimentos foram realizados em condições ideais, mais controladas do que um cenário realístico de uma partida de futebol de robô tanto quanto às condições de iluminação quanto ao movimento da câmera. No entanto, os resultados obtidos se mostraram bastante satisfatórios no sentido de um trabalho introdutório e preliminar na área de visão computacional em robótica, obtendo valores reduzidos na estimativa de distância e posicionamento dos objetos quando detectados. O ambiente configurado e desenvolvido pode ajudar trabalhos futuros na área.

O sistema ainda falha na detecção de objetos em circunstâncias de oclusão, movimentação rápida de câmera e é muito dependente das cores predefinidas, ficando bastante sensível em relação a mudanças na iluminação. A detecção de cantos não consegue distinguir por si só tipos diferentes de canto, como escanteio, área e meio de campo, mas pode ser utilizada como informação de entrada para um modelo probabilístico, que analisará o conteúdo restante da imagem e mesmo o histórico de localização estimada para realizar a distinção do que foi detectado.

Referências

[1] RoboCup Federation. http://www.robocup.org. (Visitado em 05/2017)

[2] Arjun B. Krishnan, S. Aswath, and Ganesha Udupa. 2014. *Real Time Vision Based Soccer Playing Humanoid Robotic Platform.* In Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing (ICONIAAC '14). ACM, New York, NY, USA, Article 52, 8 pages.

[3] A. Hartl, U. Visser, and T. Rofer. RoboCup 2013: Robot World Cup XVII, *Robust and Efficient Object Recognition for a Humanoid Soccer Robot*, pages 396–407. Springer Berlin Heidelberg, 2014

[4] Estivill-Castro, V., Radev, J.: Humanoids Learning who are teammates and who are opponents. In: The 8th Workshop on Humanoid Soccer Robots 13th at IEEE-RAS International Conference on Humanoid Robots, http://www.humanoidsoccer.org/ws13/program.html (2013)

[5] A Gudi, P de Kok, GK Methenitis, N Steenbergen. *Feature detection and localization for the RoboCup Soccer SPL* Project report, Universiteit van Amsterdam (February 2013)

[6] J. Mu and Y. Li, *A new efficient real-time arbitrary colored ball recognition method for a humanoid soccer robot*, 2016 12th World Congress on Intelligent Control and Automation (WCICA), Guilin, 2016, pp. 494-499.

[7] Schulz H., Liu W., Stückler J., Behnke S. (2011) Utilizing the Structure of Field Lines for Efficient Soccer Robot Localization. In: Ruiz-del-Solar J., Chown E., Plöger P.G. (eds) RoboCup 2010: Robot Soccer World Cup XIV. RoboCup 2010. Lecture Notes in Computer Science, vol 6556. Springer, Berlin, Heidelberg

[8] Cyberbotics - Webots Reference. https://www.cyberbotics.com/doc/reference/nodes-and-api-functions. (Visitado em 05/2017)

[9] Haldo Spontón, and Juan Cardelino, *A Review of Classic Edge Detectors*, Image Processing On Line, 5 (2015), pp. 90–123.

[10] PanoTools - Wiki - Field of View http://wiki.panotools.org/Field_of_View. (Visitado em 05/2017)

[11] Rectilinear Lens - Wikipedia https://en.wikipedia.org/wiki/Rectilinear_lens (Visitado em 06/2017)

[12] Georgakis G., Liu W., Stückler J., Behnke S. (2012) Field Landmark Recognition and Localization for the Robotstadium Online Soccer Competition.

[13] OpenCV library - http://opencv.org/ (Visitado em 06/2017)

[14] CMRoboBits Course - Veloso M., Rybski P. http://www.andrew.cmu.edu/course/15-491/ (Visitado em 06/2017)

[15] Pedrini H., Notas de Aula da disciplina de Introdução ao Processamento de Imagem Digital - http://www.ic.unicamp.br/ helio/disciplinas/MC920/MC920.html (visitado em 06/2017)