Centro Universitário da FEI

Projeto de pesquisa

Projeto de iniciação científica

Sistema de Visão Computacional para detecção de Robôs em Imagens da categoria Small Size de Futebol de Robôs

Orientador: Flavio Tonidandel
Departamento: Ciência da Computação
Candidato: Gabriel Parelli Francischini
N° FEI: 22.207.043-5

RESUMO

O projeto tem como objetivo desenvolver o novo sistema de visão para o time de Futebol de Robôs do Centro Universitário da FEI, categoria Small-Size, possibilitando a participação da equipe RoboFEI[1], em competições nacionais e internacionais, dando continuidade ao desenvolvimento da pesquisa em torno dessa categoria.

Este documento abordará as características dessa categoria, regras e objetivos, além das tecnologias envolvidas na parte de visão computacional pelas equipes que se destacam neste segmento.

1. - Introdução

O time de Futebol de Robôs iniciou visando promover a pesquisa e o desenvolvimento tecnológico envolvendo alunos de graduação e pós-graduação.

O projeto foi então intitulado de RoboFEI, onde foram desenvolvidas diversas pesquisas na área de hardware, software e projetos mecânicos. Em cinco anos foram concluídos dois trabalhos de projeto de formatura e seis iniciações científicas, sendo duas delas responsáveis pela participação da FEI na categoria Small-Size[2].

Este projeto de iniciação cientifica será desenvolvido de forma a atender as especificações da Robocup[3] na parte de visão computacional.

1.1 - Objetivo

Esta Iniciação Científica tem como objetivo desenvolver o novo sistema de visão para o time de Futebol de Robôs, categoria Small-Size[2], utilizando técnicas para que seja possível a identificação de cores e a detecção de pequenos objetos na imagem, buscando dar continuidade ao desenvolvimento de pesquisa da categoria e possibilitando a participação da equipe RoboFEI em campeonatos.

1.2 - Justificativa

Com o trabalho de Iniciação Cientifica de dois alunos, a nova equipe de Futebol de Robôs da FEI, categoria Small-Size, conta com o hardware e parte mecânica dos robôs pronta. Com isso, surgiu a necessidade de um sistema de visão adequado para suprir as dificuldades encontradas no ambiente do Futebol de Robôs. A categoria *Small-Size*, também conhecida como F-180, gerou dificuldades que não existiam na categoria Very-Small, exigindo um software melhor estruturado. Para se ter uma idéia da complexidade do sistema de visão, os robôs desta categoria incorporam tecnologias como o sistema de chute e de retenção da bola, além do sistema de movimentação omnidirecional, aumentando a

velocidade dos robôs e bola, sendo necessário o processamento de cada imagem em um tempo menor ou igual a 30ms, para que as imagens possam ser processadas em tempo real.

2. - Revisão Bibliográfica

Este tópico visa explicar um pouco sobre o Futebol de Robôs e seu funcionamento no Centro Universitário da FEI.

2.1 - O Futebol de Robôs

Nos últimos anos o desenvolvimento de Futebol de Robôs aumentou devido a sua extensa linha de pesquisa, possibilitando pesquisas interdisciplinares.

O principal foco de pesquisa é o desenvolvimento de robôs móveis autônomos, tendo como finalidade realizar tarefas sem intervenção humana.

A principal organização do Futebol de Robôs, a Robocup[3], foi criada em 1993, tendo em vista o desenvolvimento da Inteligência Artificial, robótica e assuntos relacionados. A Robocup promove diversas categorias de Futebol de Robôs, todas com um único objetivo, promover o desenvolvimento tecnológico dos robôs para que em 2050 um time de futebol de robôs possa entrar em campo contra a seleção humana campeã mundial do mesmo ano. Para isso a organização promove congressos que ocorrem junto às competições, fazendo com que os resultados das pesquisas sejam amplamente divulgados e aproveitados.

2.2 - O Futebol de Robôs na FEI

A equipe de Futebol de Robôs da FEI começou em 2003, coordenada pelo Prof. Reinaldo Bianchi, visando o desenvolvimento de pesquisas dedicadas à simulação. Após alguns meses nasceu o projeto RoboFEI, coordenado pelos professores Flavio Tonidandel e Reinaldo Bianchi, para que o Centro Universitário da FEI pudesse participar da competição de Futebol de Robôs *Mirosot* [4], subordinada a *Fira* e paralelamente ao 6^{O}SBAI (Simpósio Brasileiro de Automação Inteligente), que ocorreria em setembro desse mesmo ano.

Desde o inicio do projeto a FEI competiu seis vezes na categoria Very-Small alcançando excelentes resultados, a equipe foi campeã duas vezes, 2004 e 2006, vice-campeã três vezes, 2003, 2007 e 2008 além de um terceiro lugar em 2005.

A equipe RoboFEI Small-Size nasceu em 2006 aproveitando os bons resultados obtidos na categoria Very-Small. A equipe participou de apenas uma competição, em 2008, junto com o 19ºSBAI, na qual obteve o quarto lugar.

2.3 - Estrutura do *Small-Size*

Nesta seção serão descritas algumas das regras para a categoria Small-Size[2].

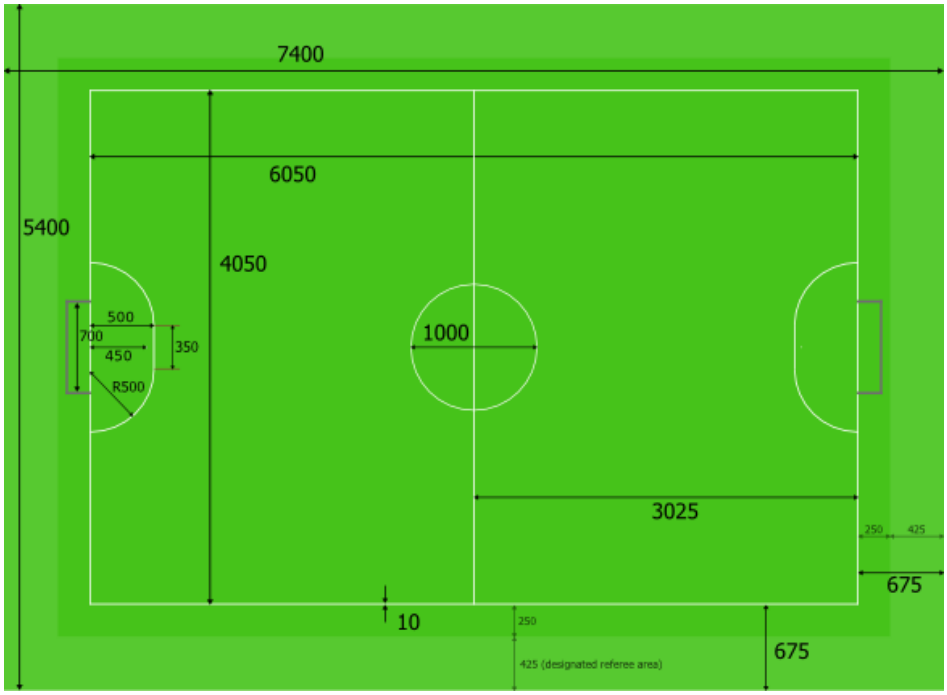


Figura 1 – Dimensões do campo

O campo de jogo dever ser devidamente nivelado, plano e recoberto com feltro verde, deve ter 7400 mm de comprimento por 5400 mm de largura, sendo que devem ser reservados à área de saída de robôs e bola 1350 mm de cada um dos lados do campo, como visto na Figura 1. A iluminação deve ser uniforme com 500LUX ou mais. O sistema de visão pode

ser local ou global, sendo que no segundo caso as câmeras devem ser dispostas a mais de 4m de altura e não podem interferir no jogo.

Assim como no Very-Small, a bola utilizada no jogo é uma bola de golfe laranja, pesando aproximadamente 46g e com um diâmetro de aproximadamente 43 mm. Cada equipe joga com cinco jogadores em campo, sendo um desses o goleiro.

Ainda segundo [2], os robôs não podem ultrapassar o tamanho máximo, um cilindro de 180 mm de diâmetro e 150 mm de altura caso o time use visão global, ou 225 mm de altura quando o time utiliza visão local.

Para a identificação dos times, cada um dos robôs deve conter no centro um círculo de 50 mm de diâmetro de cor azul ou amarela, para que os robôs sejam identificados e diferenciados da outra equipe. Os robôs também devem ser enumerados de maneira visível para que os juízes possam identificá-los facilmente.

2.4 – Visão Computacional

A área de processamento de imagens é um objeto de crescente interesse pois viabiliza grande número de aplicações para o aprimoramento das imagens digitais aplicando técnicas que utilizam operações matemáticas alterando os pixels e corrigindo os defeitos .

Segundo [6], com a imagem capturada e digitalizada, pode-se melhorar a imagem corrigindo defeitos provenientes da aquisição e realçando detalhes de interesse, de modo a facilitar sua segmentação.

A segmentação de imagem é descrito como o processo responsável por subdividir uma dada imagem em regiões homogêneas, que compartilhem determinada similaridade. Esse processo é de grande relevância em visão computacional e processamento e análise de imagens, possuindo grandes desafios ainda em abertos e largamente estudados. O motivo dessa grande importância pode ser justificado pelo fato da segmentação ser o primeiro passo para a análise e processamento de imagens [6].

Existem inúmeros métodos para segmentação das imagens como os descritos em [12]. A maioria desses métodos está restrita a um determinado domínio, não sendo capazes de realizar sua tarefa com eficiência em todos os domínios. Com isso vários trabalhos envolvendo métodos adaptativos e capazes de inferir decisão vêm sendo apresentados.

2.4.1. Detecção de bordas

Uma borda é o limite entre duas regiões com propriedades relativamente distintas de nível de cinza, segundo [6]. A técnica para detectar bordas consiste em obter a derivada primeira e a derivada segunda de um ponto da imagem a partir de um operador diferencial. A derivada primeira detecta a presença de borda e a derivada segunda determina em qual parte o pixel está.

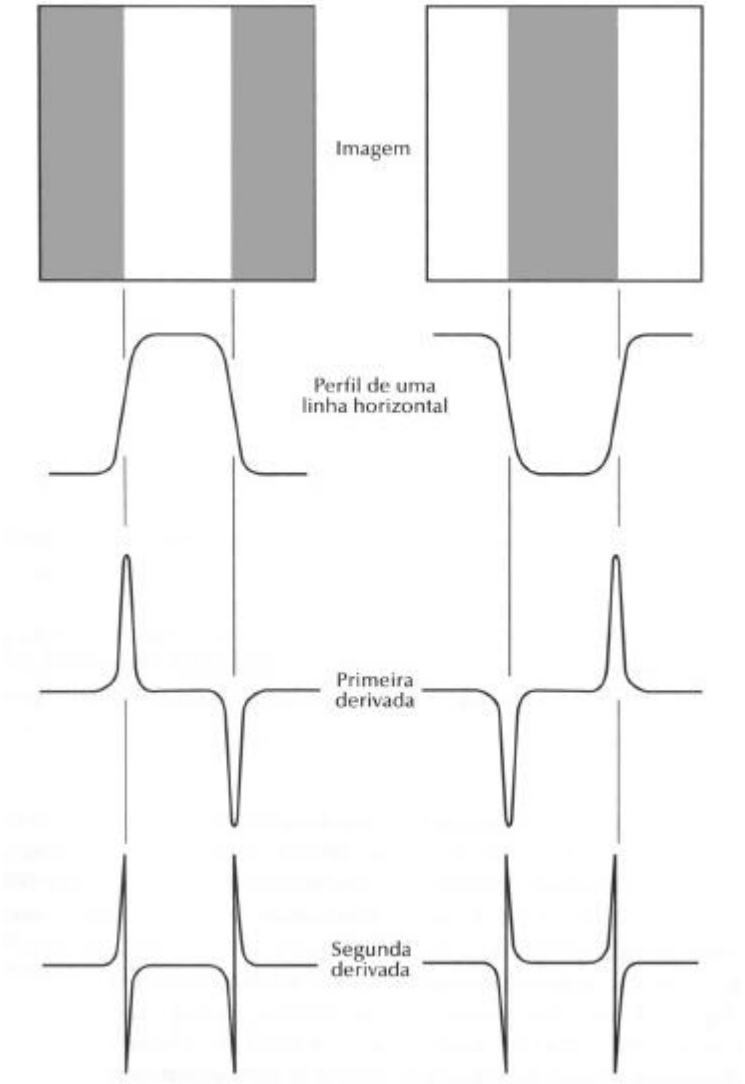


Figura 2 - Operadores de derivação para detecção de bordas

2.4.1.1 - Canny

O detector de bordas Canny[10] surgiu através dos trabalhos de Marr e Hildreth. Esse detector de bordas é um operador gaussiano de primeira derivada que suaviza os ruídos e localiza as bordas.

O detector de bordas especificado por Canny deve atender três critérios:

-Taxa de erro: o detector de bordas devera responder somente para bordas verdadeiras

-Localização: distância entre os pontos de borda encontrados pelo detector e a borda atual deverá ser a menor possível.

-Resposta: O detector de bordas não deverá identificar múltiplos pixels de bordas onde somente existe uma borda

A detecção de bordas é usada muitas vezes como uma saída a análise de toda a imagem, pois quando a borda de um objeto presente em campo é encontrada a análise ocorrerá apenas na área delimitada por essa borda, diminuindo o custo de processamento caso fosse preciso a análise de pixel a pixel.

2.4.1.2 – Sobel

O operador de Sobel[6] é utilizado no processamento de imagem em algoritmos de detecção de bordas. Sobel é um operador de diferenciação discreta que computa uma aproximação do gradiente da função de intensidade da imagem. Em cada ponto na imagem o operador calcula o vetor gradiente do ponto ou a norma deste vetor. O filtro produz apenas uma aproximação de gradiente, muitas vezes causando resultados inesperados, em especial para as variações de alta freqüência na imagem.

2.4.1.3. Transformadas de Hough

A transformada de Hough ajuda a resolver o problema de detecção de circunferências com raio fixo determinando quais pontos de uma imagem pertencem a esta circunferência de raio r. Assim temos um conjunto de coordenadas (x, y) que pretendemos encontrar os possíveis valores para os parâmetros (xc; yc), que correspondem aos pontos centrais da circunferência.

O espaço de Hough é construído como uma matriz (de dimensões iguais as da imagem original) em que as linhas e colunas representam possíveis valores de xc e yc.

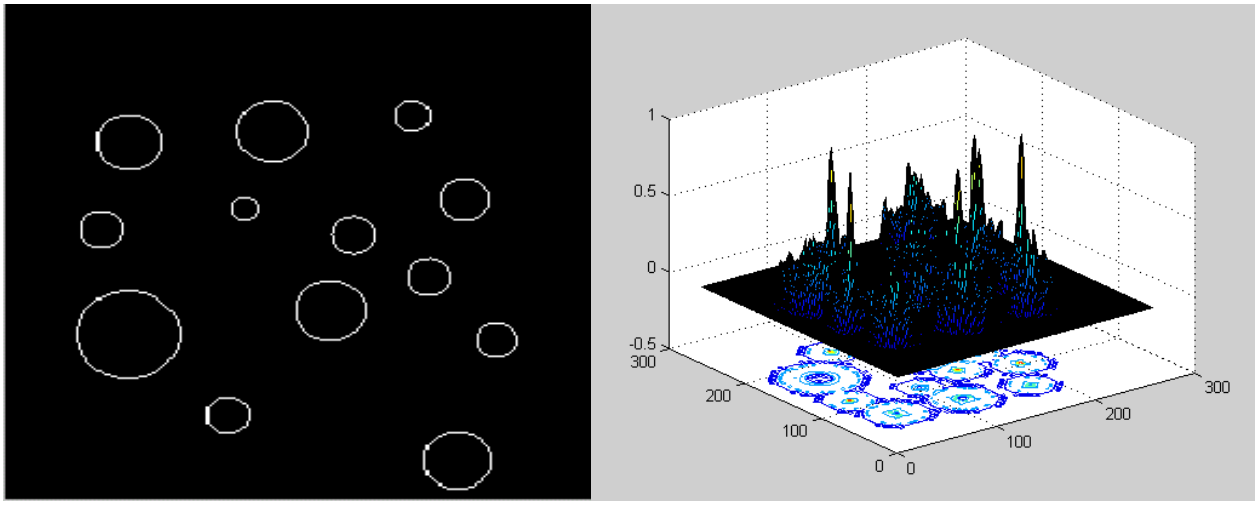


Figura 3 - (a)Bordas de um imagem qualquer. (b) Domínio da Transformada de Hough

A transformada de Hough combinada com um algoritmo de detecção de bordas resolverá o problema para encontrar círculos (círculos de 50mm de diâmetro estão dispostos no centro dos robôs), que após encontrados serão usado para diferenciar o time dos robôs, como no algoritmo de detecção de objetos por forma proposto na seção 3.4.3.

2.4.2 – Blob Colouring

Algoritmos de blob colouring[15] realizam a segmentação da imagem agrupando áreas de características similares, utilizando algoritmos de merging e growing.

Um simples algoritmo de blob colouring pode ser definido da seguinte forma:

T é o valor de threshold para a região

x é parte do objeto se f(x) > T

caso contrario x é parte do fundo

No exemplo acima a região é classificada em pertencente ou não ao fundo.

Está técnica pode ser facilmente empregada no domínio de futebol de robôs devido à diferença entre as possíveis cores adotadas para os robôs. Outro ponto favorável a implementação desta técnica é que após o processamento do algoritmo de segmentação pode-se aplicar algoritmos de operações morfológicas.

2.4.3 – Run Length Enconde

Run Length Enconde(RLE)[13] é um algoritmo de compressão sem perda de dados com boa taxa de compressão em casos específicos.

O algoritmo RLE trabalha de forma simples, substituindo sequências de caracteres de mesmo valor pelo número de vezes que aquele caractere apareceu e seu próprio valor.

O funcionamento do algoritmo pode ser visto na figura 4.

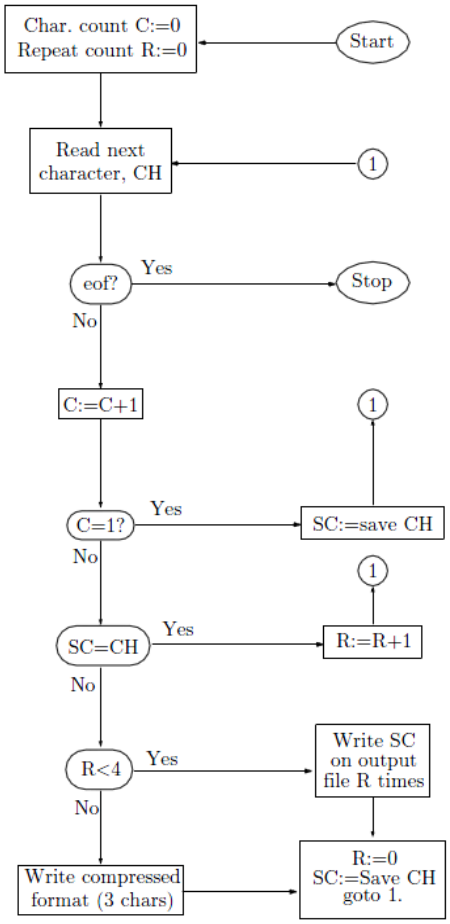


Figura 4 – Compressão RLE. Fonte: [13]

A compressão com o algoritmo RLE só é efetiva quando existe uma sequência de quatro ou mais caracteres repetidos, pois a cadeia de controle é formada por 3 caracteres, assim dados com sequências sem caracteres repetidos ficam maiores do que os dados sem compressão. Por outro lado o algoritmo não requer um alto uso de CPU e, segundo[13], é comumente usado para comprimir imagens (objetivo deste sistema de visão computacional).

Devido ao pequeno numero de cores, 6, podemos aplicar RLE satisfazendo a premissa de vários caracteres repetidos ao longo de um sequência.

2.4.4 – Floresta de Conjuntos-Disjuntos

Uma estrutura de dados de conjuntos-disjuntos é uma coleção S = {S1, . . . , Sk} de conjuntos dinâmicos disjuntos. Cada conjunto Sk é identificado por um representante, que é um membro do conjunto. Tipicamente não importa quem é o representante, apenas que ele seja consistente

Uma floresta de conjuntos-disjuntos [14] é um tipo de estrutura de dados representada conjuntos-disjuntos implementados em uma estrutura de árvore onde cada nó tem uma referencia pro seu pai.

A figura 5 demonstra uma simples implementação dos possíveis algoritmos para uma estrutura de dados conjuntos-disjuntos.

```
function MakeSet(x)
    x.parent := x

function Find(x)
    if x.parent == x
       return x
    else
       return Find(x.parent)

function Union(x, y)
    xRoot := Find(x)
    yRoot := Find(y)
    xRoot.parent := yRoot
```

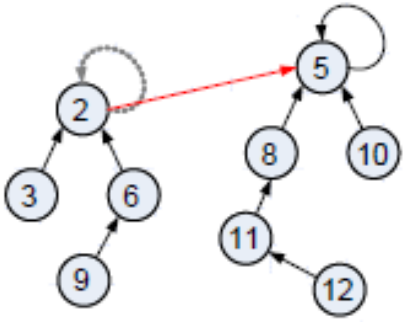
Figura 5 – Estrutura de dados de conjuntos-disjuntos.

2.4.5 - Union-Find

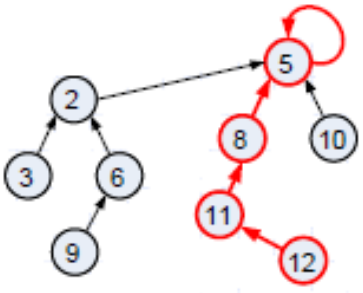
O algoritmo Union-Find[14] utiliza a estrutura de dados de conjuntos disjuntos para resolver o problema de unir dois elementos semelhantes (onde a igualdade obedece todas as propriedades de uma relação de equivalência). Na união o algoritmo une um set ao outro apontando a raiz de um set à raiz do outro set.

Esse algoritmo pode ser otimizado utilizando uma nova variável como parâmetro de classificação servindo para adicionar árvores menores em árvores maiores, minimizando o custo de processamento da operação de Find uma vez que a profundidade da árvore será a menor possível.

A figura 6(a). mostra a união dos elementos do set 2 com o elementos do set 5., para isso basta fazer raiz de um set apontar para a raiz do outro set. A figura 6(b) mostra como encontrar o set que um elemento pertence.



(a)



(b)

Figura 6 – Exemplo de união (a) e de busca do set (b).

3. - O Projeto

Para que seja possível realizar a identificação dos robôs e bola em todo a campo é necessário que o sistema disponha de duas câmeras localizadas a pelos menos 4 metros do campo para que a câmera consiga abranger o campo inteiro sem grande distorção causada pelas lentes.

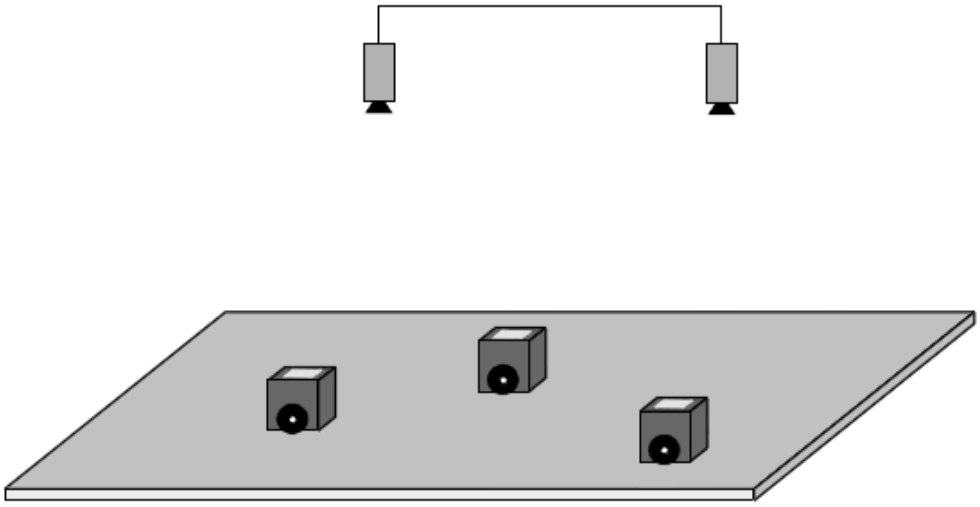


Figura 7 – Sistema de Visão Global

3.1 - Aquisições das imagens

A aquisição das imagens é feita com auxílio da biblioteca CMU 1394 Digital Camera Driver[5].

Uma classe foi implementada para tratar a captura e processamento das imagens, podendo assim ser facilmente adaptada a qualquer método de processamento escolhido.

Essa classe implementada é multithread sincronizada melhorando o desempenho e evitando que as imagens das câmeras não sejam capturadas em tempos semelhantes.

3.2 - Espaços de cores

O espaço de cores é um espaço multidimensional que consegue descrever a cor em um determinado ponto da imagem.

A primeira pesquisa foi realizada a fim de se obter o mais eficaz dos espaços de cores. Um bom espaço de cores é aquele que continua robusto mesmo quando existe uma grande variação de iluminação no ambiente, um dos grandes problemas no domínio do Futebol de Robôs.

Foram analisados três espaços de cores(HSV, RGB e YUV), ambos são espaços tridimensionais.

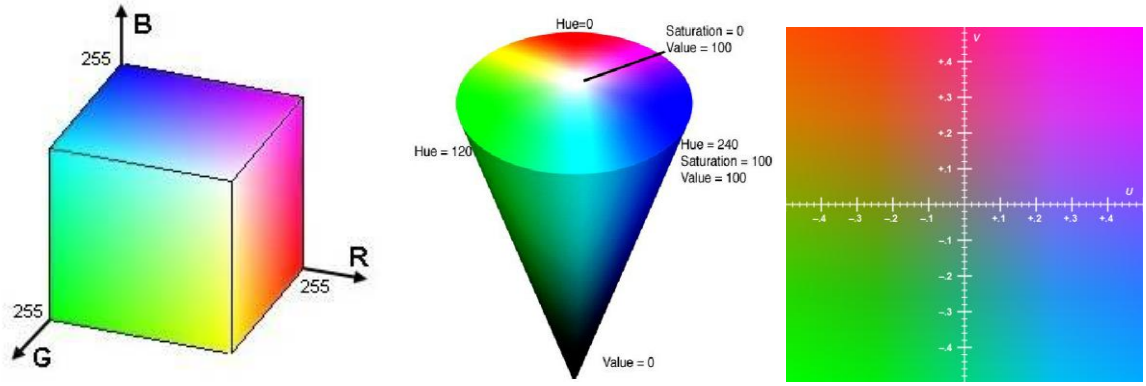


Figura 8 – Espaços de Cores(RGB, HSV, YUV – UV-Y(Y = 0.5))

O espaço de cores RGB se caracteriza por um cubo de lado R(Vermelho), comprimento G(Verde) e altura B(Azul). A figura 8 mostra a representação desse cubo.
Como é possível observar nessa imagem, cores parecidas têm diversos valores de RGB. Embora o modelo RGB seja bastante utilizado computacionalmente, a determinação ou classificação de cores nesse sistema por parte dos usuários não é uma tarefa trivial.

O espaço de cores HSV, H(HUE - Tonalidade), S(Saturation - Saturação), V (Value -Valor), descreve a relação percentual de cores mais precisamente do que em RGB, e é

mais próximo da percepção humana facilitando a classificação das cores por conta dos usuários. O grande problema para se usar o espaço de cores HSV é a transformação dos pixels que estão em RGB para HSV, pois demandam um grande custo computacional.

O espaço de cores YUV – Y(Intensidade), U e V(crominância), não apresenta descontinuidades na grade e tem uma representação que permite fácil obtenção da grandeza de iluminação do sistema, o eixo de intensidade. O sistema YUV tem como fundamento principal separar a iluminação (Y) e crominância(U e V). Tais eixos são conhecidos respectivamente como eixos “azul-vermelho” e “magenta-verde”.

3.3 – Decisão sobre o Espaço de cores

O sistema HSV foi escolhido por, segundo [6], ser de melhor percepção para o olho humano, aumentando o percentual de acerto para a calibração da rede neural. O alto custo computacional para transformar a imagem RGB para HSV foi resolvido utilizando uma Lookup Table(LUT), estrutura de dados usada para substituir os cálculos da rede neural em tempo de execução, em RGB, assim a rede neural é treinada em HSV e a LUT é gerada com valores RGB. A tabela RGB é criada com resolução 255x255x255, cada célula da tabela é convertida em HSV, com as fórmulas mostradas na figura 9, e repassada para a rede neural, a cor proveniente da rede neural é então anexada na célula da tabela RGB, assim, ao invés de transformar os pixels de RGB para HSV o sistema apenas consulta a LUT reduzindo o custo computacional.

Com o sistema escolhido a utilização de thresholds regionais se torna desnecessária uma vez que o sistema de cores HSV é tolerável a variação de iluminação.

$$H = \begin{cases} 60 \times \frac{G-B}{MAX-MIN} + 0, & \text{if } MAX = R \text{ and } G \geq B \\ 60 \times \frac{G-B}{MAX-MIN} + 360, & \text{if } MAX = R \text{ and } G < B \\ 60 \times \frac{B-R}{MAX-MIN} + 120, & \text{if } MAX = G \\ 60 \times \frac{R-G}{MAX-MIN} + 240, & \text{if } MAX = B \end{cases} \qquad S = \frac{MAX - MIN}{MAX} \qquad V = MAX$$

Figura 9 – Equações de transformação de coordenadas RGB para HSV

3.3 - Segmentação de Cor

Segundo [6] "os algoritmos para segmentação são usualmente baseados em duas propriedades: Descontinuidade e Similaridade". A segmentação baseada na descontinuidade contém alterações bruscas nos níveis da função imagem enquanto a segmentação baseada em similaridade agrupa regiões com características semelhantes

A representação das cores no HSV pode conter um numero indeterminado de regiões descontínuas que não conseguem ser discretizadas usando um threshold manual. Nesse ponto houve uma grande evolução no nosso sistema, uma vez que, o sistema antigo usava thresholds manuais para calibrar as cores, a calibração exigia tempo e nem sempre o resultado era o esperado.

A segmentação de cores do novo sistema é feita via uma rede neural [7] que aprende facilmente a classificar as cores a partir de um set de treinamento produzido pelo usuário. A rede neural construída contém três neurônios de entrada, vinte neurônios na camada oclusa, e sete neurônios para a saída.

O gráfico 1 ilustra o tempo em milissegundos para realizar a classificação da imagem com diferentes algoritmos.

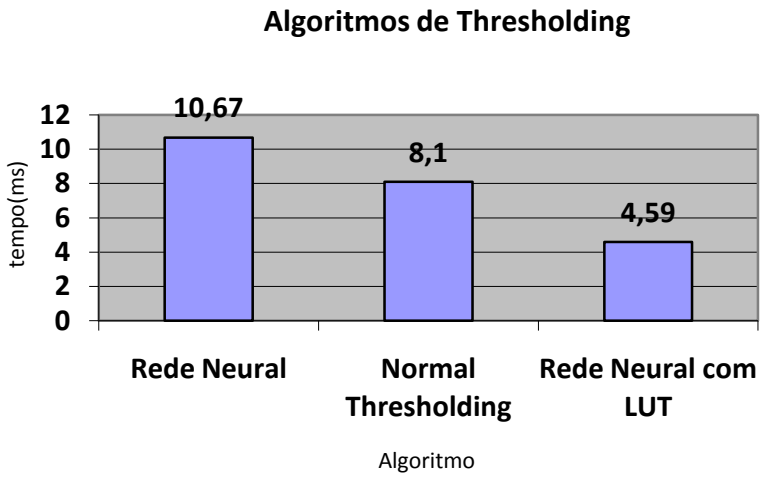


Gráfico 1 - tempo(ms) x algoritmo

3.4 - Análises das imagens

Foram analisados três diferentes algoritmos para identificação de objetos na imagem:

3.4.1- CMVision

O CMVision[8] é uma biblioteca de algoritmos para o processamento de imagens.

O algoritmo começa com uma matriz que contém a cor de todos os pixels levando em consideração os pixels de mesma cor que sejam eight-connectedness(qualquer pixel com a cor selecionada que esteja em volta do pixel analisado é um visinho). Essa é uma operação de alto custo computacional. Para diminuir esse custo computacional usa-se o algoritmo de RLE[13] que agrupa os pixels semelhantes que estão na mesma linha, como visto na figura 10. Após esse processo só será necessário olhar para a conexão vertical uma vez que o RLE já conectou as regiões horizontalmente. Para essa união usa-se o algoritmo Union Find.

LLLLLVVVALVVVAAA

5L3V1A1L2V3A

Figura 10 – Exemplo de algoritmo de RLE

Com as regiões agrupadas precisamos extrair algumas informações como o centróide, área total, bouding box (um contorno retangular que contém a região analisada). Outras informações podem ser extraídas, mas no domínio do Futebol de Robôs da categoria Small Size elas seriam desprezíveis. Com todas as informações necessárias extraídas, é criada uma lista ligada para cada cor e organizadas em ordem crescente de área total do blob, conjunto de pixels vizinhos de mesma cor, os menores tem preferência, pois normalmente os círculos têm 20 pixels de área, ficando mais fácil a identificação de cada blob por outros algoritmos.

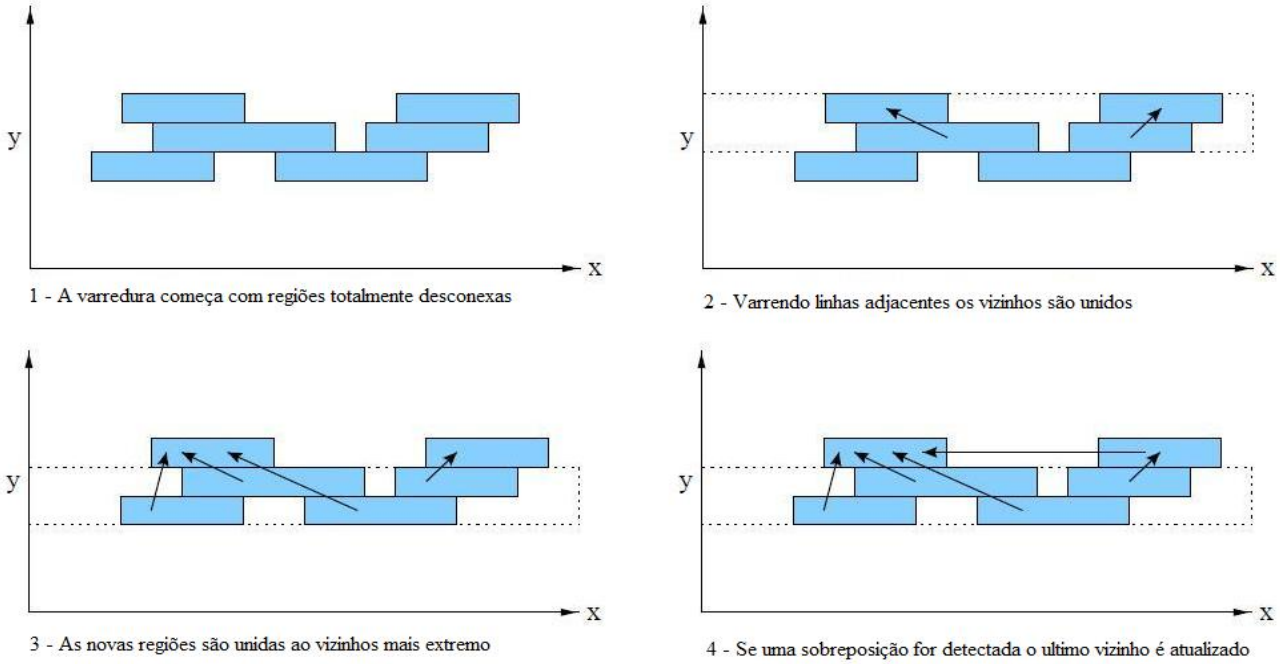


Figura 11 – Algoritmo Union Find adaptado de [8].

A figura 11 representa os passos que o algoritmo Union Find deve realizar após receber os dados comprimidos pelo RLE.

3.4.1.1 - Resultados Obtidos

Esse algoritmo[8] apresentou um bom desempenho rodando em um tempo <10ms tornando possível rodar o software a 60 frames por segundo mesmo após a inserção dos algoritmos de localização e previsão da localização dos robôs.

Função Tempo de execução 640x480	
Color Map(Via Rede Neural)	~5ms
RLE	~<1ms
Union-Find	~<1ms
Extract Regions	~<1ms
Separate Regions	~<1ms
Sort Regions	~1ms
Total	~<10ms

Tabela 1 - Dados obtidos no experimento

Os primeiros testes mostraram uma grande quantidade de ruído, diminuindo assim a confiabilidade deste algoritmo. Para amenizar os ruídos foi proposto como solução um Filtro Gaussiano de tamanho, onde o blob que melhor se aproxima do tamanho desejado tem uma maior confiança, aumentando a taxa percentual de acerto dos reais objetos e diminuindo o tempo de processamento (uma vez que possivelmente os blobs com maior confiança realmente serão os blobs pesquisados).

3.4.2 - Detectando cor independente da forma

Outro método testado foi proposto por [9] que utiliza a idéia de traçar segmentos internos aos objetos detectados de uma cor especificada.

O processo para encontrar os segmentos é basicamente percorrer a imagem na vertical ou horizontal a partir de um ponto A0, até encontrar um pixel que não seja da cor especificada, sendo A1 o último ponto com a cor especificada.

O grande problema do processamento em cruz é que ele precisa ser executado várias vezes para extrair as informações necessárias, pois o algoritmo gera vários centros provisórios que serão posteriormente serão analisados para encontrar o provável centro do objeto. O centro provável é a media espacial dos centros provisórios. Agrupar centros provisórios permite que imagens com ruídos possam ter o centro dos objetos estimados sem problemas como mostra a figura 12, onde os centros dos objetos estão indicados pelo ponto vermelho. Esse método apenas calcula o centro estimado do objeto e não seu centro real, levando em conta que ele calcula o centro de qualquer forma de objeto.

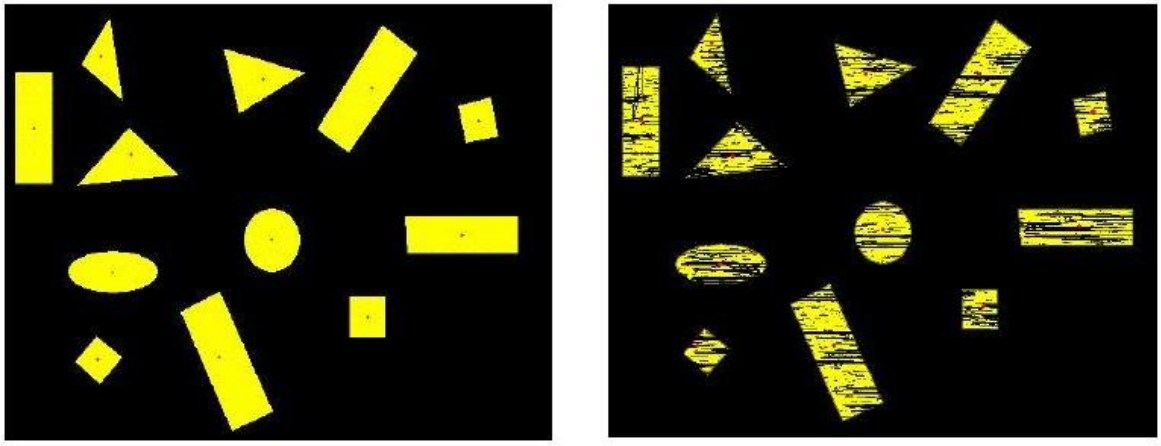


Figura 12 – Centros estimados com o algoritmo de detecção por cor[9].

3.4.2.1 - Resultados Obtidos

Os resultados obtidos com esse algoritmo[9] não se adéquam ao tempo necessário para um software de visão de qualidade e com alto desempenho. Obtivemos os seguintes tempos de execução para uma imagem de 640x480:

Função Tempo de execução 640x480	
Subtração de fundo	~4ms
Processamento em cruz	~17ms
Total	~21ms

Tabela 2 - Dados obtidos no experimento

O software conseguiu detectar todas as formas e centros aproximados com pouco erro, porém para isso foi necessário 21ms, tornando impossível rodar o software em 60 quadros por segundo.

3.4.3 - Detecção de objetos por forma

Mudanças bruscas no brilho da imagem são interessantes por várias razões como as descritas em [11]. Pontos da imagem em que o brilho muda de certa forma particular são chamados de borda. Detectando a borda do objeto podemos então detectar a forma do objeto em questão.

A mudança do brilho poderia estar associada tanto a borda do objeto como com as marcas nesse objeto, que no ambiente de Futebol de Robôs seriam os padrões colocados no robô.

A detecção de bordas então, nada mais é do que a diferença de contraste com os pixels vizinhos. Esse tipo de análise demanda um alto custo computacional uma vez que os pixels vizinhos ao pixel em análise precisam ser analisados.

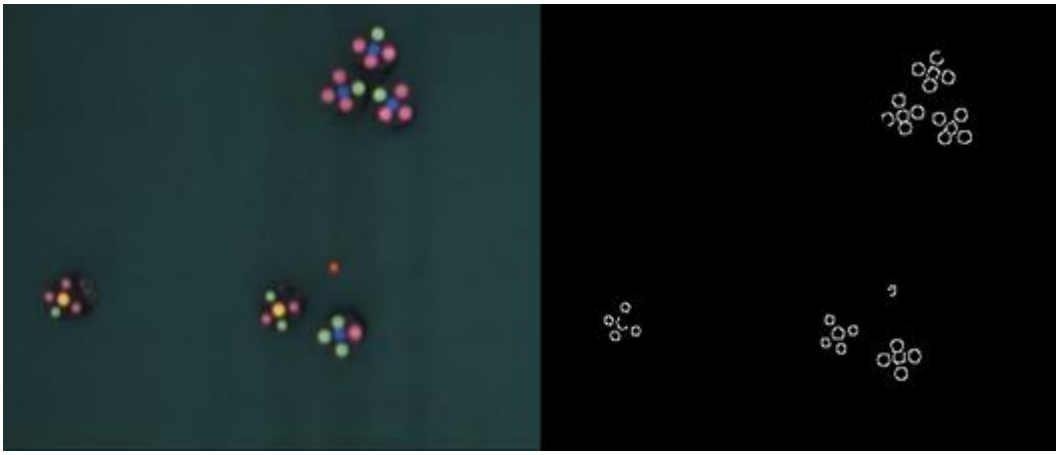


Figura 13 – a) Imagem capturada de uma câmera localizada a 3 metros de altura b) Filtro de Canny[10] (com thresolds 50 , 200).

Os teste foram realizados com o filtro de Canny[10] e o filtro Sobel[6], que usa a diferença local ponderada para computar os sinais de borda, gerando bons resultados como os da figura 13.

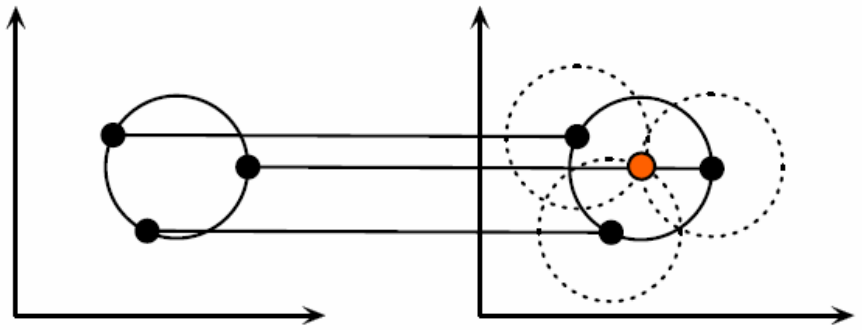


Figura 14 – Centro estimado com o algoritmo de detecção de bordas de um objeto conhecido[9].

Existem grandes limitações ao se utilizar apenas um filtro de bordas porque em alguns casos o objeto existente não tem o formato esperado, ou sua borda está corrompida. Outra limitação é a necessidade de parametrizar todos os tipos de objetos que precisamos extrair informações na imagem. A parametrizaçao de um objeto define a forma desse

objeto, deixando o sistema menos dependente de variação de cor uma vez que a cor do objeto parametrizado é dada pela soma das cores de todos os pixels dentro daquele objeto.

Círculos são uma estrutura geométrica muito comum no domínio do Futebol de Robôs e pela regra da categoria Small Size são obrigatórios na identificação dos times, deixando apenas o resto dos objetos de forma livre. Assim, para analisar a detecção de objetos por forma usamos método criado por [9] que usa detecção de borda com a transformada de Hough para círculos.

O sistema de visão proposto é bidimensional fazendo com que os círculos sejam parametrizados pela tupla (*xc*,*yc*) com raio constante.

O sistema utiliza a Transformada de Hough para determinar se um objeto é ou não um círculo, encontrando para cada ponto de borda um provável centro daquela borda. Os prováveis centros com maior número de votos são então considerados os verdadeiros centros dos círculos.

3.4.3.1 - Resultados Obtidos

Após alguns testes o algoritmo se mostrou ineficiente para detectar borda de objetos em movimento com grande velocidade, não se adequando ao perfil desejado para um software de visão computacional da categorial Small-Size onde a bola e os robôs se movimentam com grande velocidade e não dispõem de alto contraste devido à altura das câmeras. Obtivemos os seguintes tempos de execução para uma imagem de 640x480:

Função	**Tempo de execução 640x480**
Subtração de fundo e Filtro Sobel	~4ms
Conversão para Tons de Cinza e filtro Canny	~7ms
Espaço de Hough	~4ms
Localização dos centros dos círculos	~1ms
Total	~16ms

Tabela 3 - Dados obtidos no experimento

3.4.4 – Escolha do algoritmo para análise das imagens

Após alguns testes e pesquisas optou-se pela implementação do algoritmo CMVision, que já é amplamente usado no domínio de futebol de robôs. Esse algoritmo se provou robusto e eficiente. O grande problema dos outros algoritmos aqui citados é o tempo consumido, como pode ser observado no gráfico 2. Outro problema com o algoritmo de identificação de borda se da pela velocidade com que os objetos se movimentam em campo, uma bola pode chegar a até 10m/s fazendo com que esse objeto não tenha uma borda definida.

A figura 15.a mostra a geração dos blobs com a implementação do algoritmo [8].

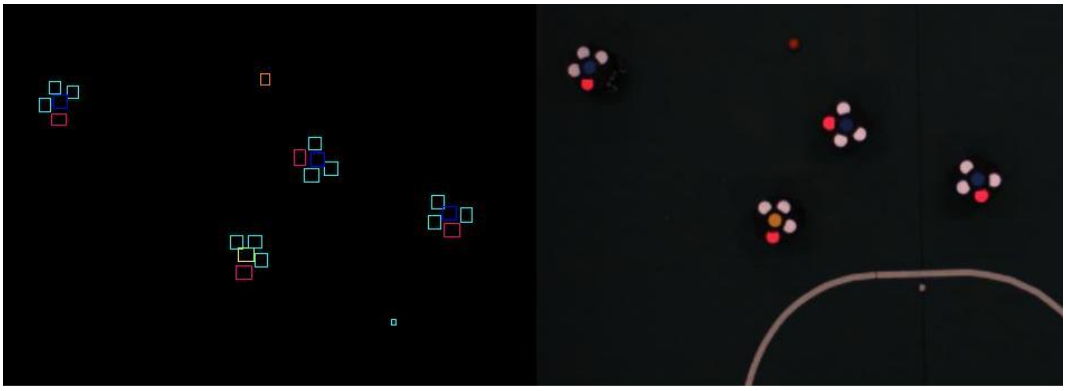


Figura 15 – a) Blobs encontrados com o algoritmo escolhido. b) Imagem capturada de uma câmera localizada a 3 metros de altura

A maior dificuldade com o algoritmo CMVision seria o custo computacional para preencher a matriz de cores, problema resolvido com a LUT.

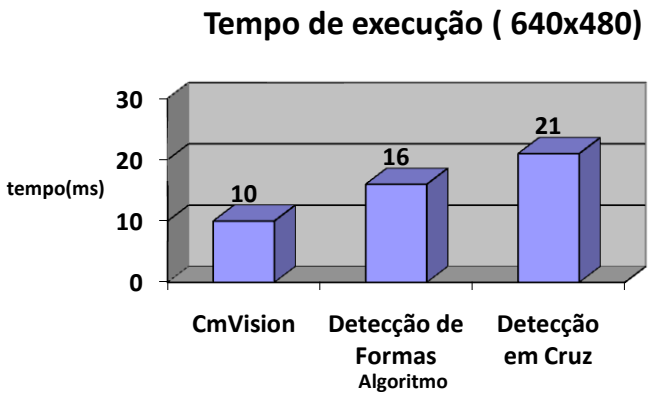


Gráfico 2- tempo(ms) x algoritmo

Pode-se observar também que o método criado por Bruce[8] é o mais preciso para detectar o centro dos objetos. Para realizar esse teste medimos as distâncias entre o objeto analisado e o centro da imagem em 10 frames, o objeto estava a 224mm da referência adotada, os seguintes resultados foram obtidos:

Dados obtidos			
	CMVision	Detecção por Forma	Processamento em Cruz
Media	223.9	223	221.2
Desvio Padrão	0.737864787	1.333333333	1.549193338
Erro Percentual	0,04464%	0,44843%	1,26582%

Tabela 4 - Dados obtidos no experimento

Assim, conforme a tabela 4, o método proposto em [8] mostrou o menor desvio padrão e erro de posição.

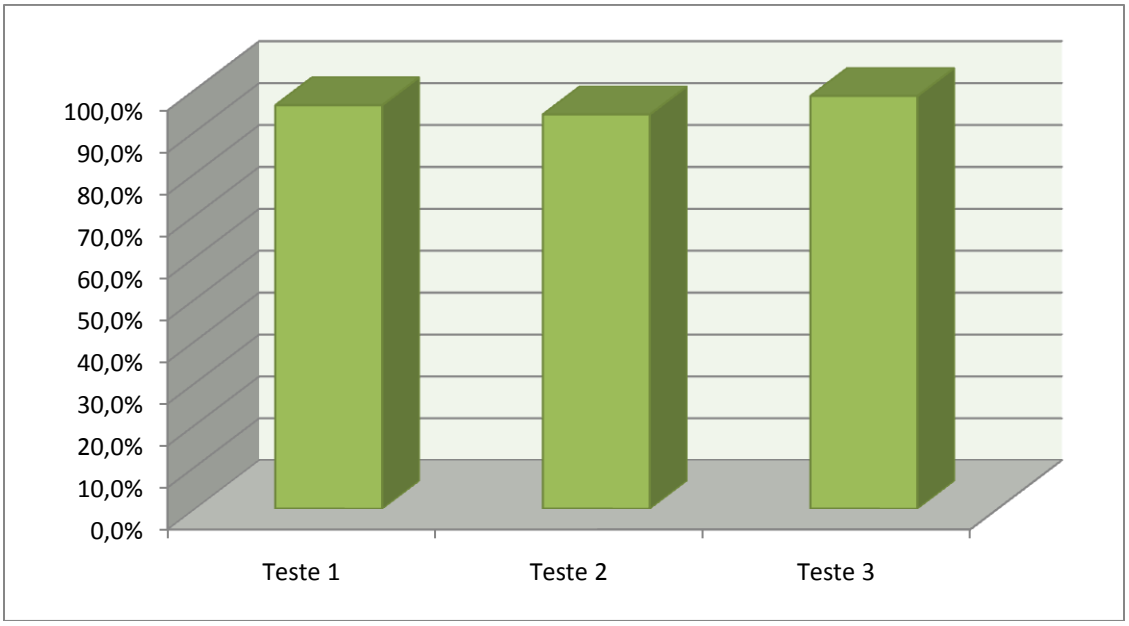


Gráfico 3- Porcentagem dos objetos encontrados.

4. Resultados

Para testar os resultados do sistema de visão utilizamos diferentes luminosidades e tempos para o obturador da câmera. Neste teste analisamos se os objetos presentes em campo foram detectados, como pode ser visto no gráfico 3. Apesar da mudança no tempo do obturador, a calibração da rede neural permaneceu a mesma para as diferentes luminosidades encontradas no campo.

O obturador da câmera aumenta ou diminui o tempo de exposição da captura de um frame da câmera. Quanto menor o tempo de exposição mais escura a imagem fica, pois o tempo de exposição à luz é menor.

Teste 1: Obturador: 400

Foi registrado um padrão de 96,3% de acerto, com media de consumo de tempo 16597.06us Essa configuração tem bom custo beneficio para ser usada em jogos uma vez que atende o requisito de tempo máximo para processar as imagens com uma boa porcentagem de acerto.

Teste 2: Obturador 300

Foi registrado um padrão de 94,1% de acerto, com media de consumo de tempo 16034.74us, está configuração atende as necessidades do sistema de visão, porém apresenta uma menor percentagem de acerto que a configuração usada no Teste 1 para uma pequena redução de tempo.

Teste 3: Obturador 900

Foi registrado um padrão de 98,5% de acerto, com média de consumo de tempo de 20392.30us.

Apesar do ótimo acerto registrado, está configuração se torna inválida devido ao tempo necessário para o obturador captar a imagem e entregá-la ao software.

Usando as configurações do primeiro teste foi registrado o consumo de tempo por 200 frames seguidos como o exibido no gráfico 4.

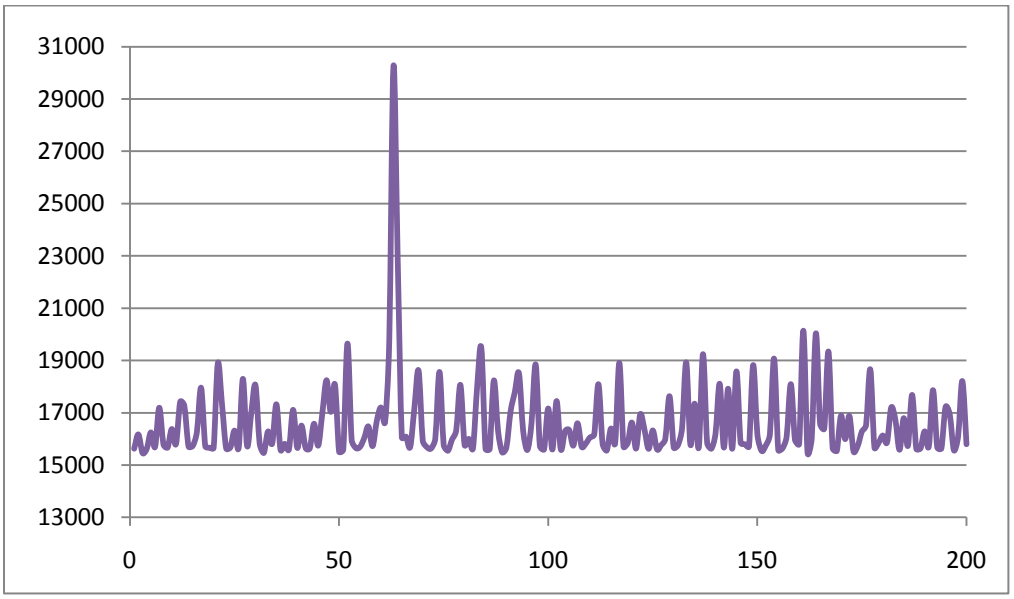


Gráfico 4- tempo(us) x frames.

O algoritmo implementado se mostrou estável, com média de 16597.06us e desvio padrão de 1544.551us. A baixa variação de tempos ajuda o software da estratégia que precisa tratar a variância de tempos para realizar uma melhor predição dos objetos.

A equipe de Futebol de Robôs do Centro Universitário da FEI participou de dois campeonatos utilizando os algoritmos aqui descritos conseguindo bons resultados:

2° colocado na Robocup Brasil Open;

12° colocado na Robocup.

5. Conclusão

Este relatório foi escrito com o objetivo de apresentar as etapas de desenvolvimento de um software confiável e de alto desempenho para aplicações com robôs, especialmente para a categorial Small-Size.

Por ter sido implementado com vários módulos, o sistema pode ser facilmente adaptado a outras categorias como já está sendo estudado o uso do software por equipes de outras universidades.

O software desenvolvido contempla todos os requisitos que um bom sistema de visão computacional necessita, como o baixo consumo de tempo e o alto grau de precisão na localização dos objetos.

A análise dos tempos demonstra que o software desenvolvido tem capacidade para rodar em tempo real e com uma baixa latência fatores cruciais para o desenvolvimento do software de estratégia dos robôs.

A maior parte dos algoritmos aqui descritos já possui um tempo de processamento mínimo, mas algumas funções ainda podem ser melhoradas, como a consulta a Lookup Table e a implementação de novas heurísticas para reduzir a quantidade de dados que precisam ser analisados. Outra possibilidade para aumentar a velocidade de processamento pode ser pela segmentação da imagem na GPU(Graphics Processing Units) uma vez que nos últimos anos elas têm aumentado rapidamente seu poder de processamento.

6. Bibliografia

[1] - *Equipe de competições robóticas da FEI* – RoboFEI . Disponível em: <http://www.fei.edu.br/robo/ >. Acesso em: 11 dez. 2008.

[2] - *Official RoboCup Small Size League*. Disponível em: <http://small-size.informatik.uni-bremen.de>. Acesso em: 11 dez. 2008.

[3] - H. Kitano, et al. **The Robot World Cup Initiative**. 1997.

[4] - *Robo Soccer Mirosot*. Disponivel em: <http://www.fira.net/soccer/mirosot/overview.html>. Acesso em: 11 dez. 2008.

[5] - *CMU 1394 Digital Camera Driver*. Disponivel em: <http://www.cs.cmu.edu/~iwan/1394/>. Acesso em: 26 jul. 2009.

[6] - Gonzalez, R. C; Woods, R. **Digital Image Processing**. Prentice-Hall. 2008.

[7] - Gurzoni Jr., J. A.; Martins, M. F.; Tonidandel, F.; Bianchi, R. **A neural approach to real time colour recognition in the robot soccer domain.** In Proceedings of SBAI'09, The IX Brazilian Symposium of Intelligent Automation. 2009 – "to appear".

[8] - Bruce, James. **Realtime Machine Vision Perception and Prediction.** 2000. Tese (Bacharelado em Ciência da Computação) – School of Computer Science, Universidade Carnegie Mellon, Pittsburgh.

[9] - Martins, M. F.; Tonidandel, F.; Bianchi, R. **Reconhecimento de Objetos em Tempo Real para Futebol de Robôs**. II Jornada de Robótica Inteligente. Anais do XXVI Congresso da Sociedade Brasileira de Computação. 2006.

[10] - Canny, J. **A computational approach to edge detection**. IEEE Transactions on Pattern Analysis and Machine Intelligence, v.8, n. 6, p.679-698. 1986.

[11] – Forsyth, D. A.; Ponce, J. **Computer Vision: A Modern Approach.** Prentice Hall. 2003.

[12] – Pal, N. R.; Pal, S. K. **A review on image segmentation techniques**. *Pattern Recognition*, 26, 1277–1294. 1993.

[13] – Salomon, David. **Data Compression: The Complete Reference.** Springer-Verlag. 2000.

[14] - Cormen, Thomas H; et al. **Introduction to Algorithms**. MIT Press & McGraw-Hill. 2001.

[15] - Andreas,K.; Abidi,M. **Digital Color Image Processing.** Wiley-Interscience. 2008.