

Trabalho de Conclusão de Curso

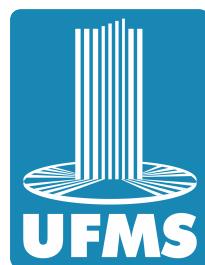
Sistema Automático de Calibração de  
Cores para a Equipe de Futebol De  
Robôs Cedro

Jasane Schio

Orientação: Prof. Dr. Gedson Faria

Coorientação: Prof. Me. Angelo Darcy Molin Bru

Área de Concentração: Sistemas de Informação, Visão Computacional



Sistema de Informação  
Universidade Federal de Mato Grosso do Sul  
21 de Setembro de 2016

# Sistema Automático de Calibração de Cores para a Equipe de Futebol De Robôs Cedro

Coxim, 21 de Setembro de 2016.

Banca Examinadora:

- Prof. Me. Angelo Darcy Molin Brun (CPCX/UFMS)
- Prof. Dr. Gedson Faria (CPCX/UFMS) - Orientador
- Prof. Gustavo Yoshio Maruyama (IFMS)
- Prof. José Alves de Sousa Neto(CPCX/UFMS)

# Resumo

Este trabalho apresenta um sistema automático de calibração de cores. Além da automatização o sistema reduz o tempo gasto pelas equipes de futebol de robôs nesta tarefa. Utilizou-se a técnica Algoritmo de Canny, que faz a detecção das bordas presentes na imagem e em seguida utiliza-se do algoritmo *border following*. Cada um dos objetos encontrados foi analisado de acordo com o estudo das cores para assimilar a qual cor a mesma pertencia, e seus valores comparados para identificar se esta seria um limite do intervalo da cor a qual pertence.

Palavras-chave: Calibração de Cores, Futebol de Robôs, OpenCV, Visão Computacional, HSV, Equipe Cedro;

# Abstract

# Conteúdo

<b>Lista de Figuras</b>	<b>7</b>
<b>1 Introdução</b>	<b>9</b>
1.1 Considerações Iniciais . . . . .	9
1.2 Trabalhos Correlatos . . . . .	10
1.2.1 Calibra . . . . .	10
1.2.2 VSS-Vision . . . . .	10
1.3 Motivação e Justificativa . . . . .	11
1.4 Objetivos . . . . .	11
1.5 Organização da Trabalho . . . . .	11
<b>2 Fundamentação Teórica</b>	<b>13</b>
2.1 Considerações Iniciais . . . . .	13
2.2 Processamento Digital de Imagens . . . . .	13
2.2.1 Detecção de Objetos . . . . .	14
2.2.2 Detecção de Bordas . . . . .	14
2.3 Universo de Cores . . . . .	15
2.4 Futebol de Robôs . . . . .	18
2.5 Considerações Finais . . . . .	19
<b>3 Desenvolvimento</b>	<b>20</b>
3.1 Considerações Iniciais . . . . .	20
3.2 Tecnologias Usadas . . . . .	20
3.3 Descrição do Projeto . . . . .	21
3.3.1 Organização do Projeto . . . . .	21
3.3.2 Apresentação das Classes . . . . .	21

3.4	Detalhes de Implementação do Sistema SCIMM . . . . .	23
3.4.1	1 <sup>a</sup> Etapa - Configuração de Câmera . . . . .	23
3.4.2	2 <sup>a</sup> Etapa - Reconhecimento de Fundo . . . . .	25
3.4.3	3 <sup>a</sup> Etapa - Extração dos Objetos do Fundo . . . . .	25
3.4.4	4 <sup>a</sup> Etapa - Detecção e validação de Objetos . . . . .	26
3.4.5	5 <sup>a</sup> Etapa - Classificação do Pixel . . . . .	27
3.4.6	6 <sup>a</sup> Etapa - Gerar Arquivo de Cores . . . . .	28
3.5	Considerações Finais . . . . .	28
<b>4</b>	<b>Testes e Resultados</b>	<b>29</b>
4.1	Considerações Iniciais . . . . .	29
4.2	Calibração . . . . .	29
4.3	Testes . . . . .	31
4.3.1	Cores Comuns . . . . .	33
4.3.2	Cores Com Problemas . . . . .	38
4.3.3	Análise Geral . . . . .	40
4.4	Considerações Finais . . . . .	40
<b>5</b>	<b>Conclusão</b>	<b>41</b>
5.1	Principais Considerações . . . . .	41
5.2	Trabalhos Futuros . . . . .	42
	<b>Referências Bibliográficas</b>	<b>43</b>

# **Lista de Figuras**

2.1	Imagen original (CANNY, 1986) . . . . .	15
2.2	Bordas encontradas(CANNY, 1986) . . . . .	15
2.3	Exemplo dos sistemas de cores RGB E CMY. . . . .	16
2.4	Diagrama de cromaticidade estabelecido pelo CIE. . . . .	16
2.5	Exemplo do Modelo de Cor RGB. Horvath (2008) . . . . .	17
2.6	Exemplo do Modelo de Cor HSV, Horvath (2008) . . . . .	18
2.7	Estrutura de um ambiente de futebol de robôs VSSS(FARIA, 2006) . . . . .	19
3.1	Organização das pastas do projeto . . . . .	21
3.2	Diagrama de Fluxo . . . . .	23
3.3	Configuração de Camera . . . . .	24
3.4	Configuração de Camera . . . . .	25
3.5	Geração de Máscara . . . . .	26
4.1	Imagen do fundo com a seleção de campo . . . . .	30
4.2	Objetos dispostos no campo para calibração . . . . .	30
4.3	Divisão do campo em quinze partes nomeadas alfabeticamente. . . . .	31
4.4	Imagen do campo com marcadores de cores . . . . .	31
4.5	Disposição de cada parte quanto as cores . . . . .	32
4.6	Divisão do campo. . . . .	33
4.7	Objetos da cor amarela . . . . .	33
4.8	Divisão do campo. . . . .	34
4.9	Os objetos da cor azul . . . . .	34
4.10	Divisão do campo. . . . .	35
4.11	Objetos da cor verde . . . . .	35

4.12 Divisão do campo. . . . .	36
4.13 Objetos da cor rosa . . . . .	36
4.14 Divisão do campo. . . . .	37
4.15 Objetos da cor roxo . . . . .	37
4.16 Divisão do campo. . . . .	38
4.17 Imagem somente com os objetos dentro do intervalo do valor da cor vermelho	38
4.18 Divisão do campo. . . . .	39
4.19 Objetos da cor laranja . . . . .	39
4.20 Gráfico de análise do resultado dos testes . . . . .	40

# Capítulo 1

## Introdução

### 1.1 Considerações Iniciais

Em 1997 foi estabelecida a Federation of International Robot-soccer Association (FIRA), com o intuito de promover o desenvolvimento nas áreas de multi-agentes autonomos e co-operação entre robôs, bem como pesquisas e estudos relacionados a mecatrônica, processamento de imagens e robótica(FIRA, 2016a; FIRA, 2016b). O Futebol de Robôs tem sido uma das principais áreas de foco por ser um domínio complexo, exigindo autonomia do sistema e solução de problemas em tempo real(COSTA; PEGORARO, 2000; FARIA, 2006).

Nos jogos de futebol, o técnico passa as informações pessoalmente para seus jogadores em campo, porem isso não é possível no futebol de robôs, já que neste ambiente tais informações precisam ser passadas via comunicação de dados.Os robôs são diferenciados computacionalmente por um identificador único (e.g. endereço MAC) e em sua carcaça por marcadores com duas cores, uma cor designando o time a qual o robô pertence e outra que o identifica de maneira única dentro de sua equipe. O software de estratégia de cada equipe deve detectar os marcadores de cor utilizando técnicas de visão computacional e decidir a partir da posição dos robôs em campo qual atitude será realizada.

Para identificar cada uma das cores devem considerados alguns fatores como: a iluminação local, a tonalidade de cor dos marcadores, a posição dos refletores e sombras sobre o campo, portanto, é necessário que se faça o processo chamado de calibração. A calibração de cores ocorre para designar o intervalo de valores que corresponde a cada cor naquele determinado momento. Intervalos de cores podem ser pré definidos, no entanto, de acordo a mudança de qualquer um dos fatores já citados, uma nova calibração se faz necessária.

Neste trabalho, foi estudado o problema da calibração de cores para competição de futebol de robôs categoria *IEEE Very Small Size Soccer* (VSSS), na qual a calibração de cada uma das cores em campo acaba se tornando um processo exaustivo por ter de ser feito um a um, cerca de 5 minutos para cada cor. Sendo assim, automatizar o processo de calibração, auxiliaria em um problema comum a todas as equipes. Na próxima seção, serão apresentados dois trabalhos exemplificando a calibração de forma manual.

## 1.2 Trabalhos Correlatos

Nas pesquisas sobre calibração de intervalo de cores para times de futebol de robôs, não foram encontrados trabalhos que descrevessem com detalhes a implementação do sistema de calibração, sendo os mais completos o *Team Discription Papers*<sup>1</sup> dos times.

### 1.2.1 Calibra

O Centro Universitário da Fundação Educacional Inaciana(FEI) propôs o sistema denominado CALIBRA, desenvolvido para sistemas Linux e com *Graphical User Interface*. O sistema de calibração possui um módulo chamado de *Main Window* que disponibiliza a configuração de brilho, cor e contraste da imagem adquirida pela câmera, e grava estas configurações em um arquivo que é utilizado no momento de definir os intervalos no espaço de cores HSI para cada uma das cores dos marcadores de identificação dos robôs(PENHARBEL et al., 2004a; PENHARBEL et al., 2004b). Deve-se ressaltar que a descrição do sistema CALIBRA não contempla os detalhes de implementação da calibração das cores.

### 1.2.2 VSS-Vision

O VSS-Vision é um sistema completo de controle da equipe de futebol de robôs da categoria VSSS do Laboratório de Sistemas Inteligentes e Robótica (SIRLab) da Faeterj/Petrópolis (ROSA, 2015).

Rosa (2015) menciona que a calibração de cores é feita calibrando obrigatoriamente laranja, amarelo e azul, e então as outras cores referentes aos jogadores em campo. A imagem da câmera é dividida em nove partes, e para calibrar a cor o usuário deve clicar sobre a cor que gostaria de ser calibrada, assim salvando um intervalo de cor tratado como RGB máximo e o mínimo daquela cor, a medida que vão havendo os cliques, o sistema verifica para cada atributo (R,G e B) se ele é maior que o atributo máximo salvo ou menor que mínimo salvo, caso seja, o mesmo assume o lugar de menor ou maior, e esse processo deve ser feito em cada uma das nove partes da imagem. Para desenvolvimento do sistema de processamento de imagens foi utilizada a biblioteca OpenCV e para telas interativas a biblioteca ImGui.

O atual sistema de visão computacional do SIRLab passou por algumas mudanças desde 2015 e conta com uma interface e método de calibração diferentes do inicial, utilizando-se da calibração no espaço de cores HSV, no lugar do RGB. A antiga interface do sistema, feita inicialmente em ImGui deu lugar a nova, desenvolvida em Qt(??).

O método de calibração de cores também foi modificado possibilitando a calibração de 8 cores: laranja, amarelo, azul, vermelho, verde, rosa, roxo e marrom(??). O processo de calibração das cores ocorre de forma manual, primeiramente o usuário escolhe uma cor e clica em algum objeto desta cor, o sistema então reconhece os valores mínimos e máximos de HSV. No segundo passo, o usuário realiza um ajuste fino em cada um dos atributos (H, S e V) até encontrar os valores adequados que representem não só objeto selecionado, mas todos os objetos da cor escolhida.

---

<sup>1</sup>Documento obrigatório para inscrição nas competições brasileiras que descreve o time de robôs, contendo informações sobre o chassi, eletrônica, visão computacional e estratégia.

## 1.3 Motivação e Justificativa

Em 2015 surgiu a equipe Cedro, equipe de futebol de robôs categoria VSSS da UFMS Câmpus Coxim. Neste mesmo ano, a equipe participou da Competição Latino Americana de Robótica (LARC), na qual foram realizadas algumas observações sobre as demais equipes e as partidas, mais especificamente, sobre a calibração de cores, motivando os estudos e pesquisas presentes neste trabalho. Embora não conste nas regras oficiais, todo jogo possui um tempo de 20 minutos para aquecimento, no qual a maioria das equipes realiza a calibração de cores. Nas equipes que observadas, o processo de calibração foi feito de forma manual com uma interface muito semelhante ao desenvolvido por Kyle Hounslow(HOUNSLOW, 2013), sendo que, a equipe POTI/UFRN desenvolveu em 2007 um novo modelo de cores, o HPG, o qual utilizaram em todas as competições desde então (MARTINS et al., 2007; MENDES; MEDEIROS, 2008).

Automatizar o processo de calibração das cores, reduziria o tempo utilizado, e assim, haveria mais tempo para ser usado em melhorias técnicas, hardware dos robôs com problemas, melhorias na estratégia de jogo ou resolvendo problema de comunicação. Assim, o benefício trazido para as equipes justifica o desenvolvimento um sistema autônomo de registro do valores HSV, que faça a definição de intervalos de cores baseando-se nos objetos em campo, os identificando de forma automática, e assim reduzindo o tempo gasto na calibração de cores.

## 1.4 Objetivos

Neste trabalho tem-se por objetivo principal automatizar o sistema de calibração de cores no espaço HSV em imagens capturadas em tempo real, identificando os limites mínimos e máximos dos atributos H, S e V para as cores de identificação dos robôs. Para desenvolver um sistema que supra essas necessidades, foram propostos os seguintes objetivos específicos:

- Estudar detecção de bordas e implementar a detecção automática dos objetos em campo;
- Estudo do espaço de cores RGB, HSV e as diferenças para BGR e HSV presentes na biblioteca *Open Source Computer Vision Library*(OpenCV);
- Analisar as cores dos objetos detectados e classificá-los no espaço de cores HSV do OpenCV;
- Testar a eficiência do sistema de calibração automático no reconhecimento de objetos estáticos em diversas localizações do campo.

## 1.5 Organização da Trabalho

Este trabalho está dividido em cinco capítulos. No segundo capítulo encontra-se a fundamentação teórica, conteúdo informações sobre processamento de imagens referente à detecção de objetos e cores e uma breve descrição sobre o futebol de robôs. No terceiro

capítulo encontra-se todo o desenvolvimento do projeto, suas classes, a descrição das tecnologias utilizadas no projeto, e a descrição do processo de calibração. No quarto capítulo são apresentados os testes e discutidos os resultados obtidos. No quinto capítulo são feitas as conclusões do trabalho, bem como expostas melhorias que possam vir a ser implementadas em trabalhos futuros.

# Capítulo 2

## Fundamentação Teórica

### 2.1 Considerações Iniciais

Neste capítulo serão abordados os temas relacionados as imagens digitais, o universo de cores e o futebol de robôs. Incia-se com as teorias sobre o processamento digital de imagens, detecção de objetos e detecção de bordas. Apresenta-se também os estudos sobre as Cores, explicando seus espaços, sistemas e modelos de cores. Por fim, descreve-se o ambiente do futebol de robôs e as regras para categoria VSSS.

### 2.2 Processamento Digital de Imagens

O processamento digital de imagens é uma área de estudo que consiste na manipulação da imagem e seus elementos, transformando-a visando obter mais facilmente as informações nela contidas(ALBUQUERQUE; ALBUQUERQUE, 2001). Além da retirada de informações, o processamento de imagens também inclui a modificação ou melhoria da imagem por meio de redução de ruídos, melhoria no contraste, segmentações da imagem entre outras técnicas. Gonzalez e Woods (2008) escrevem que embora não existam barreiras definidas, um imagem pode ser processada de três maneiras:

**Processo de baixo nível** aplica ações primitivas de modificação de imagem. Este processo tem como característica seu resultado final ser também uma imagem, semelhante a imagem inicial contendo melhorias.

**Processo de nível intermediário** consiste na divisão de uma imagem em regiões ou objetos, afim de processá-los individualmente para obter suas informações. Se caracteriza por seu resultado final ser muitas vezes apenas regiões, objetos ou outras informações extraídas da imagem.

**Processo de alto nível** se define por ser um processamento mais sensorial. Trata-se da análise de objetos usando funções cognitivas associadas a visão computacional, essa usa informações relevantes para o reconhecimento de objetos.

### 2.2.1 Detecção de Objetos

A detecção de objetos é uma das áreas de visão computacional que obtêm muita atenção dos pesquisadores, contudo, pode ser considerada uma técnica herdada do reconhecimento de padrões, pois consiste em separar objetos por categorias de acordo com uma ou mais características específicas. Ou seja, devido a junção das técnicas de reconhecimento de padrões com o processamento digital de imagens tornou-se possível a detecção de objetos em imagens(NASCIMENTO, 2007).

O primeiro *framework* de métodos que usam base de dados categorizando uma ou mais características de um objetos para fazer o reconhecimento por meio de aprendizado foi apresentado por Viola e Jones (2001). Desde o *framework* de Viola e Jones até os dias atuais, muitos métodos e teorias para detecção já foram propostos e implementados, como detecção de faces utilizando um classificador de redes neurais na intensidade de padrões de uma imagem (ROWLEY; BALUJA; KANADE, 1998; GARCIA; DELAKIS, 2004), *support vector machine* para localizar rostos humanos e carros (OSUNA; FREUND; GIROSI, 1997; PAPAGEORGIOU; POGGIO, 1999), análise de componentes principais (JOLLIFFE, 2002), análise independente de componentes(HYVÄRINEN; KARHUNEN; OJA, 2004), fatoração de matriz não-negativa (LEE; SEUNG, 1999) , análise discriminativa linear (HART; STORK; DUDA, 2001), *boosting* (FREUND; SCHAPIRE, 1995), classificação binária, que considera a detecção do objeto em tamanho fixo apenas variando a posição na imagem (AMIT; FELZENZWALB, 2014), além de algoritmos de análise estrutural topológica por *border-following* (SUZUKI et al., 1985) que é utilizado na bibliotecas de processamento digital de imagens OpenCV (ITSEEZ, 2016).

### 2.2.2 Detecção de Bordas

Para um objeto poder ser destacado por algum método de detecção, a imagem passa por um processo de segmentação. A segmentação pode ser dita como o processo de divisão da imagem em objetos(GONZALEZ; WOODS, 2008). De acordo com WANGENHEIM (2014), o processo de segmentação se baseia em dois conceitos: similaridade e descontinuidade. A descontinuidade é o processo no qual se separa o fundo das partes e estas umas das outras, utilizando-se linhas, bordas ou pontos. Já a similaridade é o processo no qual os pixels provenientes da descontinuidade são agrupados de acordo com a proximidade um dos outros para formar os objetos de interesse.

De acordo com Canny (1986), a detecção de bordas é um processo simplificado que serve para diminuir drasticamente o total de dados a serem processados e ao mesmo tempo preservar informações valiosas sobre os objetos. Assim, como dito por Ziou e Tabbone (1998 apud VALE; POZ, 2002), se torna difícil encontrar um algoritmo que tenha bom desempenho em diferenciados contextos e capture os requisitos necessários aos estágios subsequentes do processamento. Quando se trata de detecção de bordas existem dois critérios que devem ser levados em consideração, Taxa de Erro e Localização (CANNY, 1986; VALE; POZ, 2002).

**Taxa de Erro** É importante que as bordas contidas na imagem não sejam confundidas ou perdidas, tampouco que não sejam detectadas bordas falsas. É necessário que o algoritmo de detecção de borda tenha uma baixa taxa de erro para que seja eficiente (WANGENHEIM, 2014; CANNY, 1986; VALE; POZ, 2002).

**Localização** A distância entre os pixels de borda encontradas pelo algoritmo e a borda atual deveriam ser o menor possível (WANGENHEIM, 2014).

Ao tentar aplicar esses dois critérios para desenvolver um modelo matemático para detecção de bordas, sem a necessidade de base em regras preestabelecidas. No artigo *A Computational Approach to Edge Detection*, Canny (1986) descreve que somente esses dois critérios não são suficientes para obter uma boa precisão da detecção de bordas, portanto, propôs um terceiro critério: Resposta.

**Resposta** Para contornar a possibilidade de mais de uma resposta para a mesma borda, ou seja, o detector de bordas não deveria identificar múltiplos pixels de borda onde somente existe um único pixel.

Com o acréscimo do terceiro critério, o processo de detecção de bordas de Canny (1986) mostrou-se bastante flexível, independente da origem da imagem utilizada (VALE; POZ, 2002).

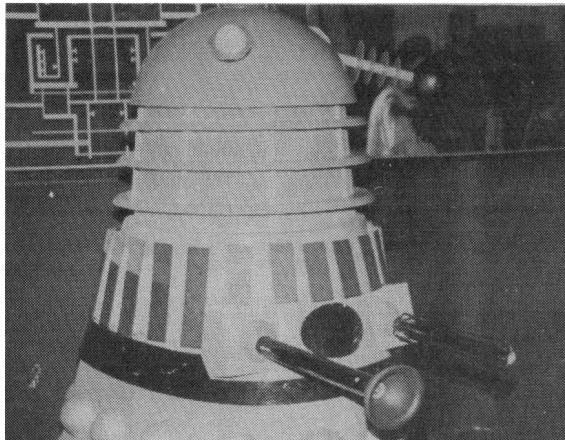


Figura 2.1: Imagem original (CANNY, 1986).

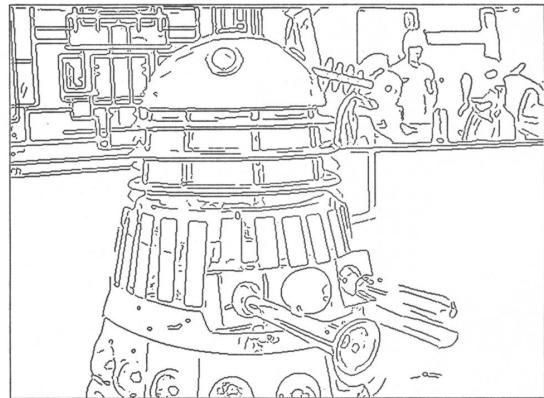


Figura 2.2: Bordas encontradas(CANNY, 1986).

## 2.3 Universo de Cores

Para redação desta seção foi tomada como base o Capítulo 5, Cores, de Azevedo e Conci (2003, pp. 181-212).

O olho humano é capaz de identificar cores mesmo com as mais diferentes interferências, luminosidade, tonalidade, intensidade, entre outras ações de agentes externos, graças aos **cones** e **bastonetes**. Os bastonetes são os responsáveis por distinguirem tons de cinza e pela visão periférica e tem como característica serem sensíveis a baixo nível de luminosidade, os cones, como descrito por Azevedo e Conci (2003), “são sensíveis ao alto nível de iluminação e responsáveis pela percepção de cores”. Segundo a teoria tricomática de Thomas Young, aprimorada por Hermann von Helmholtz, a retina humana é formada por três tipos de fotopigmentos que seriam os três receptores de cor, ou seja, respondiam ao comprimento de onda de apenas três cores: Vermelho, Verde e Azul (AZEVEDO; CONCI, 2003).

A informação obtida pelo sistema visual humano é assimilada pelo cérebro, ligando a cor a sua aparência, levando em consideração o aprendizado que obtivemos sobre a mesma.

Já para uma máquina, cores são representadas como números, na qual cada cor contém um código específico. Para o nosso cérebro é muito fácil entender, por exemplo, que as cores verde, verde lima e verde escuro são parte da mesma cor: verde, apenas possuem tonalidades diferentes. Para o computador cada uma das tonalidades é representada por um código diferente, e cada nível de luminosidade nessas cores, as modificam, tornando-se códigos diferentes das cores originais.

Quando fez sua primeira experiência com a decomposição da luz em um prisma para obter cores Newton percebeu que não havia a cor branca. Ele tentou então misturar as sete cores que obteve para gerar a branca, porém, não obteve sucesso. A resposta para o porque de Newton não ter conseguido chegar à cor branco veio somente em 1931, quando a Comissão Internacional de Iluminação(CEI) definiu três primárias (X, Y e Z) que combinadas formam todas as cores e que existe duas formas de se obter cores: através da emissão ou reflexão de luz (SOUTO, 2003). Assim, entendeu-se o porque em seu experimento Newton não ter sucesso. Para gerar a cor branca é necessário a soma das três cores primárias azul, verde e vermelho, uma vez que seu experimento utilizava a reflexão e não emissão de cores. Estas formas de se obter cores são conhecidas como **Sistemas de cores**. O sistema de cor relativo a emissão de luz é o RGB e o sistema de reflexão de luz é conhecido como CMY, ambos são mostrados na Figura 2.3.

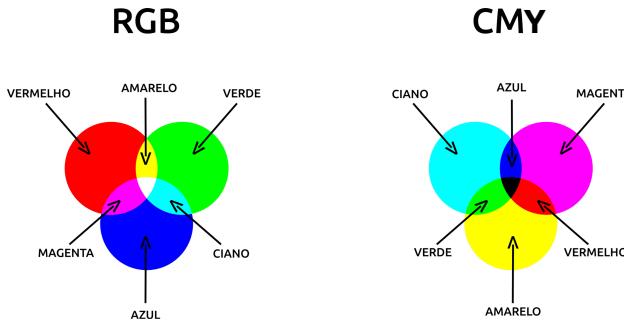


Figura 2.3: Exemplo dos sistemas de cores RGB E CMY.

Juntamente com a definição dos sistemas de cor, foi proposto o Diagrama de cromaticidade, exposto na Figura 2.4, representando as cores presentes no espectro da luz visível (SOUTO, 2003), ou seja, todas as cores puras visíveis. O diagrama é apresentado por uma forma plana contendo somente a cromaticidade das cores, luminosidade e intensidade são consideradas em um ponto invariável.

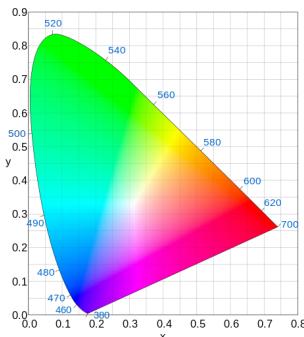


Figura 2.4: Diagrama de cromaticidade estabelecido pelo CIE.

Segundo Azevedo e Conci (2003) as cores que podem ser reproduzidas por um sistema cores é chamado de **Espaço de Cores**. Hughes et al. (2013 apud SOUTO, 2003) define o espaço de cores como sendo tridimensional com coordenadas correspondentes a cada uma das cores primárias. O valor do eixo é designado pela quantidade necessária de cor primária na reprodução de uma determinada cor. O espaço de cores pode ser entendido como a quantidade de detalhamento, tonalidades de uma cor, dentro do espectro de cores de um determinado modelo de cor.

Para utilização das teorias de sistemas e espaços de cores, foram criados os **Modelos de Cores** para descrever as cores na prática. Neste trabalho, será dado enfoque nos modelos RGB e HSV, devido a sua relevância na utilização da biblioteca OpenCV.

Modelo de cores são modelos matemáticos utilizados para classificação das cores de acordo com sua tonalidade, saturação, luminosidade ou crominância na tentativa de conseguir cobrir o maior número de cores possíveis e assim simulando a visão. A representação da cor é definida por um único ponto em um modelo tridimensional. Os modelo de cores tem como função definir as cores nos programas gráficos de computadores, de forma que combine com a percepção das cores pelo sistema visual humano que também utiliza três eixos similares para definição da cor (LEÃO; ARAÚJO; SOUZA, 2005).

O modelo de cores RGB pode ser considerado mais básico dos modelos de cores. Seu nome possui a mesma definição do espaço de cores RGB, e não utiliza atributos como luminosidade ou tonalidade, criando-se cores pela combinação das cores primárias azul, verde e vermelho (Figura 2.5). Este é o padrão mais usado e conhecido, e está baseado na teoria tricomática.

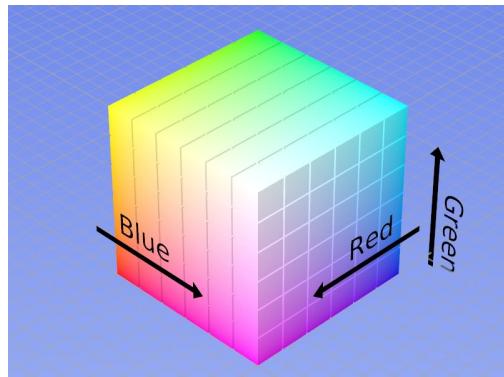


Figura 2.5: Exemplo do Modelo de Cor RGB. Horvath (2008)

O modelo HSV, representado pela Figura 2.6, é definido pela tonalidade(*Hue*) que é a cor em si, variando de 0 a  $360^\circ$ ; a saturação(*Saturation*) que define o grau de pureza da cor, variando de 0 a 1; e luminância (*Value*) que representa os tons de cíza, que faz referência à percepção humana, também variando de 0 a 1 (LEÃO; ARAÚJO; SOUZA, 2005; AZEVEDO; CONCI, 2003). O modelo HSV utiliza definições de cor mais intuitivas que o conjunto de cores primárias, por isso são mais adequados quando se necessita obter várias tonalidades da mesma cor.

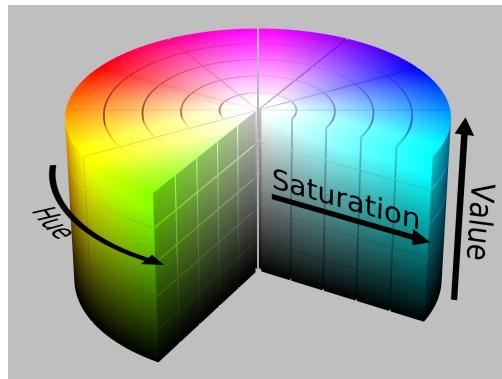


Figura 2.6: Exemplo do Modelo de Cor HSV, Horvath (2008)

## 2.4 Futebol de Robôs

Visto como um domínio bastante complexo, dinâmico e imprevisível, o futebol de robôs surgiu como uma tentativa de promover pesquisas nos campos de Inteligência Artificial e robótica, pela avaliação teorias, algoritmos e arquiteturas por meio de problemas (FARIA et al., 2006; COSTA; PEGORARO, 2000; KITANO et al., 1997). Além disso pode-se considerar o futebol de robôs um problema padrão, no qual se torna possível fazer a avaliação de algoritmos, arquiteturas, desempenhos e teorias (FARIA, 2006).

A Equipe Cedro se enquadra na categoria VSSS que é regulamentada pelo Instituto de Engenheiros Eletricistas e Eletrônicos(IEEE) e possui regras baseadas na MiroSot (TIENE, 2014). O futebol de robôs se assemelha ao futebol humano cujo o objetivo do jogo é fazer gols para vencer a partida, porém tendo regras adaptadas para o âmbito robótico. Rosa (2015) e Tiene (2014) fazem boas enumeração das regras básicas:

- A equipe deve ser constituída por 3 robôs, sendo 1 destes o goleiro;
- Cada robô deve ter seu tamanho limita à 7,5 cm x 7,5 cm x 7,5 cm;
- A partida dura 10 minutos com dois tempos de 5 minutos;
- Há um intervalo de 10 minutos entre um tempo e outro;
- Cada time tem direito a dois tempos de 2 minutos que podem ser pedidos a qualquer momento;
- Durante o tempo de jogo somente são permitidas 2 substituições;
- Caso a diferença de gols entre os dois times chegue a 10 a partida é encerrada;
- Uma falta ocorre quando há mais de um robô de um mesmo time dentro de sua própria área de gol ou quando um robô empurrar outro robô de outro time;
- Um pênalti ocorre quando a bola fica mais de 10 segundos dentro de alguma das áreas;
- Um chute-livre ocorre quando os robôs ficam travados por mais de 10 segundos, caso ocorra, o juiz posiciona a bola na marca de chute-livre mais próxima de onde ela ficou parada e posiciona os robôs de cada time equidistantes a bola;

- A câmera usada pelo sistema de visão computacional deve estar acima do centro do campo, para que não seja necessário ser reposicionada quando os times trocarem de lado;
- A câmera deve estar à uma altura mínima de 2 metros;
- A cada inicio de partida ou gol feito a bola deve ser posicionada no centro do campo e os robôs devem ser posicionados de acordo com a posse de bola.

Uma ótima descrição do ambiente de futebol de robôs foi dada por Faria (2006) e pode ser visualizada na Figura 2.7:

*“(...) futebol de robôs considera-se uma câmera para capturar a imagem de todo o campo e utilizando-se de recursos de processamento de imagem, um computador poderá fornecer as posições dos robôs pertencentes ao time, dos adversários, do gol e da bola. Um computador, chamado de técnico, pode utilizar algum sistema de controle que utilize as informações disponíveis para dar ordens de como os robôs de seu time devem agir.”(FARIA, 2006)*

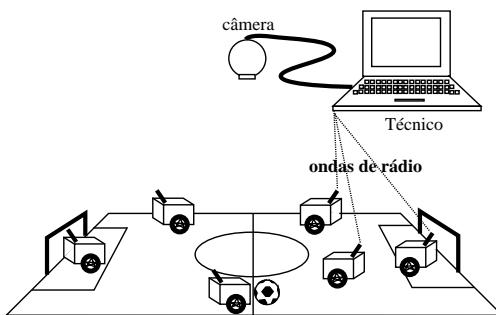


Figura 2.7: Estrutura de um ambiente de futebol de robôs VSSS(FARIA, 2006)

Pode-se considerar a câmera uma das partes mais importantes de uma equipe. É ela que torna possível identificar a localização dos robôs dentro do campo, para que o sistema de estratégia da equipe consiga escolher a melhor jogada a ser feita a partir da posição dos robôs.

## 2.5 Considerações Finais

A detecção de bordas é uma técnica de segmentação que prepara a imagem para a detecção de objetos. Utilizando-se o Algoritmo de Canny, foi minimizado os erros da detecção de bordas, assim, conseguindo detectar objetos com maior precisão.

Como estudado, os modelo de cores têm como base o sistema visual humano para simular as cores em âmbito computacional, e contam com a utilização três eixos para seguir a teoria tricromática do olho humano. Mesmo o RGB sendo o mais básico e utilizado, é o modelo HSV que consegue lidar melhor com diferentes tonalidades de uma cor.

# Capítulo 3

## Desenvolvimento

### 3.1 Considerações Iniciais

Neste capítulo descreve-se o sistema proposto, denominado SCIMM, as tecnologias utilizadas no projeto, a organização do projeto e suas classes, o fluxo do sistemas e cada uma das etapas do seu funcionamento.

Para o desenvolvimento foi escolhida a biblioteca OpenCV por ser de código aberto<sup>1</sup>, multiplataforma, conter uma grande quantidade de métodos e algoritmos já implementados e pelo seu rápido desempenho de máquina.

A linguagem escolhida para o desenvolvimento foi o C++, pois é uma linguagem de programação compilada, o que torna sua execução mais rápida que as linguagens interpretadas, dando ao sistema uma performance em tempo satisfatório.

Todo o projeto foi desenvolvido em sistema operacional Ubuntu e utilizou-se a ferramenta *Github* como repositório *online* para o controle de versão do projeto. A escolha desse repositório se deve ao fato da plataforma conter um grande número de projetos de código aberto, e uma vez que este trabalho tem como finalidade principal beneficiar as equipes de times de futebol de robôs, deixar seu código aberto permite que melhorias possam ser adicionadas além de soluções para possíveis erros.

### 3.2 Tecnologias Usadas

Para realização deste trabalho, utilizou-se a biblioteca de processamentos de imagens OpenCV. O trabalho foi elaborado na linguagem C++, com uso do *framework* Qt para a composição da interface gráfica.

**OpenCV** foi lançado em 1999 pela Intel(CULJAK et al., 2012), com objetivo de ser otimizada, portável e com um grande número de funções, se tornou uma ferramenta que possui mais de 2500 algoritmos e 40 mil pessoas em seu grupo de usuários(CULJAK et al.,

---

<sup>1</sup>Modelo de desenvolvimento de *software*, no qual o código fonte é aberto e livre para modificações e adaptações

2012). Já possui interface para as linguagens C++, C, Python e Java, além de suporte para as principais plataformas como Windows, Linux, Mac OS, iOS e Android. A biblioteca pode ser utilizada para manipular tanto imagens em tempo real, quanto vídeos e imagens estáticas.

**Qt** é um *framework* de desenvolvimento de aplicações multiplataforma. Entre suas funcionalidades está a possibilidade de criar interfaces gráficas diretamente em C++ usando seu módulo *Widgets*.

C++ é uma linguagem de programação projetada por Stroustrup (1996) para fornecer eficiência e flexibilidade da linguagem C para programação de sistemas. A linguagem C++ evoluiu a partir de um projeto chamado C com Classes que foi realizado entre 1979 e 1983. A linguagem foi oficialmente lançada em 1986.

## 3.3 Descrição do Projeto

### 3.3.1 Organização do Projeto

O projeto foi desenvolvido seguindo o paradigma de programação Orientada à Objetos, esse paradigma baseia-se na utilização de objetos individuais para criação de um sistema maior e complexo. A IDE usada para o desenvolvimento foi a QT Creator, esta separada o projeto em três pastas: *Headers*, *Sources* e *Forms* (Figura 3.1). Na pasta *Headers* estão os arquivos de cabeçalho(.h), onde estão as declarações dos métodos e variáveis usados nas classes executáveis. Na pasta *Sources* estão os arquivos fonte(.cpp), são nesses arquivos que os métodos declarados nos arquivos da pasta *Headers* são implementados. Na pasta *Forms* está o arquivo de interface gráfica(.ui) que é usado no projeto.

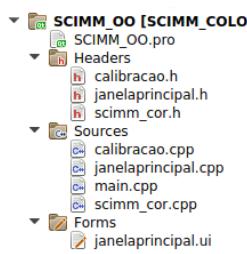


Figura 3.1: Organização das pastas do projeto

### 3.3.2 Apresentação das Classes

#### Classe main

Esta é o que se chama de *ponto de entrada* em programação. *Ponto de entrada* é onde o sistema operacional irá iniciar a execução do sistema desenvolvido. Esta classe possui somente o método *main* e este instancia e inicia o objeto **JanelaPrincipal** que apresenta a interface gráfica para interação com o usuário.

### Classe JanelaPrincipal

Nesta classe utiliza-se da implementação de objetos *QWidget* e suas subclasses disponíveis pelo Qt, para a criação de uma interface gráfica que faz a interação com o usuário. Todos os métodos presentes nessa classe são para utilização gráfica, comportamento de botões e menu de seleção, e para a comunicação com a classe de calibração.

### Classe Calibracao

Esta classe contém todos os métodos e variáveis usados no processo de calibração, é também dentro dessa classe que são feitas todos os tipos de manipulação em imagem.

Os métodos contidos na classe Calibração são:

Iniciar: Método utilizado para fazer referencia à **JanelaPrincipal**, devido a necessidade de comunicação com a interface que não seja por meio de retorno. Este método também inicializa a câmera, se esta estiver disponível.

ConfigurarCamera: Método utilizado para delimitar, um retângulo, ou seja, o espaço de trabalho dentro da imagem proveniente da câmera. Também é utilizada para ajuste de brilho e contraste para tornar mais nítido os contornos durante a detecção de objetos dentro do retângulo já delimitado.

ReconhecerFundoExtrairObjetos: Método no qual utiliza-se o algoritmo de subtração de fundo. Identifica-se, inicialmente, o campo e uma vez identificado, qualquer objeto colocado sobre o campo ficará em destaque pois não faziam parte do fundo inicial. Esse destaque é uma **máscara** sobre a imagem que é usada melhorar a precisão do algoritmo de detecção dos objetos.

DetectarObjetos: Método onde serão aplicados filtros de diminuição de ruido, detectadas as bordas existentes na imagem, detectados os objetos a partir das bordas contidas na imagem, o aumento da precisão do contorno dos objetos e diminuição do tamanho do objeto de acordo com a porcentagem de borda a ser eliminada.

Calcular: Método utilizado para analisar cada pixel pertencente aos objetos detectados. O valor H do HSV é categorizado de acordo com as cores pre-definidas no sistema.

Calibrar: Método utilizado para administrar a utilização dos métodos **DetectarObjetos** e **Calcular**.

ObterPorcentagem: Método simples para devolver a porcentagem de um valor, ao ser informado o valor e a porcentagem escolhida.

### Classe SCIMM\_COR

Classe na qual se instancia cada cor que é reconhecida pelo sistema. Os objetos desse tipo armazenam os valores máximos e mínimos de H, S e V.

## 3.4 Detalhes de Implementação do Sistema SCIMM

O sistema consiste na apresentação da **interface gráfica** objetiva. Possuindo seis botões, um menu de escolha e uma janela de exibição. Na Figura 3.2 pode ser observado o fluxo do sistema, que se constitui em cinco etapas com o mínimo de ação possível do usuário.

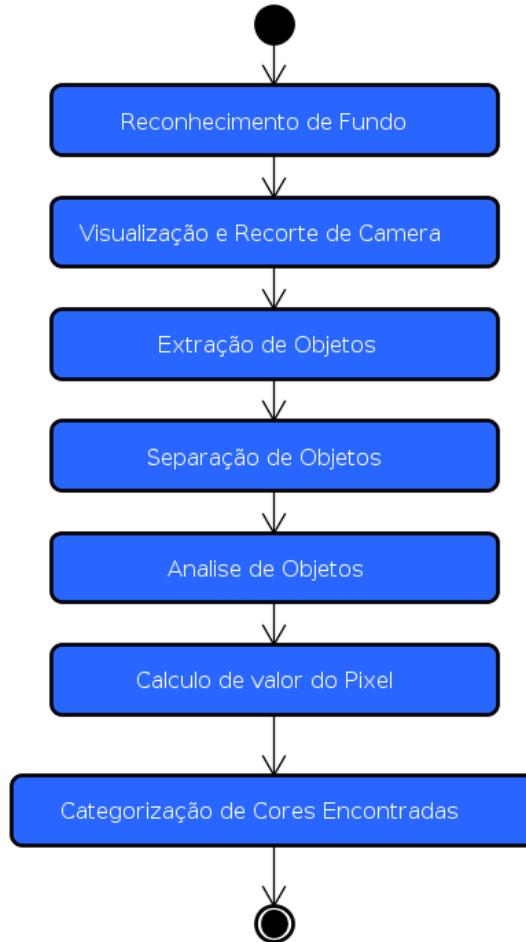


Figura 3.2: Diagrama de Fluxo

Nas subseções à seguir serão detalhadas cada uma das etapas do processo de calibração automática.

### 3.4.1 1<sup>a</sup> Etapa - Configuração de Câmera

Para melhor desempenho do algoritmo de detecção de objetos, a imagem é recortada somente para o tamanho necessário. O recorte de imagem foi feito utilizando a função *setMouseCallback*, disponível pelo OpenCV, para possibilitar a interação do usuário na imagem utilizando o mouse, sua utilização é dada da seguinte maneira:

```
cv::setMouseCallback(src_window, mouseHandler, 0);
```

O primeiro parâmetro, **src\_windows**, indica a janela na qual a função receberá a interação. O segundo, **mouseHandler**, indica a função na qual esta implementada a interação. Já o último parâmetro, **0**, indica parâmetros opcionais. Dentro da função **mouseHandler** são identificados os dois pontos da seleção, inicio e fim, além da utilização da função *rectangle* para demarcar a seleção na tela. A função *rectangle* foi utilizada da seguinte maneira:

```
cv::rectangle(frameA, point1, point2, CV_RGB(255, 0, 0), 2, 5, 0);
```

A função recebe os parâmetros **frameA** indicando a imagem na qual será demarcada a área selecionada, depois o parâmetro **point1** que é o ponto inicial de seleção na imagem, **point2** que é o ponto final da seleção. **CV\_RGB(255, 0, 0)** que indica a cor da demarcação, **2** indicando a espessura da demarcação, **5** que significa o tipo de linha a ser utilizado na demarcação e **0** que é o número de bits fracionários.

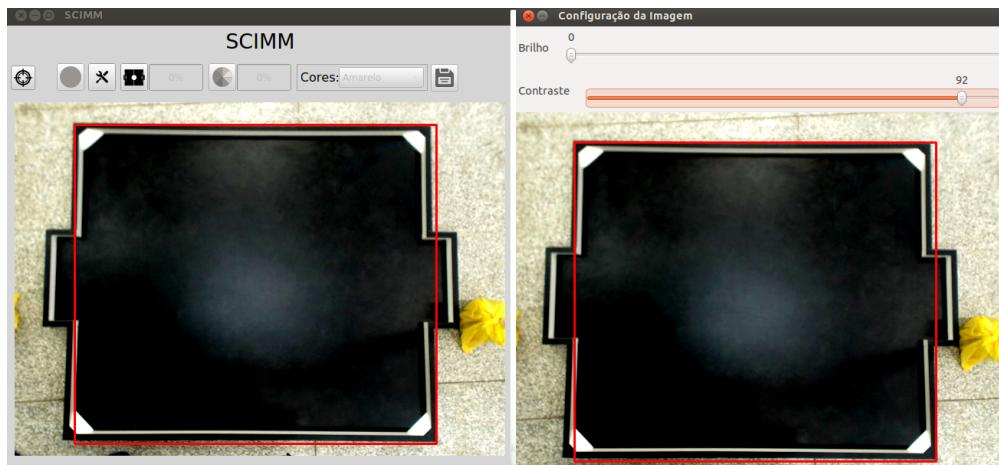


Figura 3.3: Configuração de Camera

Após confirmada a escolha, o tamanho da tela é salvo na variável nomeada *tamanho*, usado durante todo o processo de calibração. Além do recorte da imagem, nessa etapa também são configurados Brilho e Contraste, de forma que as cores no campo se tornem mais vívidos, e os objetos consequentemente mais nítidos, fazendo com que os processos de separação e detecção de objetos sejam mais precisos. A configuração de Brilho e Contraste utiliza o método *convertTo* da biblioteca, que é utilizada para o melhoramento da imagem antes da detecção dos objetos. A utilização do método se deu por:

```
frameA.convertTo(frameA, -1, contrast_value / 50.0, brightness_value -200)
```

Esta função recebe quatro parâmetros. O primeiro **frameA** será o resultado da conversão. O segundo **-1** indica o tipo da matriz, ou número de canais, da imagem a ser gerada, usa-se **-1** quando se deseja que se use os valores semelhantes aos da imagem original(ITSEEZ, 2016). O terceiro **contrast\_value / 50.0** indica o valor de contraste, ou alpha, a ser usado para multiplicar os valores do pixel da imagem(ITSEEZ, 2016) e por último **brightness\_value -200** que é o valor do brilho, ou beta, a ser adicionado à imagem. É importante ressaltar que a configuração de Brilho e Contraste é somente usada para a melhor precisão na detecção dos objetos, no momento da análise do pixel, a imagem não contém alterações.

### 3.4.2 2<sup>a</sup> Etapa - Reconhecimento de Fundo

Um dos principais problemas que ocorrem na detecção de objetos é a confusão do fundo junto ao próprio objeto, fazendo com que o mesmo seja detectado com o contorno incorreto ou em outras vezes ignorando-o por ser considerado parte do fundo. Para eliminar este problema, foi utilizada a técnica de Subtração do fundo usando Mistura de Gaussianas, por meio do objeto *createBackgroundSubtractorMOG2()* disponível na biblioteca OpenCV. Esta técnica utiliza um algoritmo de análise pixel a pixel, classificando-o com base na distribuição da gaussiana que o representa. Para separar o fundo do resto da imagem é levada em consideração que a gaussiana que representa o fundo tenha grande peso e baixa variância, isso significa que a mesma ocorre frequentemente e varie pouco com o tempo. O algoritmo atualiza o modelo de fundo a cada quadro da imagem, baseando-se na movimentação do objetos já contido na imagem ou na inserção ou remoção de objetos. O objeto criado pela biblioteca, por meio do método *apply*, analisa quadro a quadro a imagem, e a compara com o fundo obtido e gera uma imagem chamada de **máscara**, contendo os objetos que não fazem parte do fundo, no exato momento de um quadro, com o passar dos quadros o objeto se torna parte do fundo. Uso do método:

```
pMOG2->apply(frame, mask);
```

No qual **frame** significa o quadro atual capturado pela câmera e **mask** a máscara gerada pela diferença da imagem atual com o modelo de fundo.

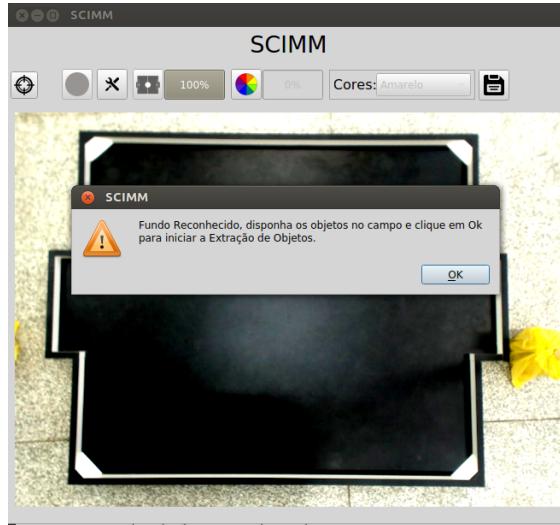


Figura 3.4: Configuração de Camera

Sabendo-se inicialmente que o modelo de fundo do objeto **pMOG2** está vazio, basta que ele seja executado algumas vezes para que o campo se torne o modelo de fundo. Nesse caso a máscara gerada pelo método não chega a ser utilizada.

### 3.4.3 3<sup>a</sup> Etapa - Extração dos Objetos do Fundo

Como visto na 2<sup>a</sup> etapa, o objeto **pMOG2** chamando o método *apply* gera uma máscara de diferença da imagem atual para o modelo de fundo. Sendo assim para obtermos os objetos

que virão a ser detectados, basta que o método *apply* seja chamado um número de vezes suficientes para criar a máscara ao mesmo tempo que não altere o modelo de fundo.



Figura 3.5: Geração de Máscara

### 3.4.4 4<sup>a</sup> Etapa - Detecção e validação de Objetos

Para ser feita a detecção dos objetos é necessário que primeiro seja feita a detecção de bordas dos objetos. Como recurso para eliminação de ruídos e melhoria da imagem, antes de ser executada a detecção, utiliza-se o desfoque na imagem. O algoritmo de detecção de bordas utilizado no desenvolvimento foi o algoritmo de Canny, descrito na Seção 2.2.2, que é implementado pela biblioteca OpenCV, e foi utilizado neste trabalho como segue:

```
Canny(src_gray, canny_output, limiar, limiar * 3, 3);
```

O algoritmo de Canny utiliza por padrão imagem em escalas de cinza, sendo assim **src\_gray** é a imagem original transformada para escala de cinza, na qual o algoritmo será aplicado. **canny\_output** será a imagem de saída da função. **limiar** e **limiar\*3** são os limites mínimos e máximos, respectivamente, para considerar uma borda. **3** é o valor de apertura ou kernel, sendo o valor 3 é utilizado como padrão.

Após o uso do algoritmo de Canny para detecção de bordas, é necessário então fazer uso da função *findContours*, nativa no OpenCV que utiliza do algoritmo *border following* para fazer a detecção de objetos aqui chamados de contornos.

```
findContours(canny_output, contours, hierarchy, CV_RETR_EXTERNAL,
            CV_CHAIN_APPROX_SIMPLE, Point(0, 0))
```

O primeiro parâmetro, **canny\_output**, é a imagem que o algoritmo de Canny gerou com as bordas encontradas, e que será utilizada pelo método *findContours* para detectar os contornos. **contours** é o parâmetro que armazenará os contornos encontrados, no qual cada contorno é armazenado como sendo um vetor de pontos. Na variável **hierarchy** será salvo um vetor de informações sobre a topologia da imagem, e terá como total de elementos o mesmo número que o total de contornos encontrados. O quarto

parâmetro, **CV\_RETR\_EXTERNAL** indica o modo de obtenção de contornos, nesse caso **CV\_RETR\_EXTERNAL** indica que o método só obterá os contornos exteriores. **CV\_CHAIN\_APPROX\_SIMPLE** designa o método de aproximação de contornos utilizado, o método utilizado comprime segmentos horizontais, verticais e diagonais os deixando apenas com seus pontos finais. O último parâmetro, **Point(0, 0)**, indica o valor a ser usado para deslocar a imagem ao encontrar os objetos, neste caso esse valor é 0 para Y e 0 para X, pois não haverá deslocamento (ITSEEZ, 2016).

Uma vez obtidos os contornos é necessários que se faça a eliminação de vértices dos polígonos encontrados nos objetos, deixando o objeto mais preciso. Isso é necessário para deixar a forma encontrada mais semelhante com a forma original do objeto. Para este ajuste foi usado o método *approxPolyDP*, já implementado dentro na biblioteca OpenCV. Este método foi aplicado em cada um dos contornos encontrados, e foi utilizado como mostrado a seguir:

```
approxPolyDP(Mat(contours[i]), contours_poly[i], 3, true)
```

O método inicia recebendo como parâmetro **Mat(contours[i])**, que é a criação de uma nova imagem, somente com aquele único objeto que esta sendo analisado. A seguir é informado no segundo parâmetro a variável de destino **contours\_poly[i]**, na qual será armazenado o objeto com a eliminação dos vértices. O terceiro parâmetro indica o valor do *epilson*, usado o valor **3**, a distância máxima entre a curva original para sua aproximação (ITSEEZ, 2016). O último parâmetro indica se a curva aproximada será fechada ou não, foi usado o valor **true**.

Por último os objetos possuem sua borda ignorada, calculando o tamanho interior deles para que, por ventura, não hajam pixeis de cor preta ou escurecidos devido a proximidade da borda.

### 3.4.5 5<sup>a</sup> Etapa - Classificação do Pixel

Para ser feita a classificação do pixel é necessário que se faça primeiramente a conversão da imagem obtida pela câmera. A imagem proveniente da câmera está no modelo de cores RGB, e será convertida para o modelo HSV, pois este lida melhor com diferentes tonalidades.

A biblioteca OpenCV converte o espaço de cor usando a função *cvtColor*, que recebe como parâmetro a imagem original, e uma imagem vazia com memória alocada, onde será salva a imagem após a conversão, além do parâmetro do tipo de conversão. Exemplo do uso do método:

```
cvtColor(frame, HSV, CV_RGB2HSV);
```

Após a conversão, é necessário ser feita uma análise dos objetos encontrados. Para cada objeto serão analisados todos os seus pixeis separadamente, e categorizados de acordo com o intervalo de valores de H representados na Tabela 3.1. Esta tabela foi gerada, durante a pesquisa e implementação do sistema SCIMM, a partir das observações da representação do modelo de cor HSV do OpenCV, que difere da teoria das cores.

Cor	Intervalo de H
Laranja	de 0 à 20
Amarelo	de 21 à 30
Verde	de 61 à 90
Azul	de 91 à 120
Roxo	de 125 à 160
Rosa	de 161 à 168
Vermelha	de 169 à 180

Tabela 3.1: Intervalo de Valores de Cores - HSV do OpenCV

Durante o desenvolvimento foi observado que, em sua maioria, as cores necessitavam de um valor (S,V) pré estabelecidos. Para as cores Laranja, Azul, Rosa e Vermelho foi pré estabelecido o valor (100,100). A cor Amarelo usa a pré definição (50,50). A cor Verde é a única que possui os valores pré definidos diferenciados um do outro, tendo 30 para S e 50 para V, ou seja (30,50). Já a cor Roxo, (30,30), possui tanto para S quanto para V o valor 30.

### 3.4.6 6<sup>a</sup> Etapa - Gerar Arquivo de Cores

Assim que todas as cores já estiverem sido assimiladas e a calibração finalizada, gera-se um arquivo chamado **cores.arff**, contendo 14 linhas. Sabendo que o sistema calibra 7 cores, o arquivo gerado contém duas linhas para cada cor, sendo a primeira com os valores mínimos de H, S e V e a segunda com os valores máximos.

## 3.5 Considerações Finais

O projeto desenvolvido conta com funcionalidades bem divididas, e cada classe exerce somente um único papel. Com a utilização da biblioteca OpenCV, conseguiu-se atender todos os objetivos relacionados ao processamento digital de imagens necessários para o desenvolvimento do sistema proposto.

O sistema consiste em um único módulo, em cinco partes principais: (1) detecção de bordas da imagem, (2) identificação dos objetos por meio das bordas encontradas, (3) análise de pixéis nos objetos, (4) categorização do HSV do pixel e (5) criação do arquivo com os valores mínimos e máximos para cada cor. É importante destacar que todos os processos do sistema acontecem de forma automática.

# Capítulo 4

## Testes e Resultados

### 4.1 Considerações Iniciais

Este capítulo está dividido em duas seções, nas quais estão presentes os testes realizados bem como se discute os mesmos. Primeiramente se descreve o processo de calibração de cores bem como a preparação do campo, configuração de imagem, e seu resultado. Em seguida são apresentados os resultados dos testes executados com o arquivo gerado na calibração. Os resultados foram separados em Cores Comuns e Cores com Problemas. Por último se faz uma análise geral dos resultados obtidos.

Para realização dos testes foi desenvolvido um aplicativo Desktop para fazer a leitura do arquivo gerado pela calibração, que possibilita a exibição de uma imagem contendo os objetos de acordo com a cor selecionada, por meio de um menu de escolha. Sendo assim possível a análise dos resultados obtidos para cada uma das cores.

O teste foi feito de uma só vez, porém sua análise está separada em duas partes. A primeira parte Cores Comuns, são as cores que não apresentam problema devido a semelhança com outras cores, na qual estava presente as cores amarelo, azul, verde, rosa, roxo. Na segunda parte estão presentes as Cores com Problemas, que contém as cores vermelho e laranja.

A seguir está detalhado o processo de calibração, bem como a maneira com que o mesmo ocorreu. Após a explanação da calibração estão expostos os resultados deste teste e a validação de cada um.

### 4.2 Calibração

A calibração descrita a seguir ocorreu no dia 19 de Agosto de 2016, entre 17:36 e 17:39. A rotina de calibração do sistema, já descrita no Capítulo 3, envolve primeiramente uma aquisição da imagem do campo vazio, como visto na Figura 4.1.

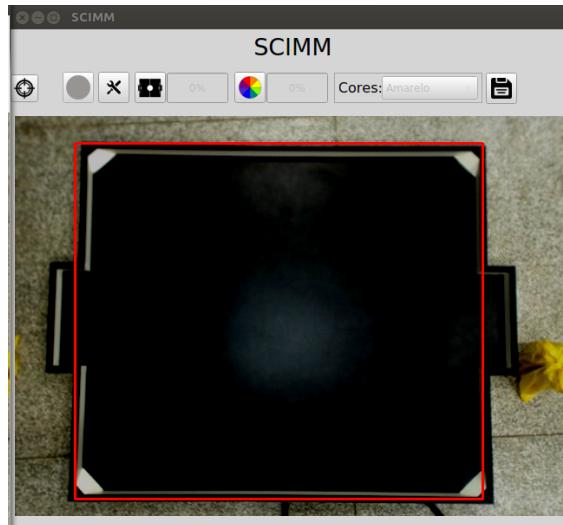


Figura 4.1: Imagem do fundo com a seleção de campo

Após ter o campo identificado pelo sistema, deve-se dispor sobre o campo os objetos coloridos, com as cores que se deseja obter o intervalo. É preferido que se usem tiras coloridas, pois quanto maior o tamanho da tira de cor, maior a quantidade de pixels representando os vários espectros da cor, melhorando sensivelmente o processo de calibração.

Neste teste foram dispostos no campo tiras coloridas com largura entre  $17\text{cm}$  e  $40\text{cm}$  e altura entre  $5,5\text{cm}$  e  $10,5\text{cm}$ . Cada cor com 3 tiras, uma vez que o campo foi separado em três partes, com a finalidade de capturar diferentes luminosidades. A disposição das tiras pode ser vista na Figura 4.2.

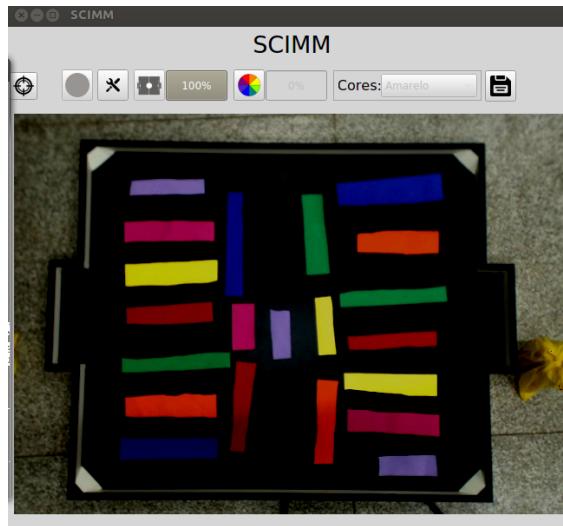


Figura 4.2: Objetos dispostos no campo para calibração

Como resultado da rotina de calibração, foi gerado um arquivo .arff contendo 14 linhas. A Tabela 4.1 possui uma explicação sobre o arquivo gerado. A primeira coluna da tabela designa a linha contida no arquivo, a segunda coluna possui o conteúdo existente no arquivo, na última coluna está a descrição do conteúdo de cada linha do arquivo.

Linha	Conteúdo	Descrição
1	21.50.50	Valor mínimo da cor Amarelo
2	30.255.255	Valor máximo da cor Amarelo
3	92.100.100	Valor mínimo da cor Azul
4	120.255.255	Valor máximo da cor Azul
5	62.30.100	Valor mínimo da cor Verde
6	90.255.255	Valor máximo da cor Verde
7	169.100.100	Valor mínimo da cor Vermelho
8	179.255.255	Valor máximo da cor Vermelho
9	0.100.100	Valor mínimo da cor Laranja
10	20.255.255	Valor máximo da cor Laranja
11	161.100.100	Valor mínimo da cor Rosa
12	168.255.255	Valor máximo da cor Rosa
13	126.30.30	Valor mínimo da cor Roxo
14	160.255.255	Valor máximo da cor Roxo

Tabela 4.1: Os valores de HSV estão separados por pontuação, H.S.V

### 4.3 Testes

O teste aqui descrito ocorreu dia no 26 de Agosto de 2016, entre 13:28 e 16:28. Para elaboração do teste optou-se por dividir o campo no maior número de partes possíveis, e ainda assim que essas partes coubessem todas as 7 cores calibradas. Essa divisão foi feita para simular as cores em todas as partes possíveis do campo. Sendo assim, o campo foi dividido em 15 partes de  $29cm \times 41cm$ , nomeadas alfabeticamente de A a O, como mostrado na Figura 4.3.



Figura 4.3: Divisão do campo em quinze partes nomeadas alfabeticamente.



Figura 4.4: Imagem do campo com marcadores de cores

Em cada uma das 15 partes do campo foi colocado marcadores representando as sete cores: Vermelho, Amarelo e Azul na primeira linha; Verde, Roxo, Laranja e Rosa na segunda. As cores estão distantes verticalmente  $6cm$ . Na primeira linha a distância horizontal entre as cores é de  $7,25cm$  e na segunda linha esta distância é de  $5cm$ . Estas distâncias entre os marcadores de cores e suas disposições podem ser vistas nas Figuras 4.4 e 4.5.

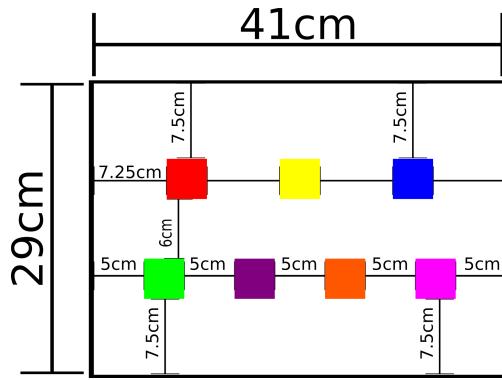


Figura 4.5: Disposição de cada parte quanto as cores

Quanto ao tipo de objeto encontrado, foram criados 7 categorias: Objetos Completos; Objetos com Falha de Preenchimento; Objetos com Diminuição de Contorno; Objetos com Diminuição de Área; Objetos com Falhas Críticas e Objetos Extrapolados.

**Objetos Completos** São os objetos que não contém nenhuma falha e foram encontrados corretamente.

**Objetos com Falha de Preenchimento** São os objetos que contém apenas falha de preenchimento, contendo seu contorno encontrado de forma correta.

**Objetos com Diminuição de Contorno** Foram os objetos encontrados com seu contorno reduzido devido à não detecção de sua borda.

**Objetos com Diminuição de Área** Foram os objetos que foram encontrados porém sua área foi reduzida, e.g. Contém preenchimento e borda, mas a área encontrada é igual à metade do tamanho real do objeto.

**Objetos com Falhas Críticas** São os objetos que apresentam mais de uma falha, diminuição de contorno, diminuição de borda ou diminuição de área. Estes objetos geralmente possuem sua aparência deformada, o que torna sua identificação pelos algoritmos de detecção de objetos.

**Objetos Extrapolados** São os objetos detectados dentro do intervalo uma cor específica, que fazem parte de outra cor.

### 4.3.1 Cores Comuns

#### Amarelo

Dentre os objetos da cor amarela o sistema encontrou quatorze deles completamente (Figura 4.7), e apenas um, que devido a luminosidade implicada em seu centro deixando a tonalidade muito perto do branco, não totalmente preenchido.

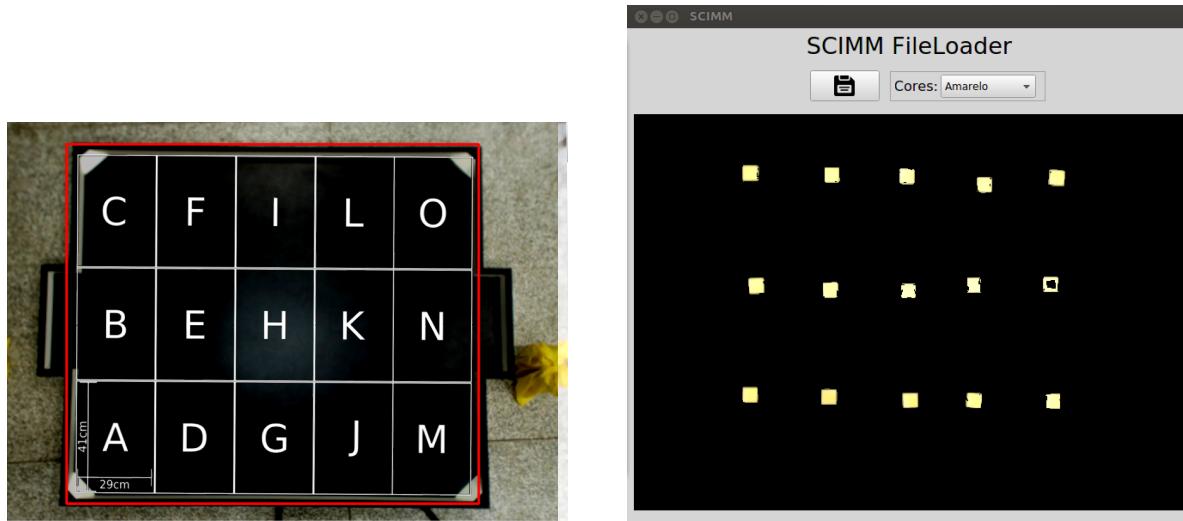


Figura 4.6: Divisão do campo.

Figura 4.7: Objetos da cor amarela

O único objeto detectado com falha encontra-se na parte N do campo. O detalhamento das quantidades e porcentagens dos objetos encontrados está na Tabela 4.2.

Tipo de Objeto	Quantidade	%
Objetos Completos	14	93,33
Objetos Com Falha de Preenchimento	1	6,66
Objetos Com Diminuição de Contorno	0	
Objetos Com Diminuição de Área	0	
Objetos Com Falhas Críticas	0	
Objetos Extrapolados	0	

Tabela 4.2: Categorização Dos Objetos

Dentre as classificações dos objetos as que descaracterizam o objeto para detecção são: *Objetos Com Falhas Críticas* e *Objetos Com Falha de Preenchimento*, que para a cor amarela resultou em 6,66% dos objetos.

### Azul

Os objetos da cor azul foram os que obtiveram os melhores resultados, os quinze objetos foram encontrados de forma preenchida. Apesar dos quinze estarem totalmente preenchidos um dos objetos apresentou um tamanho reduzido aos demais devido ao fato de sua borda não ter sido totalmente detectada(Figura 4.9).



Figura 4.8: Divisão do campo.

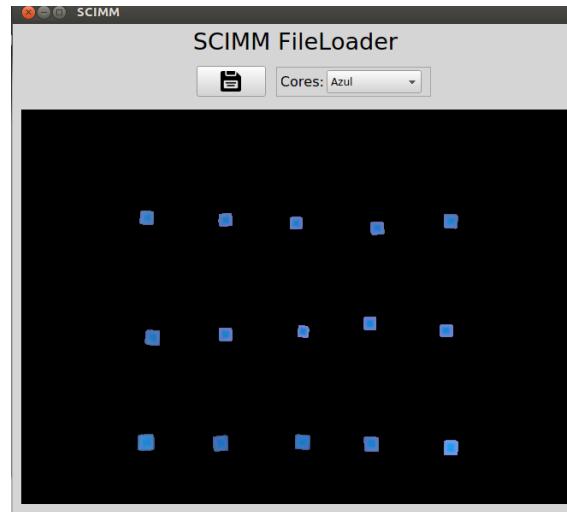


Figura 4.9: Os objetos da cor azul

O único objeto que conteve falha está presente na parte H do campo. Este não teve sua borda identificada dentro do intervalo de cor. O detalhamento das quantidades e porcentagens dos objetos encontrados está na Tabela 4.3.

Tipo de Objeto	Quantidade	%
Objetos Completos	14	93,33
Objetos Com Falha de Preenchimento	0	
Objetos Com Diminuição de Contorno	1	6,66
Objetos Com Diminuição de Área	0	
Objetos Com Falhas Críticas	0	
Objetos Extrapolados	0	

Tabela 4.3: Categorização Dos Objetos

### Verde

Os objetos da cor verde foram satisfatoriamente encontrados, com seu preenchimento total e não havendo perda de área devido a qualquer interferência de luz em sua borda (Figura 4.11).



Figura 4.10: Divisão do campo.

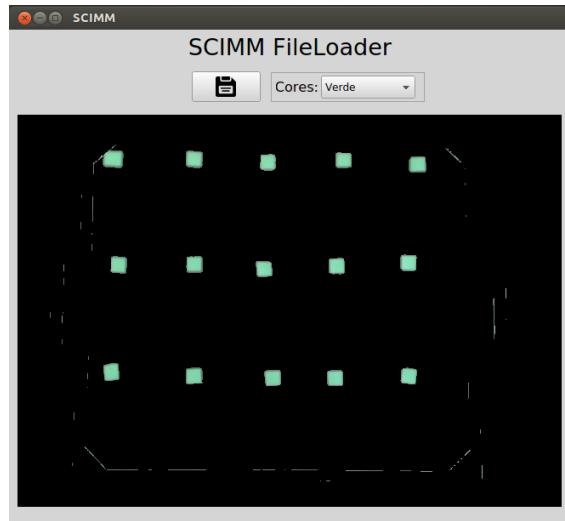


Figura 4.11: Objetos da cor verde

O detalhamento das quantidades e porcentagens dos objetos encontrados está na Tabela 4.4.

Tipo de Objeto	Quantidade	%
Objetos Completos	15	100
Objetos Com Falha de Preenchimento	0	
Objetos Com Diminuição de Contorno	0	
Objetos Com Diminuição de Área	0	
Objetos Com Falhas Críticas	0	
Objetos Extrapolados	0	

Tabela 4.4: Categorização Dos Objetos

### Rosa

Dentre os objetos Rosa detectados, quatro obtiveram falhas críticas em sua detecção, devido ao mal preenchimento e a deformação/diminuição de contorno, partes A, B, D e M do campo.



Figura 4.12: Divisão do campo.

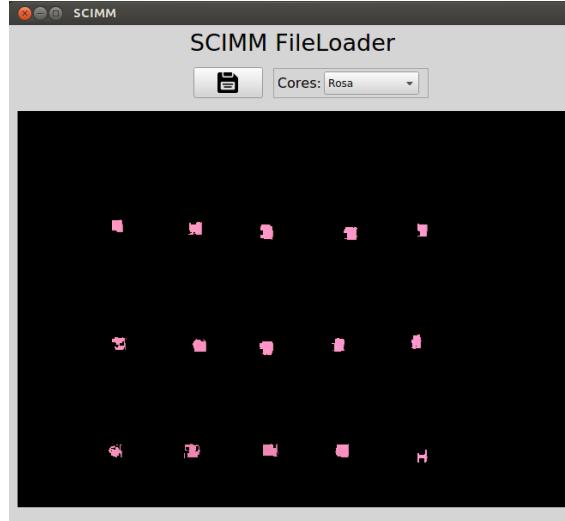


Figura 4.13: Objetos da cor rosa

Dentre os onze restantes: três foram detectados com um pequena diminuição de área, partes K, N e O; os outros oito foram encontrados apenas com diminuição de contorno (Figura 4.13). O detalhamento das quantidades e porcentagens dos objetos encontrados está apresentado na Tabela 4.5,

Tipo de Objeto	Quantidade	%
Objetos Completos	0	
Objetos Com Falha de Preenchimento	0	
Objetos Com Diminuição de Contorno	8	53,33
Objetos Com Diminuição de Área	3	20
Objetos Com Falhas Críticas	4	26,66
Objetos Extrapolados	0	

Tabela 4.5: Categorização Dos Objetos

Dentre as classificações dos objetos as que descaracterizam o objeto para detecção são: *Objetos Com Falhas Críticas* e *Objetos Com Falha de Preenchimento* que juntos somam 26,66% dos objetos. Apesar de ser um erro bastante significativo, não é de costume da equipe Cedro utilizar a cor rosa como marcadores de robôs em seus jogos, sendo assim, essa taxa de erro não influenciará na eficiência do sistema.

### Roxo

Todos os objetos roxos dispostos no campo foram encontrados pelo intervalo da cor(Figura 4.15).



Figura 4.14: Divisão do campo.



Figura 4.15: Objetos da cor roxo

Dentre os quinze, quatro não apresentaram problema algum e foram completamente detectados, partes B, I, H e N do campo; três apresentaram diminuição de área, partes E, G e J, e oito possuíram diminuição em seu contorno. O detalhamento das quantidades e porcentagens dos objetos encontrados está na Tabela 4.6.

Tipo de Objeto	Quantidade	%
Objetos Completos	4	26,66
Objetos Com Falha de Preenchimento	0	
Objetos Com Diminuição de Contorno	8	53,33
Objetos Com Diminuição de Área	3	20
Objetos Com Falhas Críticas	0	
Objetos Extrapolados	0	

Tabela 4.6: Categorização Dos Objetos

### 4.3.2 Cores Com Problemas

Cores como Vermelho e Laranja possuem o problema de por vezes, devido a interferência externa, se assemelharem a outras. Este problema é observado por todas as equipes participantes das competições do VSSS.

#### Vermelho

A cor rosa, em determinadas luminosidades pode acabar sendo semelhante a cor vermelha, por este motivo, alguns traços dos objetos rosas estão aparecendo dentro do intervalo de calibração(Figura 4.17), mas não configuram um objeto.



Figura 4.16: Divisão do campo.



Figura 4.17: Imagem somente com os objetos dentro do intervalo do valor da cor vermelho

Dos 15 marcadores da cor vermelha, 2 apresentaram falha de preenchimento, parte A e B do campo; 4 apresentaram diminuição de contorno, presentes nas partes E, F, H e M; 1 apresentando falha crítica, parte I do campo, e 8 foram encontrados de forma completa. Dentre as classificações, as que descaracterizam o objeto para detecção são: *Objetos Com Falhas Críticas* e *Objetos Com Falha de Preenchimento* que juntos somam somente 19,99% dos objetos. As quantidades e porcentagens dos demais objetos podem ser vistas na Tabela 4.7.

Tipo de Objeto	Quantidade	%
Objetos Completos	8	53,33
Objetos Com Falha de Preenchimento	2	13,33
Objetos Com Diminuição de Contorno	4	26,66
Objetos Com Diminuição de Área	0	0
Objetos Com Falhas Críticas	1	6,66
Objetos Extrapolados	13	

Tabela 4.7: Categorização Dos Objetos

### Laranja

Devido a um problema muito comum na área de calibração de cores, a cor laranja possui a característica de ser, por muitas vezes, semelhante a vermelha(Figura 4.19). Devido ao fator luminosidade, tanto laranja quanto vermelho, tendem a se tornarem próximas.



Figura 4.18: Divisão do campo.



Figura 4.19: Objetos da cor laranja

Sabendo deste problema, o fato de terem sido encontrados objetos da cor vermelha dentro do intervalo de valores da cor laranja é ignorado, e somente serão levados em consideração os objetos visualmente laranjas. Os quinze objetos da cor laranja foram encontrados com precisão. Todos possuindo seu completo preenchimento e borda. As quantidade e porcentagens do objetos estão disposto na Tabela 4.8.

Tipo de Objeto	Quantidade	%
Objetos Completos	15	100
Objetos Com Falha de Preenchimento	0	
Objetos Com Diminuição de Contorno	0	
Objetos Com Diminuição de Área	0	
Objetos Com Falhas Críticas	0	
Objetos Extrapolados	8	

Tabela 4.8: Categorização Dos Objetos

### 4.3.3 Análise Geral

Estavam disposto pelo campo 105 objetos coloridos, sendo a maioria detectado perfeitamente e alguns com falhas. Na Figura 4.20, pode ser observada a quantidade de objetos que se encontram em cada uma das categorias. Destes, 66,7% dos objetos foram encontrados corretamente, 1,9% foram encontrados com falhas de preenchimento, 20% apresentaram perda de contorno, 6,67% apresentaram diminuição da área e 4,76% apresentaram falhas e faltas que prejudicaram totalmente o objeto.

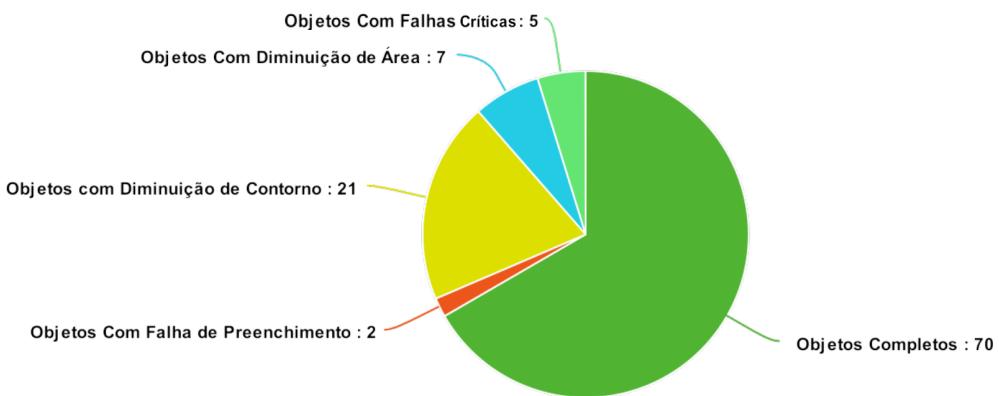


Figura 4.20: Gráfico de análise do resultado dos testes

Analizando-se estes valores, pode-se perceber que os erros considerados críticos na detecção de objetos por meio da sua cor são *Objetos Com Falhas Críticas* e *Objetos Com Falha de Preenchimento* que juntos somam 6,67% dos objetos.

## 4.4 Considerações Finais

Pode-se concluir que o sistema SCIMM, atingiu seus objetivos de calibração automática das cores para utilização no Futebol de Robôs categoria VSSS, pois apresentou ótimos resultado com as cores amarelo, azul, laranja e verde, resultados bons com a cor vermelha, e resultados satisfatório com a cor roxo. Resultados pouco satisfatórios foram obtidos utilizando-se a cor rosa.

As cores calibradas pelo sistema, quando testadas em ambiente de detecção de objetos coloridos, simbolizaram 98 dos 105 objetos, conseguindo com que somente 7 objetos coloridos não tenham sido identificados.

A automatização do processo de calibração levou em média 5 minutos, 1/4 do tempo disponível para ajustes do time durante as competições.

# Capítulo 5

## Conclusão

### 5.1 Principais Considerações

Foi realizado um estudo dos modelos de cores dentro da biblioteca OpenCV e identificado que o mesmo não possuía intervalos definidos no padrão HSV. Percebeu-se que uma simples conversão de valores seguindo a teoria das cores não atenderia a necessidade do sistema, sendo que a biblioteca utiliza pixel no formato GBR e não RGB. Deste modo por meio de experimentação e com a ajuda de fóruns especializados, foi definido espectro para cada cor, categorizando o pixel dentre seus respectivos intervalos.

Outro aspecto identificado durante o desenvolvimento deste trabalho foi quanto os objetos coloridos usados durante a calibração. Como percebido durante a participação da equipe CEDRO no LARC 2015, a maioria das equipes utilizaram o próprios robôs ou quadrados de aproximadamente  $4\text{cm} \times 4\text{cm}$  o que foi identificado neste trabalho que dificultava a detecção e por vezes diminuía o espectro de cores. Por este motivo foi usado tiras de cores com tamanho entre 4 e 10 vezes maior que o utilizado anteriormente. A disposição das tiras no campo também é fundamental no processo de calibração, foi usado uma disposição de 3 tiras por cor, estado elas em cada canto, bem como no meio.

O sistema atingiu satisfatoriamente o seu objetivo principal de diminuir o tempo do processo de calibração, tendo reduzido a duração do processo de calibração para apenas um quarto do tempo de *aquecimento*, bem como a automatização da calibração, pois o sistema encontra sozinho os objetos em campo e os categoriza na sua cor adequada.

A avaliação dos resultados foi separado em **Calibração** e **Teste**. Para ambos foram usadas 7 cores: Amarelo, Azul, Laranja, Rosa, Roxo, Verde e Vermelho. A Calibração durou ao todo aproximadamente 5 minutos e gerou o arquivo *cores.arff* contendo os valores HSV mínimo e máximo para cada cor. Para o Teste foi desenvolvida uma aplicação que faz a exibição das cores de acordo com seus valores no arquivo *cores.arff*. A aplicação mostra uma imagem somente com os objetos da cor selecionada, esta imagem é a que foi usada nos testes do Capítulo 4. Cada objeto de cada cor foi categorizado em: Completo; Com Falha de Preenchimento; Com Diminuição de Contorno; Com Diminuição de Área ou Com Falha Críticas. Das categorias de objetos citadas, as que possuem a área do objeto deformada, o que dificulta sua identificação, são *Falha de Preenchimento* e *Falha Críticas*. Somente três cores apresentaram estas falhas: Amarelo, com 6,66%; Vermelho, com 13,33%; e Rosa,

com 26,66%. De outro modo, somente uma cor não apresentou objetos completos, a cor Rosa. As cores Amarelo, Azul, Verde e Laranja obtiveram mais de 90% de seus objetos encontrados completamente. A cor Vermelha obteve 53,33% de seus objetos encontrados e o Roxo 26,66%. Analisando os resultados obtidos pelo sistema, aconselha-se que seja usadas as cores Vermelho, Verde e Roxo para designar os membros de equipe. Quando a cor da equipe, mesmo que tanto Amarelo quanto Azul tenham obtido valores considerados bons, a cor Azul ainda se sobressaiu à Amarela, sendo então indicada para escolha, ao usar o sistema.

## 5.2 Trabalhos Futuros

Melhorias são indicadas para o sistema tanto na área de detecção de objetos quanto para a categorização de cores. Na detecção de objetos recomenda-se aprimorar a técnica para que não necessitem da subtração de fundo, pois a técnica utilizada neste trabalho infelizmente requer um certo tempo de processamento bem como ação manual, assim sera possível que o sistema se torne 100% automatizado e ainda mais rápido.

Quanto a categorização de cores: a aplicação da técnica também para a cor ciano, bem como melhorar os valos de S e V, uma vez que esses estão sendo usados de maneira estática, seria indicado uma melhoria para que os mesmos sejam os valores detectados das tiras de cor, o que necessita de um certo tratamento.

Outra indicação de incrementação do projeto seria a conversão do sistema para uma biblioteca modular, para que o mesmo possa ser incorporado em diferentes sistemas de estratégia.

# Bibliografia

- ALBUQUERQUE, M.; ALBUQUERQUE, M. Processamento de imagens: métodos e análises. *Revista de Ciência e Tecnologia*, FaCET, Rio de Janeiro, v. 1, n. 1, p. 10–22, 2001. ISSN 1519-8022. Centro Brasileiro de Pesquisas Físicas MCT.
- AMIT, Y.; FELZENSZWALB, P. Object detection. In: IKEUCHI, K. (Ed.). *Computer Vision, A Reference Guide*. New York, NY, USA: Springer, 2014. v. 2, p. 537–542.
- AZEVEDO, E.; CONCI, A. *Computação Gráfica: Geração de Imagens*. 1. ed. Rio de Janeiro/RJ: Elsevier, 2003. ISBN 9878535212525.
- CANNY, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, n. 6, p. 679–698, Nov 1986. ISSN 0162-8828.
- COSTA, A. H. R.; PEGORARO, R. Construindo robôs autônomos para partidas de futebol: o time guaraná. *SBA Controle & Automação*, v. 11, n. 03, p. 141–149, 2000.
- CULJAK, I. et al. A brief introduction to OpenCV. In: *MIPRO, 2012 Proceedings of the 35th International Convention*. [S.l.: s.n.], 2012. p. 1725–1730.
- FARIA, G. *Uma arquitetura de controle inteligente para múltiplos robôs*. Tese (Doutorado) — ICMC/USP, São Carlos/SP, 2006.
- FARIA, G. et al. Multi robot system based on boundary value problems. In: IEEE. *EEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing, China, 2006. p. 2065–2069.
- FIRA. *Brief History*. 2016. Acessado 09/09/2016 22:35. Disponível em: <[http://fira.net/contents/sub01/sub01\\_3.asp](http://fira.net/contents/sub01/sub01_3.asp)>.
- FIRA. *Overview*. 2016. Acessado 09/09/2016 22:36. Disponível em: <[http://fira.net/contents/sub01/sub01\\_2.asp](http://fira.net/contents/sub01/sub01_2.asp)>.
- FREUND, Y.; SCHAPIRA, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In: SPRINGER. *European conference on computational learning theory*. [S.l.], 1995. p. 23–37.
- GARCIA, C.; DELAKIS, M. Convolutional face finder: a neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, p. 1408–1423, 2004.
- GONZALEZ, R.; WOODS, R. *Digital Image Processing*. Pearson/Prentice Hall, 2008. ISBN 9780131687288. Disponível em: <<https://books.google.com.br/books?id=8uGOnjRGEzoC>>.

- HART, P. E.; STORK, D. G.; DUDA, R. O. *Pattern classification*. 2. ed. New York: John Wiley & Sons, 2001.
- HORVATH, M. *Imagens HSV, HSL e RGB*. 2008. Disponível em: <<http://isometricland.net/artwork/artwork.php>>.
- HOUNSLOW, K. *Tutorial: Real-Time Object Tracking Using OpenCV*. 2013. Acessado 09/09/2016 20:11. Disponível em: <<https://www.youtube.com/watch?v=bSeFrPrqZ2A>>.
- HUGHES, J. F. et al. *Computer graphics: principles and practice (3rd ed.)*. Boston, MA, USA: Addison-Wesley Professional, 2013. 1264 p. ISBN 0321399528.
- HYVÄRINEN, A.; KARHUNEN, J.; OJA, E. *Independent component analysis*. [S.l.]: John Wiley & Sons, 2004.
- ITSEEZ. *OpenCV*. 2016. Acessado 29/06/2016 16:53. Disponível em: <<http://docs.opencv.org/3.1.0/>>.
- JOLLIFFE, I. *Principal component analysis*. 2. ed. New York/USA: Springer, 2002. (0172-7397).
- KITANO, H. et al. Robocup: A challenge problem for AI. *AI magazine*, v. 18, n. 1, p. 73–85, 1997.
- LEÃO, A. C.; ARAÚJO, A. de A.; SOUZA, L. A. C. Implementação de sistema de gerenciamento de cores para imagens digitais. In: TEIXEIRA, A. C.; BARRÉRE, E.; ABRÃO, I. C. (Ed.). *Web e multimídia: desafios e soluções*. [S.l.]: PUC Minas, 2005.
- LEE, D. D.; SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature*, Nature Publishing Group, v. 401, n. 6755, p. 788–791, 1999.
- MARTINS, D. L. et al. A versão 2007 da equipe POTI de futebol de robôs. In: *Team Description Paper. Competição Brasileira de Robótica, Florianópolis, Brasil*. [S.l.: s.n.], 2007.
- MENDES, E. P.; MEDEIROS, A. A. D. Sistema de localização visual da equipe de futebol de robôs POTI-UFRN (versao 2008) na categoria very small size. In: *Team Description Paper: Competição Brasileira de Robótica*. Salvador, Brasil: [s.n.], 2008.
- NASCIMENTO, M. C. *Detecção de Objetos em Imagens*. Dissertação (Trabalho de Graduação) — Centro de Informática, Universidade Federal de Pernambuco, 2007. Disponível em: <[www.cin.ufpe.br/~tg/2007-2/mcn2.doc](http://www.cin.ufpe.br/~tg/2007-2/mcn2.doc)>.
- OSUNA, E.; FREUND, R.; GIROSI, F. Training support vector machines: an application to face detection. *IEEE Conference on Computer Vision and Pattern Recognition*, p. 130–136, 1997.
- PAPAGEORGIOU, C.; POGGIO, T. *A training object system: car detection in static images*. 1999. MIT AI Memo.
- PENHARBEL, E. A. et al. Filtro de imagem baseado em matriz rgb de cores-padrão para futebol de robôs. *I Encontro de Robótica Inteligente*, 2004.

- PENHARBEL, E. A. et al. Time de futebol de robôs y04 do centro universitário da FEI. *Jornada de Robótica Inteligente - Encontro Nacional de Robótica Inteligente*, 2004.
- ROSA, J. F. da. *Construção de um time de futebol de robôs para a categoria IEEE Very Small Size Soccer*. Dissertação (Trabalho de Conclusão de Curso) — Faculdade de Educação Tecnológica do Estado do Rio de Janeiro, Petrópolis/RJ, 2015. Disponível em: <<http://docplayer.com.br/11866286-Construcao-de-um-time-de-futebol-de-robos-para-a-categoria-ieee-very-small-size-soccer.html>>.
- ROWLEY, H.; BALUJA, S.; KANADE, T. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, p. 29–38, 1998.
- SAINI, S.; KASLIWAL, B.; BHATIA, S. *Comparative study of image edge detection algorithms*. 2013. Disponível em: <<https://arxiv.org/pdf/1311.4963.pdf>>.
- SOUTO, R. P. *Segmentação de imagem multiespectral utilizando-se o atributo matiz*. Dissertação (Dissertação de Mestrado) — INPE, São José dos Campos, 2003.
- STROUSTRUP, B. A history of C++: 1979–1991. In: BERGIN, T. J.; GIBSON, R. G. (Ed.). *History of programming languages — II*. [S.l.]: Addison-Wesley, 1996.
- SUZUKI, S. et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, Elsevier, v. 30, n. 1, p. 32–46, 1985.
- TIENE, P. E. S. *Implementação de Melhorias no Time de Futebol de Robôs Ararabots Azul – Categoria IEEE Very Small Size Soccer (VSSS)*. Campo Grande/MS, 2014.
- VALE, G. M. do; POZ, A. P. D. O processo de detecção de bordas de canny: Fundamentos, algoritmos e avaliação experimental. In: BERGIN, T. J.; GIBSON, R. G. (Ed.). *Simpósio Brasileiro de Geomática*. [S.l.: s.n.], 2002. p. 292–303.
- VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society, Conference on*. [S.l.], 2001. v. 1, p. ?–511.
- WANGENHEIM, A. v. *Encontrando a Linha Divisória: Detecção de Borda*. 2014. 9-24 p. Disponível em: <<https://www.yumpu.com/pt/document/view/12577417/encontrando-a-linha-divisoria-deteccao-de-bordas-ufsc>>.
- ZIOU, D.; TABBONE, S. Edge detection techniques—an overview. *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, NAUKA/INTERPERIODICA PUBLISHING, v. 8, p. 537–559, 1998.