

# Trabalho de Conclusão de Curso

Desenvolvimento de um Sistema de  
Calibração de Cores com  
Reconhecimento de Objetos para a  
Equipe de Futebol De Robôs Cedro  
Categoria IEEE Very Small Size

Jasane Schio

Orientação: Prof. Dr. Gedson Faria

Área de Concentração: Sistemas de Informação, Visão Computacional

Sistema de Informação  
Universidade Federal de Mato Grosso do Sul  
30 de Setembro de 2015

Desenvolvimento de um Sistema de  
Calibração de Cores com  
Reconhecimento de Objetos para a  
Equipe de Futebol De Robôs Cedro  
Categoria IEEE Very Small Size

Coxim, 01 de Outubro de 2015.

Banca Examinadora:

- Prof. Me. Angelo Darcy (CPCX/UFMS)
- Prof. Dr. Gedson Faria (CPCX/UFMS) - Orientador

# Conteúdo

<b>Lista de Figuras</b>	<b>5</b>
<b>1 Introdução</b>	<b>6</b>
1.1 Justificativa . . . . .	7
1.2 Objetivos . . . . .	7
1.2.1 Objetivo Geral . . . . .	7
1.2.2 Objetivos Específicos . . . . .	7
<b>2 Fundamentação Teórica</b>	<b>8</b>
2.1 ???? . . . . .	8
2.1.1 Detecção de Objetos . . . . .	8
2.1.2 Detecção de Bordas . . . . .	9
2.2 Espaço e Modelo de Cores . . . . .	10
2.2.1 Espaços de Cores . . . . .	10
2.2.2 Modelo de Cores . . . . .	11
2.3 Linguagem de Programação e Bibliotecas . . . . .	14
2.4 Probabilidade e Estatística . . . . .	14
2.4.1 T de Student . . . . .	14
2.4.2 Graus de Liberdade . . . . .	15
2.4.3 Intervalos de Confiança . . . . .	15
<b>3 Metodologia e Desenvolvimento</b>	<b>16</b>
3.1 Projeto . . . . .	16
3.1.1 Organização do Projeto . . . . .	16
3.2 Fluxo do Sistema . . . . .	20
3.2.1 Etapa 1 - Inicialização . . . . .	21

3.2.2	Etapa 2 - Abertura do Programa . . . . .	21
3.2.3	Etapa 3 - Calibração . . . . .	21
3.2.4	Etapa 4 - Armazenamento . . . . .	22
<b>4</b>	<b>Aplicativo</b>	<b>23</b>
4.1	Tela Principal . . . . .	23
4.2	Seleção do Objeto Colorido . . . . .	24
4.3	Geração de Gráfico . . . . .	24
4.4	Seleção de Minimos e Maximos . . . . .	25
4.5	Arquivo Final . . . . .	25
<b>5</b>	<b>Primeiros Resultados - Seleção Manual de Mínimos e Máximos</b>	<b>26</b>
	<b>Referências Bibliográficas</b>	<b>27</b>
	<b>Apêndices</b>	<b>28</b>

# **Lista de Figuras**

2.1	Exemplo dos espaços de cores RGB E CMY. . . . .	11
2.2	Exemplo do Modelo de Cor RGB. Horvath(1) . . . . .	12
2.3	Exemplo do Modelo de Cor HSV(a) e HSL(b). Horvath(1) . . . . .	13
2.4	Exemplo do Modelo de Cor HPG. Mendes (2) . . . . .	13
2.5	Distribuição de Student(t) para valores de v. Spiegel (3) . . . . .	15
3.1	Organização das pastas do projeto . . . . .	17
3.2	Diagrama de Classes do projeto . . . . .	17
3.3	Diagrama de Fluxo . . . . .	20
4.1	Janelas do Sistema: Janela 1 e Janela 2 . . . . .	23
4.2	A seleção do ponto com a cor, é obtido pela interação do usuário. . . . .	24
4.3	Gráfico gerado após seleção dos pontos de cor. . . . .	24
4.4	Gráfico apóis escolhas de mínimos e máximos. . . . .	25
4.5	Arquivo com minimo e máximo dos valores HSV para cada cor. . . . .	25
5.1	Threshold gerado apos seleção manual. . . . .	26

# Capítulo 1

## Introdução

Avanços tecnológicos tentam a cada dia mais dar vida as máquinas aplicando sentidos e percepções que se assemelham as humanas. Dentre estes, Inteligência Artificial, Aprendizado de Máquina, processamento de áudio, processamento de imagem entre outros. Segundo Albuquerque(4) processar uma imagem, da mesma maneira que o nosso sistema visual humano é capaz de fazer é extremamente complexo, realizar as mesmas tarefas com a ajuda de máquinas, exige por antecedência uma compreensão “filosófica” do mundo ou dos conhecimentos humanos. Sem esse conhecimento pré existente por parte da máquina, a interpretação de uma imagem e seu processamento se baseia nas informações contidas na mesma.

Essa obtenção e entendimento das informações contidas na imagem se dá pela Visão Computacional, o ramo encarregado por simular o sistema visual humano. O que é feito de uma maneira única pelo sentido humano é separado em varias tarefas dentro da Visão Computacional com a captura de imagem, seu processamento, aquisição de informações da mesma, processamento dessa informação e aplicação de parâmetros para classificação da informação entre outros. Gonzalez(5) descreve que uma imagem digital é composta por um número finito de elementos, cada um dos quais tem um determinado local e valor, assim o Processamento de Imagem Digital tem como tarefa a retirada de informações dos elementos de uma imagem.

Dentre tipos de processamento de imagens existem, Gonzalez(5) os define como: aplicações de ações primitivas de modificação de imagem, esta caracterizada por seu resultado final ser também uma imagem semelhante a imagem inicial porém modificada (Low-Level-Process), divisão de imagem em regiões e alguns tipos de reconhecimento e classificação de objetos, caracterizada por seu resultado final ser muitas vezes apenas regiões ou informações da imagem inicial (Mid-Level-Process), e o mais “sensorial” de todos que é a analise de objetos usando funções cognitivas associadas a visão computacional, essa usa informações relevantes para o reconhecimento de objetos (Higher-Level-Process).

Neste trabalho propõe-se o a automatização do processo de detecção de objetos desenvolvendo um sistema de calibração de intervalo de Mínimos e Máximos dos valores HSV de cores para ser usado pela equipe de Futebol de Robôs Cedro, categoria Very Small Size. Valores HSV são um dos tipos de valores usados para definir as cores em computação, esses valores são correspondentemente Hue, a cor pura, Saturation, o grau de pureza da cor, e Lightness, que é o luminosidade aplicada. No sistema além da detecção automática estará também disponível a detecção manual de objetos.

## 1.1 Justificativa

- A falta, na equipe, de um sistema de fácil manuseio para detecção de valores HSV
- A falta, na equipe, de um sistema automático de registro do valores HSV mínimos e máximos
- A falta, na equipe, de um sistema que defina mínimos e máximos de forma automática, baseando-se nos objetos escolhidos
- A falta, na equipe, de um sistema autônomo de calibração de intervalo de cores
- Aplicação do sistema proposto na identificação de robôs moveis em times de futebol de robôs.

## 1.2 Objetivos

### 1.2.1 Objetivo Geral

Este trabalho tem por objetivo principal automatizar o sistema de identificação de objetos coloridos em imagens provenientes de uma câmera em tempo real, fazendo a calibração de intervalo de Mínimos e Máximos dos valores HSV. Para alcançar o objetivo principal, foram propostos os seguintes objetivos específicos.

### 1.2.2 Objetivos Específicos

- Implementar uma interface que conte com disposição de informações no estilo gráfico ou histograma de cores para um corte manual de valores visando diminuição da velocidade de detecção;
- Estudo e implementação de um sistema inteligente de calibração de cores e no corte inteligente de valores minimo e máximo das cores
- Testar o sistema proposto para identificação de equipes e participantes do futebol de robô na categoria Very Small Size.

# Capítulo 2

## Fundamentação Teórica

### 2.1 ????

#### 2.1.1 Detecção de Objetos

Antes de descrever os métodos de classificação devemos fazer algumas definições:

- Em cada detecção de objetos são obtidas as informações sobre a imagem, essas são de acordo com o tipo de detecção desejada. Os dados podem conter informações como posição, tamanho, borda, transformação linear, rotação entre outros. Cada detecção em uma imagem é chamada de pose.
- Métodos de detecção de objeto baseado em classes constroem a classe do objeto baseada em um conjunto de treino. O conjunto de treino é composto por múltiplas imagens exemplo do objeto para que seja assim capturado os aspectos do objeto.

A detecção de objetos pode ser considerada uma técnica herdada do reconhecimento de padrões da área de aprendizado de máquina, esta consiste em separar objetos por categorias de acordo com uma ou mais características específicas. Quando essa técnica se junta ao processamento de imagens, onde são acentuadas as características específicas do objeto dentro da imagem para assim este se destacar, tornou-se possível a detecção de objetos em imagens que dentro do campo de visão computacional é uma das áreas que mais obtêm a atenção de pesquisadores. O primeiro Framework de métodos que usam base de dados categorizando uma ou mais características de um objetos para fazer o reconhecimento através de aprendizado foi apresentado em 2001 por Viola e Jones(6). 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, support vector machine para localizar rostos humanos e carros(7), análise de componentes principais, análise independente de componentes, fatoração de matriz não-negativa, análise discriminativa linear, boosting(8), além da classificação binária, onde se considera a detecção do objeto em tamanho fixo apenas variando na posição na imagem(9).

Em 2005 Ulusoy e Bishop(10) mostraram o quanto útil seria categorizar os métodos de detecção de imagens, e os dividiram em duas principais categorias: generativa e discriminativa.

Categorias que foram aceitas e utilizadas como mostram Amit e Felzenszwalb(9) e Roth e Winter(8).

O método generativo pode ser descrito como um modelo probabilístico para a variância da pose de um objeto juntando com o modelo de aparência, ou seja, um modelo de probabilidade para a aparência da imagem condicional em uma determinada pose, juntamento com um modelo de fundo. Os parâmetros do modelo são estimados a partir de dados retirados de treinamento e as decisões são baseadas nas probabilidades anteriores(9). Em resumo o método generativo tenta encontrar uma representação adequada dos dados originais através da aproximação dos dados originais, mas mantendo o máximo de informação possível(8).

Já o modelo discriminativo tipicamente constrói um classificador que pode discriminar entre imagens (ou sub-imagens) contendo o objeto e as que não contém o objeto. Os parâmetros do classificador são selecionados para minimizar os erros nos dados de treino(9).

Segundo Ulusoy e Bishop(10) o método generativo se destaca por tratar perda de dados ou dados parcialmente rotulados, pela facilidade em que uma nova classe pode ser incrementada na classificação condicional de densidade, independentemente das classes anteriores, e por conseguir facilmente lidar com composição de objetos (ex: óculos, chapéus...), considerando que os modelos discriminativos precisam analisar todas as combinações durante o treinamento. Amit e Felzenszwalb(9) ainda aponta que as vantagens descritas sobre o método discriminativo são ditas como a flexibilidade do modelo que pode ser utilizado em regiões do espaço de entrada onde as probabilidades posteriores diferem significativamente de 0 ou 1, ao passo que as abordagens detalhes generativas modelo de distribuição de X, que podem ser irrelevantes para determinar as probabilidades posteriores, além de ser tipicamente muito rápido em fazer previsões para os novos pontos (teste) de dados, enquanto os modelos generativos muitas vezes exigem solução iterativa, e pela igualdade de circunstâncias, seria de esperar que os métodos discriminativos tenham melhor desempenho preditivo, uma vez que são treinados para prever o rótulo de classe em vez de a distribuição conjunta de vetores e alvos de entrada.

### 2.1.2 Detecção de Bordas

Para um objeto poder ser detectado por algum método de detecção de objetos a imagem passa por um processo de segmentação. A segmentação pode ser dita como o processo de divisão da imagem em objetos(5). De acordo com Wangenheim(11) o processo de segmentação se baseia em dois conceitos: similaridade e descontinuidade. A descontinuidade é o processo onde se separa o fundo das partículas e estas umas das outras, através de linhas, bordas ou pontos. Já a similaridade é o processo onde 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(12) o processo de detecção de bordas é um processo simplificado que serve para diminuir drasticamente o total de dados a serem processados e ao mesmo que o mesmo preserva informações valiosas sobre os objetos. É muito comum a ocorrência de ruídos quando se trata da detecção de bordas, e por sua vez para evitar esses ruídos é necessário a suavização da imagem antes de fazer a detecção. Vale(13) lembra que a suavização possui pontos negativos como perda de informação e deslocamento de estruturas de feições proeminentes no plano da imagem. Além disso, existem diferenças entre as propriedades dos operadores diferenciais comumente utilizados, o que ocasiona bordas diferentes. Assim, como dito por Ziou e Tabbone citados por Vale(13), 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(12) para essa detecção que devem ser levados em consideração, Taxa de Erro e Localização(13).

**Taxa de Erro** É importante que as bordas contidas na imagem não sejam confundidas ou perdidas e ainda 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.(11, 12, 13)

**Localização** A distância entre os pixels de borda encontradas pelo algoritmo e a borda atual deveriam ser o menor possível.(11)

Ao tentar aplicar esses dois critérios para desenvolver um modelo matemático para detecção de bordas sem a necessidade de base em regras preestabelecidas em seu artigo *A Computational Approach to Edge Detection* Canny percebeu que somente esses dois critérios não eram o suficiente para obter uma boa precisão da detecção de bordas. E então 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 exista um único pixel. (11, 12, 13)

Com o acréscimo do terceiro critério então nota-se que o processo de detecção de bordas de Canny mostrou-se bastante flexível, independente da origem da imagem utilizada(13).

## 2.2 Espaço e Modelo de Cores

O olho humano é capaz de identificar cores mesmo com as mais diferentes interferências, luminosidade, tonalidade, intensidade, entre outras ações de agentes externos, pois nosso cérebro assimila a cor a sua aparência, já para uma máquina cores são números, códigos, cada cor contém um código específico e cada uma de suas variâncias e alterações também. Para o nosso cérebro é muito fácil entender, exemplo, que o verde, verde lima, verde escuro são a mesma cor, apenas com tonalidades diferentes, já para o computador estas são: (0,255,0),(50,205,50),(0,128,0), no padrão de cor RGB que considera uma luz visível. Mas se for aplicado luminosidade nessas cores, por exemplo, elas ainda se tornam outras diferentes cores, um código diferente para cada luminosidade possível. Para conseguir cobrir todas essas alterações nas cores foram definidos em 1921 começaram então pela Comissão Internacional de Iluminação (CEI) a serem definidos espaços e modelos de cores(14).

### 2.2.1 Espaços de Cores

Segundo Foley et. al citado por Souto(14) espaço de cores é um sistema tridimensional de coordenadas, onde cada eixo refere-se a uma cor primária. A quantidade de cor primária necessária para reproduzir uma determinada cor, é atribuída a um valor sobre o eixo correspondente. 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. 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, sem sucesso. Para gerar a cor branca é necessário a soma das três cores primárias azul, verde e vermelho. Após anos de estudo entendeu-se que existe duas formas de se obter cores: através da emissão ou reflexão de luz, espaços RGB e CMY respectivamente, Figura 2.2.

## RGB

O espaço de cores que emitem luz é conhecido como RGB que é baseado na teoria das cores primárias vermelho, verde e azul, em inglês Red, Green, Blue, para simular a tricromática visão humana, onde toda cor é composta pela soma das três cores.

## CMY

O espaço de cores que refletem luz é conhecido como CMY que são as cores ciano, magenta e amarelo, em inglês Cyan, Magenta e Yellow. Neste espaço a cor é obtida pela subtração das cores. Existe uma variação ao CMY chamada de CMYK onde é acrescentada a cor preta e foi criado como uma opção mais barata, pois não necessita de pigmentos puros(15).

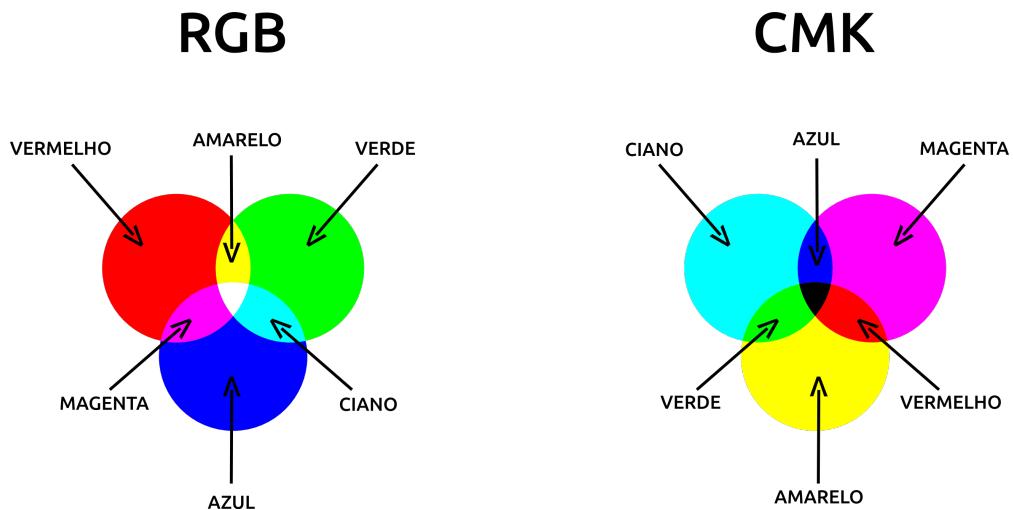


Figura 2.1: Exemplo dos espaços de cores RGB E CMY.

### 2.2.2 Modelo de Cores

Modelos 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.

## RGB

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. Ele não utiliza de nenhum atributo como luminosidade ou tonalidade, por exemplo, para a definição da cor apenas a adição das cores primárias, azul, verde e vermelho. É este também o padrão mais usado e conhecido. Os valores de R,G e B variam de 0 à 255.

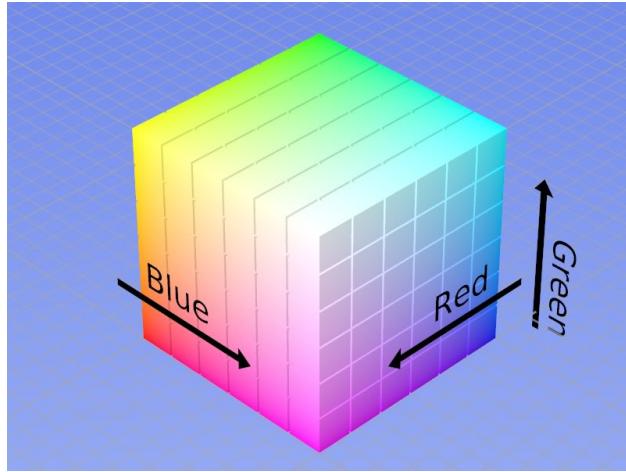


Figura 2.2: Exemplo do Modelo de Cor RGB. Horvath(1)

## HSL, HSB/HSV

Os modelos de cores tem função definir as cores nos programas gráficos de computadores de forma que combine com a percepção das cores pelo sistema visual humano e utilize três eixos similares para definirem a cor(16). O modelo HSL define tonalidade (hue) que é a cor em si, variando de 0 a  $360^\circ$ , saturação (saturation) que define o grau de pureza da cor, obtido pela mistura da tonalidade com a cor cinza, variando de 0 a 1, e luminosidade (lightness) é o brilho de um determinado objeto tendo o branco absoluto com referência. A luminosidade varia de escuro a claro tendo como limites definidos o preto e o branco(16), variando também de 0 a 1. O modelo HSV/HSB define tonalidade (hue) que é a cor em si, variando de 0 a  $360^\circ$ , saturação(saturation) que define o grau de pureza da cor, variando de 0 a 1, obtido pela mistura da tonalidade com a cor branca e brilho (value/brightness) que tenta fazer referência à percepção humana(16) que é a intensidade da cor, variando também de 0 a 1.

## HPG

O modelo de cores HPG foi proposto em 2007 pela Universidade Federal do Rio Grande do Norte estes partiram do princípio de que as cores podem ser definidas com uma mistura de cor pura e tom de cinza(17) e este é apropriado para aplicações onde seja necessário distinguir entre regiões de cor e regiões de cinza(2). O modelo define tonalidade (hue), pureza (purity) e cinzeamento (grayness). Neste modelo um pixel é definido como sendo composto por uma componente de cor pura e por uma componente de tom de cinza puro, ponderados por um

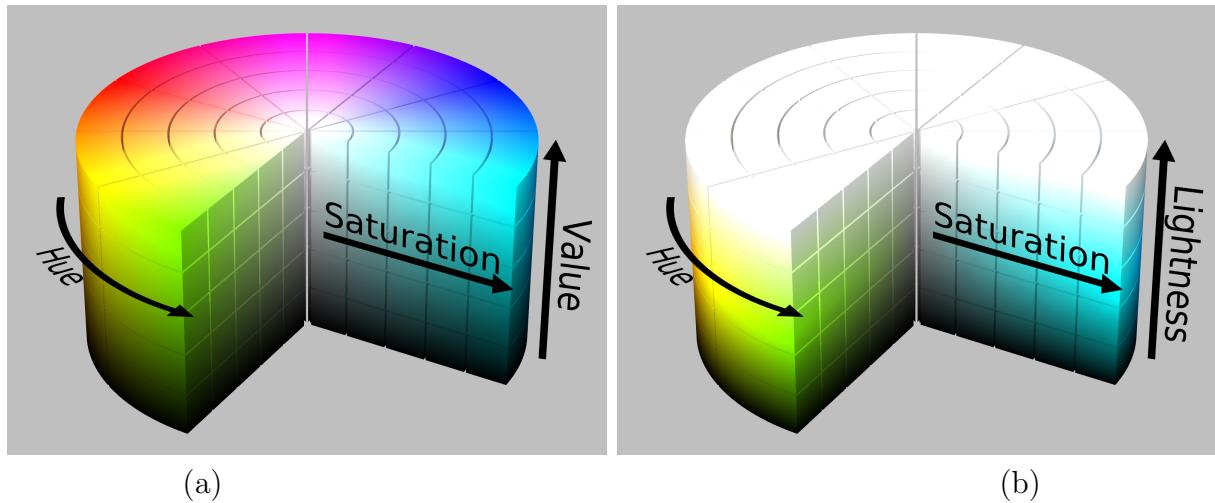


Figura 2.3: Exemplo do Modelo de Cor HSV(a) e HSL(b). Horvath(1)

fator de pureza(2). O modelo de cores HPG se baseia nos valores obtidos de cada pixel no modelo RGB e então é feito um calculo de conversão.

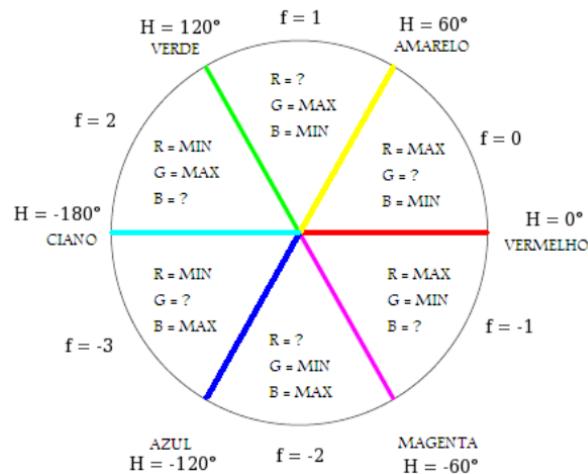


Figura 2.4: Exemplo do Modelo de Cor HPG. Mendes (2)

## 2.3 Linguagem de Programação e Bibliotecas

Para realização deste trabalho, irei utilizar a biblioteca de processamentos de imagens conhecida como OpenCV: Open Source Computer Vision Library. O trabalho será elaborado na linguagem C++, com uso do framework Qt para sua interface gráfica. Os passos detalhados do projeto e seu desenvolvimento estarão presente no Capítulo de Metodologia.

**OpenCV** Lançado em 1999 pela Intel(18), com objetivo de ser otimizada, portável e com um grande número de funções, o Open Source Computer Vision Library, OpenCV, se tornou uma ferramenta que possui mais de 2500 algoritmos e 40 mil pessoas em seu grupo de usuários(18). Já possui interface para as linguagens C++, C, Python e Java além de suporte para as principais plataformas com Windows, Linux, Mac OS, iOS e Android. A biblioteca lida tanto com imagens em tempo real, como vídeos e imagens estáticas.

**Qt** 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++** A linguagem de programação C++ foi projetado por Bjarne Stroustrup para fornecer eficiência e flexibilidade da linguagem C para programação de sistemas. A linguagem evoluiu a partir de uma versão anterior chamado C com Classes, o projeto C com Classes durou entre 1979 e 1983 e determinou os moldes para o C++. A linguagem foi oficialmente lancada em 1986.(19)

## 2.4 Probabilidade e Estatística

Para a automatização dos sistema de detecção de bordas e objetos foi utilizado a Distribuição T de Student para encontrar o valor de referência  $t$  para encontrar a Tabela T(20) para assim calcular o valor do Intervalo de Confiança, usado para limitar o tamanho desejado dos objetos. Todas as definições dessa seção foram retidas do livro Estatística de Spiegel(3).

### 2.4.1 T de Student

Com a necessidade de manipular dados de pequenas amostras William Sealey Gosset com o pseudônimo de Student derivou o teste t de Student baseado na distribuição de probabilidades t, publicando esses estudos em 1908 na revista Biometrika(21). A teoria T de Student é um teoria usada em pequenas amostras, ou seja, amostras com tamanho menor que 30.

De acordo com Spiegel (3), a definição da distribuição de "Student" t é dada por:

$$t = \frac{\bar{X} - \mu}{s/\sqrt{N}} = \frac{\bar{X} - \mu}{\hat{s}/\sqrt{N}}$$

Considerando-se amostras de tamanho N, extraídas de uma população normal de média  $\mu$ , e, se para cada amostra cacular-se o valor de  $t$ , por meio da média amostral e  $\bar{X}$  e do desvio padrão s ou  $\hat{s}$ , pode-se obter a distribuição amostral de  $t$ .

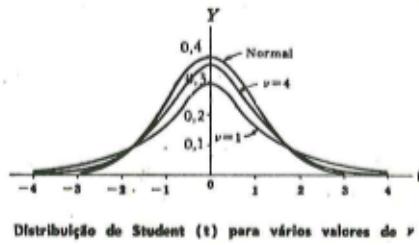


Figura 2.5: Distribuição de Student(t) para valores de v. Spiegel (3)

A distribuição (figura 2.5) é dada por:

$$Y = \frac{Y_0}{\left(1 + \frac{t^2}{N-1}\right)^{N/2}} = \frac{Y_0}{\left(1 + \frac{t^2}{\nu}\right)^{(\nu+1)/2}}$$

em que  $Y_0$  é uma constante que depende de N, de modo que a Área subentendida pela curva é igual a 1, e que a constante  $\nu = (N - 1)$  é denominada *número de graus de liberdade*  $\nu$ .

## 2.4.2 Graus de Liberdade

O numero de graus de liberdade é definido como o número N de observações independente da amostra, menos o numero  $k$  de parâmetros populacionais que devem ser estimados por meio das observações amostrais. Simbolicamente,  $\nu = N - k$ . O numero de graus de liberdade para a Distribuição T de Student é definida pelo número de observações independentes da amostra N, do qual podem ser calculados  $\bar{X}$  e  $s$ . Entretanto, como  $\mu$  deve ser avaliado,  $k = 1$ , então,  $\nu = N-1$ .

## 2.4.3 Intervalos de Confiança

A estimativa de parâmetro dada por dois números é denominada *estimativa por intervalo*, esta estimativa é considerada mais precisa e exata e assim é preferível à outras estimativas. Se a distribuição é aproximadamente normal pode se esperar que se encontre uma estatística amostral real, situada nos intervalos, assim pode-se esperar, ou estar confiante, de que o valor seja encontrado entre os intervalos, por esse motivo, esses intervalos são considerados intervalos de confiança. Para fazer o calculo dos Intervalos de Confiança é necessário escolher o *Nível de Confiança*, dados em percentagem e que ficam, na maioria dos casos, entre 95% e 99%.

Os limites do Intervalo de Confiança para médias, pode ser representado por:

$$\bar{X} \pm t_c \frac{s}{\sqrt{N-1}}$$

onde os valores, dos limites inicial e final,  $t_c$  são denominados *críticos* ou *coeficiente de confiança* e dependem do nível de confiança desejado e do tamanho da amostra. Valores retirados da Tabela T, neste trabalho foi usada para referencia a Tabela T da Universidade Federal Fluminense(20). Para os propósitos finais foi escolhido um nível de confiança de 95

# Capítulo 3

## Metodologia e Desenvolvimento

Para o desenvolvimento foi escolhida a biblioteca OpenCV por ser OpenSource, multi-plataforma, 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, tendo assim grande desempenho e por ser uma linguagem orientada objeto.

O sistema desenvolvido é separado em duas partes: Processamento e Interface Gráfica. A parte de Processamento é onde são feitas as partes de aquisição de imagem, processamento de imagem, conversão de imagem para modelo de cor HSV, seleção de pontos de cor e contagem de ocorrência de cor. Já a interface gráfica, é a onde ocorre a entrada do usuário para assim ser feita a calibração manual de mínimos e máximos de cada cor.

Passos do projeto:

**Aquisição de imagens em vídeo:** Nesse passo as imagens são adquiridas via câmera USB.

**Identificação de Objetos:** Durante o processo de aquisição de imagem são selecionados os objetos, quais serão usados como base para a detecção de máximos e mínimos de cores.

**Cálculo de Mínimos e Máximos:** Nessa etapa são levados em consideração os objetos teste. A imagem é "varrida" por pixel na localidade dos objetos-teste e assim são salvos seus valores e feita a contagem de ocorrências de cada cor.

### 3.1 Projeto

#### 3.1.1 Organização do Projeto

O projeto está sendo desenvolvido na IDE QT Creator e está separada em três pastas. Na pasta Headers estão os arquivos de cabeçalho(.h) onde estão as declarações dos métodos e variáveis usados nas classes executáveis. Já na pasta Sources estão os arquivos fonte(.cpp), são nesses arquivos que os métodos declarados nos arquivos da pasta Header são implementados. Na pasta Forms está o arquivo de interface gráfica(.ui) usada no projeto.

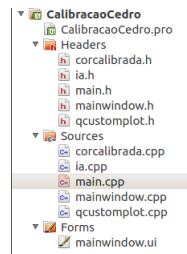


Figura 3.1: Organização das pastas do projeto

A seguir mostrarei o Diagrama de Classes e explicarei os métodos que fazem parte das classes.

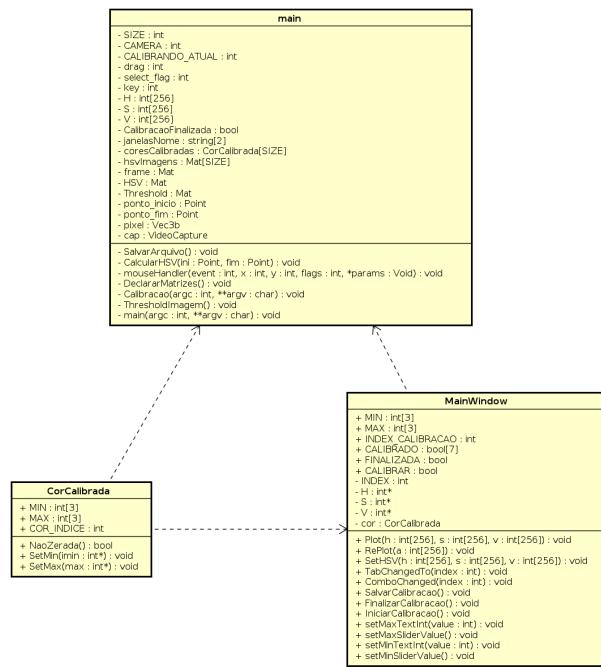


Figura 3.2: Diagrama de Classes do projeto

## Main

Classe principal onde ocorre aquisição de imagem, seleção de cores, contagem de ocorrência HSV, entre outros.

**void SalvarArquivo()** Método sem retorno nem parâmetros, é neste método que o arquivo cores.arff é gravado com os valores da calibração.

**void CalcularHSV(cv::Point ini, cv::Point fim)** Método sem retorno que recebe o ponto inicial e final da seleção da cor. Dentro desse método a imagem é convertida para o modelo de cores HSV e então são feitas as contagens de ocorrência dos valores HSV em cada pixel e adicionado ao seu vetor correspondente.

**void mouseHandler(int event, int x, int y, int flags, void \*param)** Método sem retorno com parâmetros de movimentação do mouse. Esse método segue o modelo especificado pela biblioteca OpenCV. Este método é onde são manipuladas as entradas do usuário via mouse. É onde são identificados a seleção do ponto inicial, feito então uma borda ao redor da seleção, e identificada o ponto final da seleção da cor. É nesse método que é chamado o método CalcularHSV.

**void DeclararMatrizes()** Método sem retorno e sem parâmetros. Neste método os vetores de H, S e V são limpos para então ser feita a calibração de uma outra cor.

**void Calibracao(int argc, char \*\*argv)** Método sem retorno que possui como parâmetros argc, que é um inteiro que possui o total do número de argumentos que a classe foi executada na linha de comando, e argv que são os parâmetros usados na execução do programa na linha de comando, que serão usados para a execução da interface gráfica. Neste método são inicializados a câmera e a interface gráfica, e assim inicializando o processo de calibração e até que a mesma seja finalizada.

**void ThresholdImagem()** Método sem retorno e sem parâmetros onde é mostrado o resultado da calibração. Nele são levados em consideração os mínimos e máximos de cada cor e então feito um Threshold, um método onde são separados os grupos de cinza que formam os objetos da imagem e assim separar somente os objetos desejados.

**int main(int argc, char \*\*argv)** Método sem retorno que possui como parâmetros argc, que é um inteiro que possui o total do número de argumentos que a classe foi executada na linha de comando, e argv que são os parâmetros usados na execução do programa na linha de comando. Neste método é invocado o método Calibração e após a finalização deste método é invocado o método ThresholdImagem(), para mostrar o resultado final.

## MainWindow

Interface gráfica onde são mostrados ao usuário as ocorrências de cada cor e obtidas as interações com o usuário por meio de seleção dos mínimos e máximos de cada um dos valores de HSV.

**void SetHSV(int h[256], int s[256], int v[256])** Método chamado pela Main para passar os valores de ocorrência de HSV para a interface gráfica.

**void Plot(int h[256], int s[256], int v[256])**] O método Plot é chamado pelo metodo SetHSV e faz a inicialização dos gráficos de valores e sua exibição.

**void RePlot(int a[256])**] Apos seleção via Slider ou Entrada de texto, o método RePlot atualiza a exibição do gráfico em questão.

**void TabChangedTo(int index)** Método que lida com a mudança das abas do gráfico.

**void ComboChanged(int index)** Método que lida com a mudança da cor para calibração no ComboBox.

**void IniciarCalibracao()** Método que inicia a calibração.

**void SalvarCalibracao()** Método que salva a calibração da cor atual.

**void FinalizarCalibracao()** Método que finaliza a calibração.

**void setMaxTextInt(int value)** Método que lida com alteração do valor Máximo na área de texto e atualiza tanto o gráfico quando o Slider.

**void setMaxSliderValue()** Método que lida com alteração do valor Máximo no Slider e atualiza tanto o gráfico quando a área de texto.

**void setMinTextInt(int value)** Método que lida com alteração do valor Minimo na área de texto e atualiza tanto o gráfico quando o Slider.

**void setMinSliderValue()** Método que lida com alteração do valor Minimo no Slider e atualiza tanto o gráfico quando a área de texto.

## CorCalibrada

Classe Objeto que guarda valores mínimos e máximos.

## 3.2 Fluxo do Sistema

O sistema pode ser basicamente dividido em 4 etapas. Inicialização, Abertura do Programa, Calibração e Armazenamento. Para um melhor entendimento da execução do programa, seguem abaixo o diagrama de fluxo.

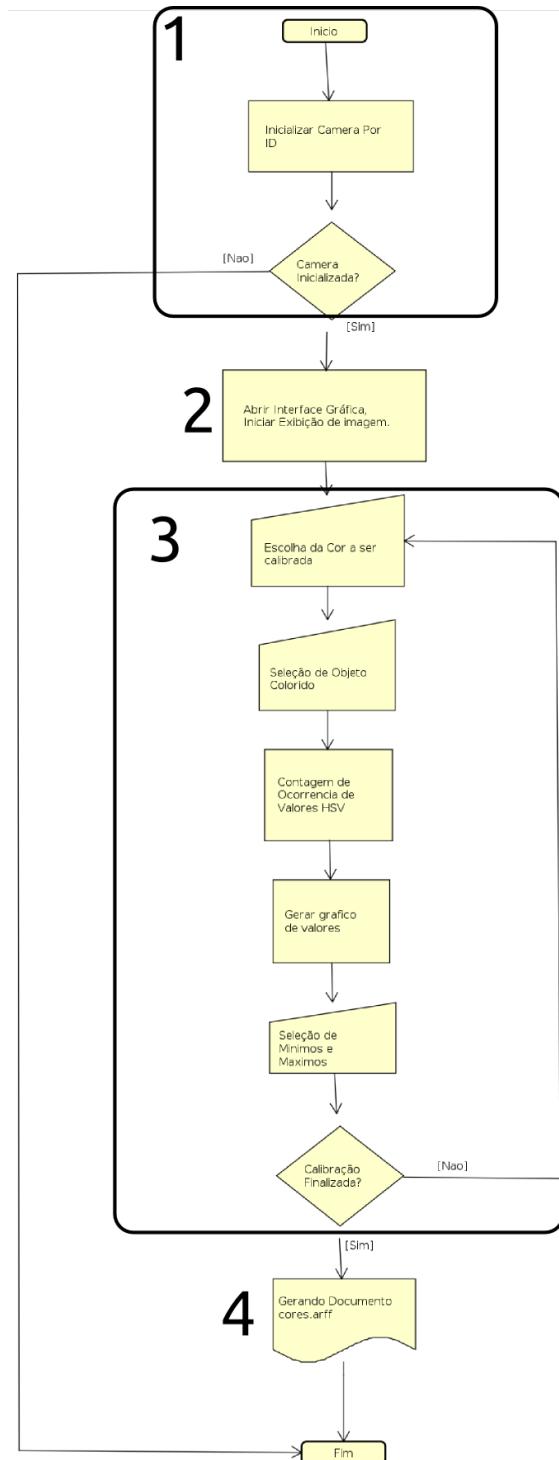


Figura 3.3: Diagrama de Fluxo

### 3.2.1 Etapa 1 - Inicialização

Nesta etapa são declaradas todas as variáveis necessárias para execução do sistema, tanto da interface gráfica quanto da parte de processamento da imagem. Também é nesta etapa que se verifica a câmera esta disponível ou não. Caso a câmera não esteja disponível o sistema não é inicializado.

### 3.2.2 Etapa 2 - Abertura do Programa

Após ter sido inicializado com sucesso o programa dispõe sua interface gráfica que é separada entre 2 janelas, a Janela 1, Gráfico, e a Janela 2, Seleção de Cores, detalhadas no Capítulo 4, sendo então possível iniciar o processo de calibração.

### 3.2.3 Etapa 3 - Calibração

Esta etapa é onde ocorre a maior parte do sistema. O processo de calibração ocorre em um laço finito, até que esteja completa a calibração das cores desejadas e está subdividida em: Escolha da Cor, Seleção do Objeto Colorido, Contagem de Ocorrência HSV, Gerar Gráfico, Seleção de Mínimos e Máximos.

#### Escolha da Cor

Na Janela 1, utilizando o método de entrada disposto é escolhida a cor à ser calibrada. No momento estão disponíveis para calibração as cores: Amarelo, Azul, Ciano, Laranja, Verde, Vermelho, Rosa e Roxo, e seus índices de 0 a 7, correspondentemente.

#### Seleção do Objeto Colorido

Apos a cor escolhida é então necessário que se selecione o ponto que obtenha a mesma na Janela 2. Essa seleção é feita usando pontos inicial e final que sera obtido quando o mouse é clicado na posição inicial e arrastado até a posição final. Apos selecionada a cor é então chamado o método que faz a contagem da ocorrência dos valores HSV em cada pixel.

## Ocorrência HSV

O primeiro passo da calibração é a varredura do(s) ponto(s) selecionado(s) pelo usuário, encontrando os valores HSV de cada pixel, e esses adicionados ao vetor de ocorrência. Quando o método é chamado são passados por referencia os pontos inicial e final da varredura, o frame da imagem é convertido para HSV e então feito sua ocorrência.

---

```
// Método de varredura de pixel e ocorrencia de HSV
cv::cvtColor(frame, hsv, CV_RGB2HSV);
for (int y = ini.y; y < fim.y; ++y) {
    for (int x = ini.x; x < fim.x; x++) {
        pixel = hsv.at<cv::Vec3b>(y, x);
        H[pixel.val[0]]++;
        S[pixel.val[1]]++;
        V[pixel.val[2]]++;
    }
}
```

---

## Gerar Gráfico

Após a seleção da cor, o usuário escolhe calibrar a cor, a geração dos valores só é feita quando o usuário apertar o botão "Calibrar", pois pode ser que a cor a ser calibrada esteja em mais de um ponto da imagem. Quando os pontos com cor são selecionados um gráfico dos valores de H, S e V são gerados e mostrados para usuário.

## Seleção de Mínimos e Máximos

Mínimos e Máximos são selecionados com interação do usuário. Após salvos são armazenados em variais dentro do programa. Quando toda a calibração é terminada o programa finaliza o laço de repetição do processo.

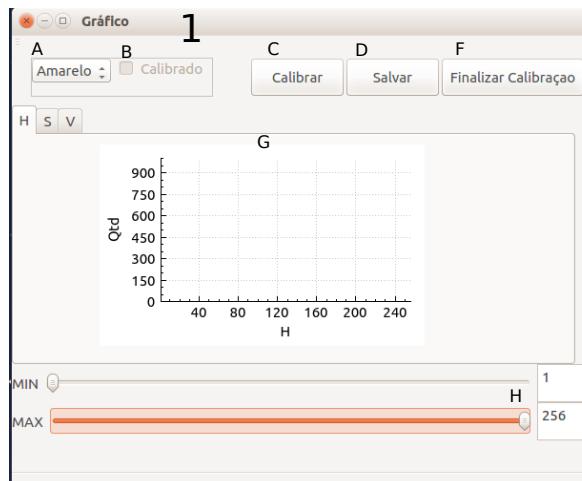
### 3.2.4 Etapa 4 - Armazenamento

Nesta etapa é onde é gerado e armazenado em disco o arquivo cores.arff onde os valores cada cor estão separados em duas linhas, a primeira linha contem os valores mínimos e a segunda os máximos.

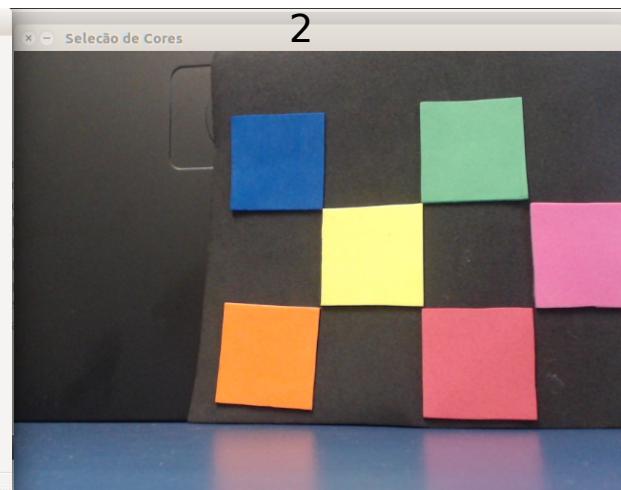
# Capítulo 4

## Aplicativo

### 4.1 Tela Principal



(a) Janela 1



(b) Janela 2

Figura 4.1: Janelas do Sistema: Janela 1 e Janela 2

A tela do sistema é formada por duas janelas. Janela 1 é a janela que recebe as interações do usuário, a Janela 2 é composta pela imagem obtida pela câmera, e para ser feita a escolha da cor. Na Janela 1 possui ações de escolha de cor via Combo Box(A), um Check Box(B) que informa se a cor já foi calibrado ou não. Botões de Calibrar(C), que é usado após a seleção da cor para a geração do gráfico, Salvar(D), que é usado após a seleção dos valores para que estes sejam salvos na memória do aplicativo, e Finalizar Calibração(F) que irá salvar os valores em arquivo. Uma área de gráfico (G) onde são mostrados os valores de ocorrência de H, S e V, cada um em uma aba diferente. A Janela 1 ainda possui uma área de seleção dos valores(H), com região MIN e MAX que possui tanto Sliders quanto Área de texto para seleção dos valores mínimos e máximos.

## 4.2 Seleção do Objeto Colorido

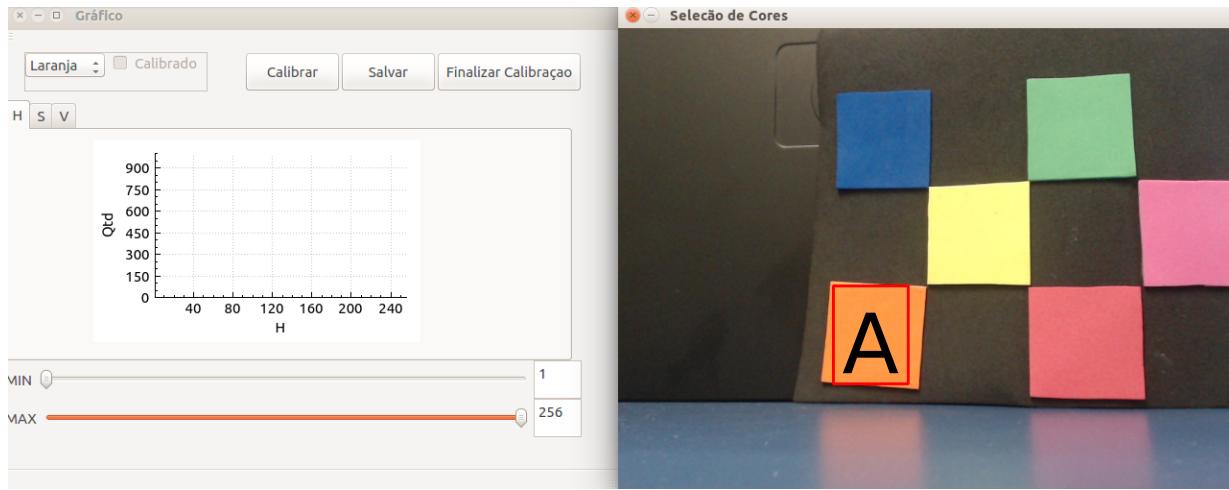


Figura 4.2: A seleção do ponto com a cor, é obtido pela interação do usuário.

A seleção das cores é feita selecionando, com o ponteiro do mouse clicando na área inicial e arrastando até a área final do objeto correspondente a cor. Um exemplo de seleção de cor ocorre na Figura 4.2 em A onde é selecionada a cor Laranja. Após seleção da cor é necessário apertar no botão Calibrar para assim ser feito os cálculos dos valores HSV dos pixels e ser gerado o gráfico.

## 4.3 Geração de Gráfico

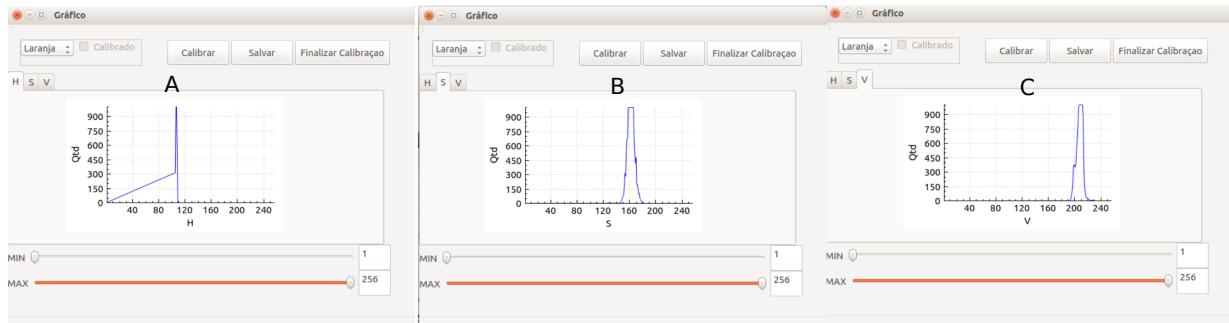


Figura 4.3: Gráfico gerado após seleção dos pontos de cor.

Quando se aperta o botão Calibrar são gerados os 3 Gráficos de ocorrência de valor de H(A), S(B) e V(C), um em cada aba. No gráfico são vistos quais os valores que possuem mais ocorrência, para assim ser feita a seleção.

## 4.4 Seleção de Minimos e Maximos

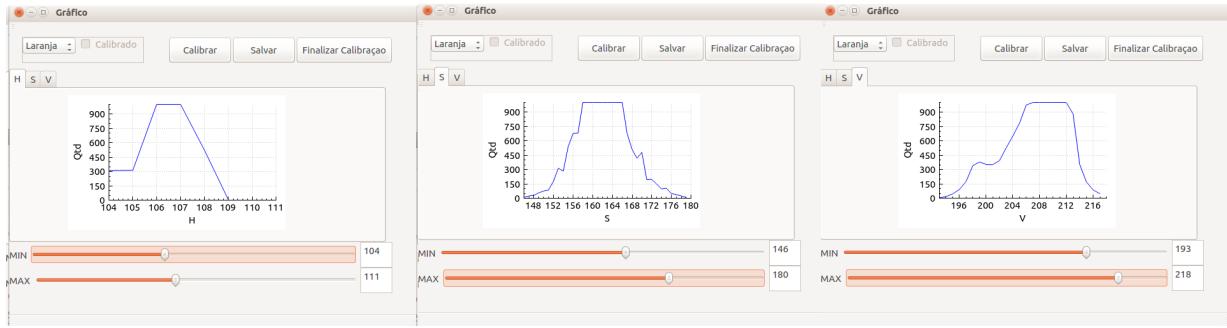


Figura 4.4: Gráfico após escolhas de mínimos e máximos.

Usando os Sliders ou Entrada de Texto(explicados na sessão 4.1) são selecionados os valores mínimos e máximos para H, S e V. Durante a seleção dos valores o gráfico é atualizado para melhor visualização das ocorrências.

## 4.5 Arquivo Final

Arquivo gerado apos valores serem selecionados e salvos. Para cada cor selecionado são feitas duas linhas, a primeira contendo os valores mínimos e a segunda os valores máximos, no inicio de cada linha há o índice da cor.

```
cores.arff (~/Dropbox/TCC/Codigos/QT/build-Tutorial-Desktop_Qt_5_5_1_C
cores.arff x
1 0 : 0.0.0
2 0 : 0.0.0
3 1 : 0.0.0
4 1 : 0.0.0
5 2 : 0.0.0
6 2 : 0.0.0
7 3 : 104.171.251
8 3 : 109.195.255
9 4 : 0.0.0
10 4 : 0.0.0
11 5 : 0.0.0
12 5 : 0.0.0
13 6 : 0.0.0
14 6 : 0.0.0
15 7 : 0.0.0
16 7 : 0.0.0|
```

Figura 4.5: Arquivo com minimo e máximo dos valores HSV para cada cor.

# Capítulo 5

## Primeiros Resultados - Seleção Manual de Mínimos e Máximos

Neste capítulo são apresentados os resultados dos testes realizados.

A primeira etapa do projeto, já concluída, é a seleção manual de mínimos e máximos. Nessa etapa foi gerado uma imagem Threshold onde somente objetos que possuam seu HSV entre os valores calibrados apareçam.

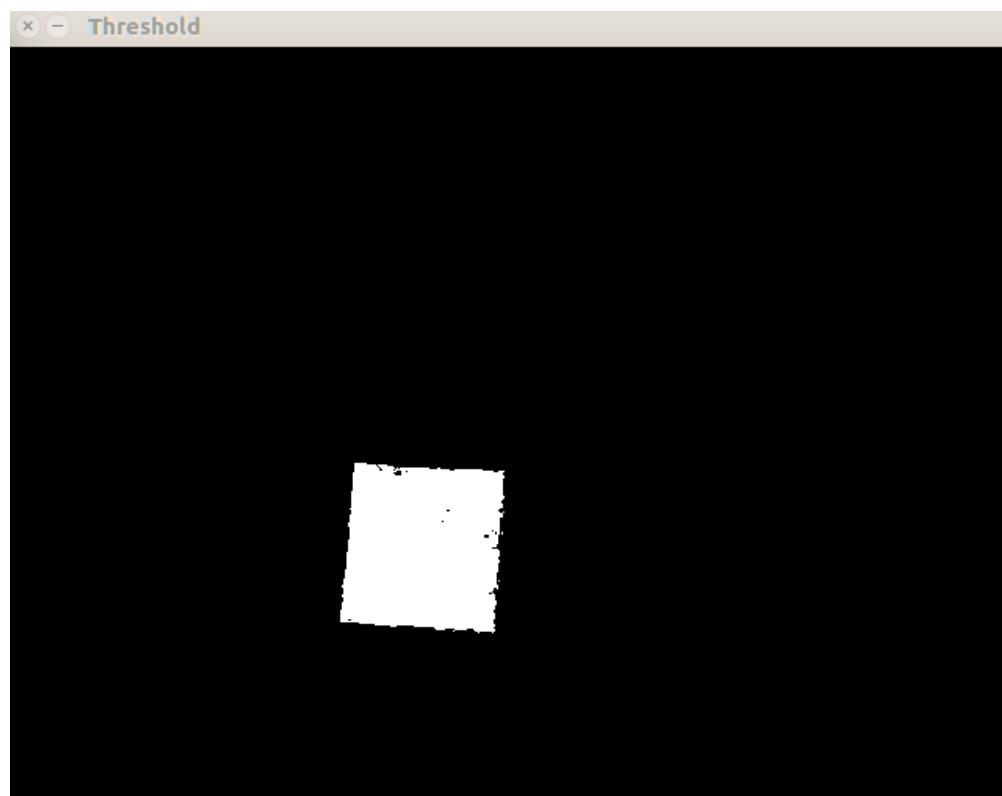


Figura 5.1: Threshold gerado apos seleção manual.

# Bibliografia

- 1 HORVATH, M. In: . [S.l.: s.n.].
- 2 MENDES, E. P.; MEDEIROS, A. A. D. Sistema de localização visual da equipe de futebol de robôs POTI-UFRN (versão 2008) na categoria very small size. In: *Team Description Paper: Competição Brasileira de Robótica*. Salvador, Brasil: [s.n.], 2008.
- 3 SPIEGEL, M. R. *Estatística*. [S.l.]: McGraw-Hill do Brasil, 1974.
- 4 ALBUQUERQUE, M.; ALBUQUERQUE, M. Processamento de imagens: métodos e análises. *Revista de Ciência e Tecnologia*, Rio de Janeiro, p. 10–22, 2001. ISSN 1519-8022. Centro Brasileiro de Pesquisas Físicas MCT.
- 5 GONZALEZ, R.; WOODS, R. *Digital Image Processing*. Pearson/Prentice Hall, 2008. ISBN 9780131687288. Disponível em: <<https://books.google.com.br/books?id=8uGOnjRGEzoC>>.
- 6 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. I–511.
- 7 NASCIMENTO, M. C. *Detecção de Objetos em Imagens*. Dissertação (Mestrado) — Centro de Informática, Universidade Federal de Pernambuco.
- 8 ROTH, P. M.; WINTER, M. Survey of appearance-based methods for object recognition. *Inst. for Computer Graphics and Vision, Graz University of Technology, Austria, Technical Report ICGTR0108 (ICG-TR-01/08)*, 2008.
- 9 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.
- 10 ULUSOY, I.; BISHOP, C. M. Generative versus discriminative methods for object recognition. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Washington, DC, USA: IEEE Computer Society, 2005. (CVPR), p. 258–265. ISBN 0-7695-2372-2. Disponível em: <<http://dx.doi.org/10.1109/CVPR.2005.167>>.
- 11 WANGENHEIM, A. Von. encontrando a linha divisória: Detecção de borda. *Departamento de Informática e Estatística-Universidade Federal de Santa Catarina, 2013a*, v. 16. Disponível em: <[www.inf.ufsc.br/~visao/bordas.pdf](http://www.inf.ufsc.br/~visao/bordas.pdf)>.

- 12 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.
- 13 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.
- 14 SOUTO, R. P. *Segmentação de imagem multiespectral utilizando-se o atributo matiz*. Dissertação (Dissertação de Mestrado) — INPE, São José dos Campos, 2003.
- 15 ROCHA, J. C. Cor luz, cor pigmento e os sistemas rgb e cmy. *Revista Belas, Centro Universitário Belas Artes de São Paulo*, v. 3, 2010. Disponível em: <<http://www.belasartes.br/revistabelasartes/downloads/artigos/3/cor-luz-cor-pigmento-e-os-sistemas-rgb-e-cmy.pdf>>.
- 16 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.
- 17 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.
- 18 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.
- 19 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.
- 20 FLUMINENSE, U. F. *Tabela T: Distribuição de t-Student segundo os graus de liberdade e uma dada probabilidade num teste bicaudal*. Epidemiologia, curso de Medicina, Acessado 01/04/2016 10:00. Disponível em: <<http://www.epi.uff.br/wp-content/uploads/2015/05/Tabela-T.pdf>>.
- 21 NORTE, U. F. do Rio Grande do. *História da Estatística*. Acessado 31/03/2016 14:50. Disponível em: <<http://www.estatistica.ccet.ufrn.br/historia.php>>.