

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

Lista de Figuras	5
1 Introdução	6
1.1 Justificativa	7
1.2 Objetivos	7
1.2.1 Objetivo Geral	7
1.2.2 Objetivos Específicos	7
2 Fundamentação Teórica	8
2.1 Processamento de Imagens	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
2.5 Futebol de Robôs	16
2.6 Trabalhos Relacionados	16
2.6.1 Calibra	16
2.6.2 VSS-Vision	18
3 Metodologia e Desenvolvimento	20

3.1 Projeto	20
3.1.1 Organização do Projeto	20
3.1.2 Classes	22
3.2 O Sistema	22
3.2.1 Calibração Automatica	23
3.2.2 Calibração Manual	28
Referências Bibliográficas	30
Apêndices	32

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
2.6	Sistema Calibra desenvolvido pelo Centro Universitário da FEI (4)	17
2.7	Sistema de calibracao desenvolvido peloSIRLab (5)	18
2.8	Sistema de calibracao desenvolvido peloSIRLab (6)	18
2.9	Nova interface do time da SIRLab (6)	19
2.10	Calibração atual do time da SIRLab (6)	19
3.1	Organizaçao das pastas do projeto	21
3.2	Diagrama de Classes do projeto	21
3.3	Diagrama de Fluxo	22
3.4	Diagrama de Fluxo Automatico	23
3.5	Grafico Exemplificando a Ocorrencia dos valores de H.	27
3.6	Grafico Exemplificando a Ocorrencia dos valores de H após eliminação.	27
3.7	Diagrama de Fluxo Manual	28

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(7) 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(8) 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(8) 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 Processamento de Imagens

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 obtém 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(9). 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(10), análise de componentes principais, análise independente de componentes, fatoração de matriz não-negativa, análise discriminativa linear, boosting(11), além da classificação binária, onde se considera a detecção do objeto em tamanho fixo apenas variando na posição na imagem(12).

Em 2005 Ulusoy e Bishop(13) 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(12) e Roth e Winter(11).

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(12). 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(11).

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(12).

Segundo Ulusoy e Bishop(13) 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(12) 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(8). De acordo com Wangenheim(14) 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(15) 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(16) 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(16), 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(15) para essa detecção que devem ser levados em consideração, Taxa de Erro e Localização(16).

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.(14, 15, 16)

Localização A distância entre os pixels de borda encontradas pelo algoritmo e a borda atual deveriam ser o menor possível.(14)

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. (14, 15, 16)

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(16).

2.2 Cores

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, por sua vez, são sensíveis ao alto nível de iluminação e responsáveis pela percepção de cores(?). E a informação obtida por ambos é assimilada pelo nosso cérebro, ligando a cor a sua aparência, já para uma máquina cores são números, códigos, cada cor contém um código específico e cada uma de suas variações 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(17).

2.2.1 Espaços de Cores

Segundo Foley et. al citado por Souto(17) 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(18).

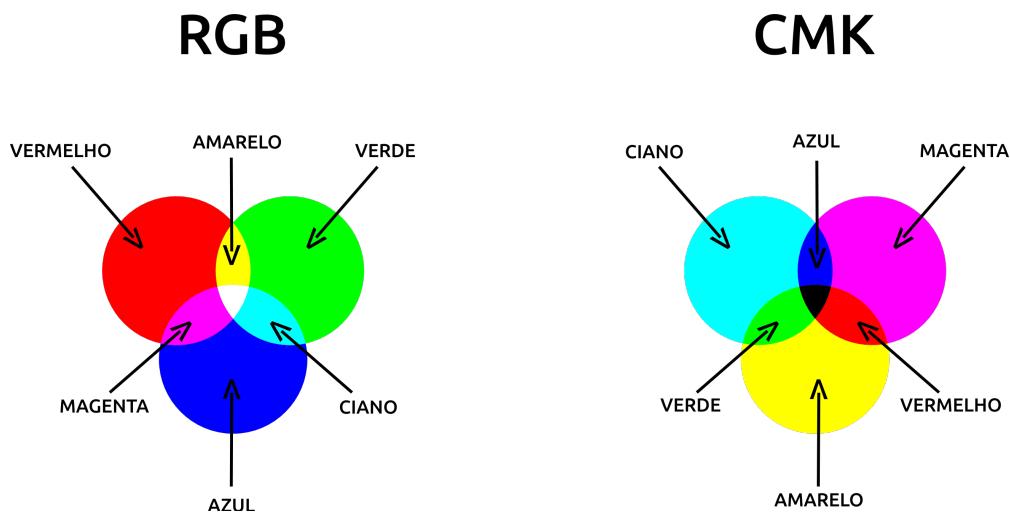


Figura 2.1: Exemplo dos espaços de cores RGB E CMY.

2.2.2 Modelo de Cores

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.

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.

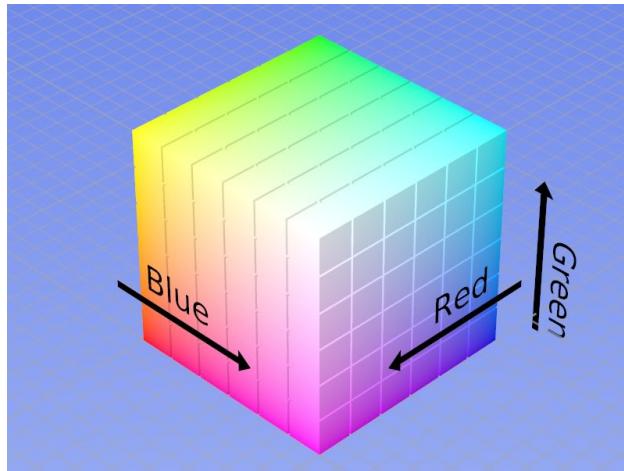


Figura 2.2: Exemplo do Modelo de Cor RGB. Horvath(1)

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(19). O modelo HSL define tonalidade (hue) que é a cor em si, variando de 0 a 360° , 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(19), variando também de 0 a 1. O modelo HSV/HSB define tonalidade (hue) que é a cor em si, variando de 0 a 360° , 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(19) 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(20) 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

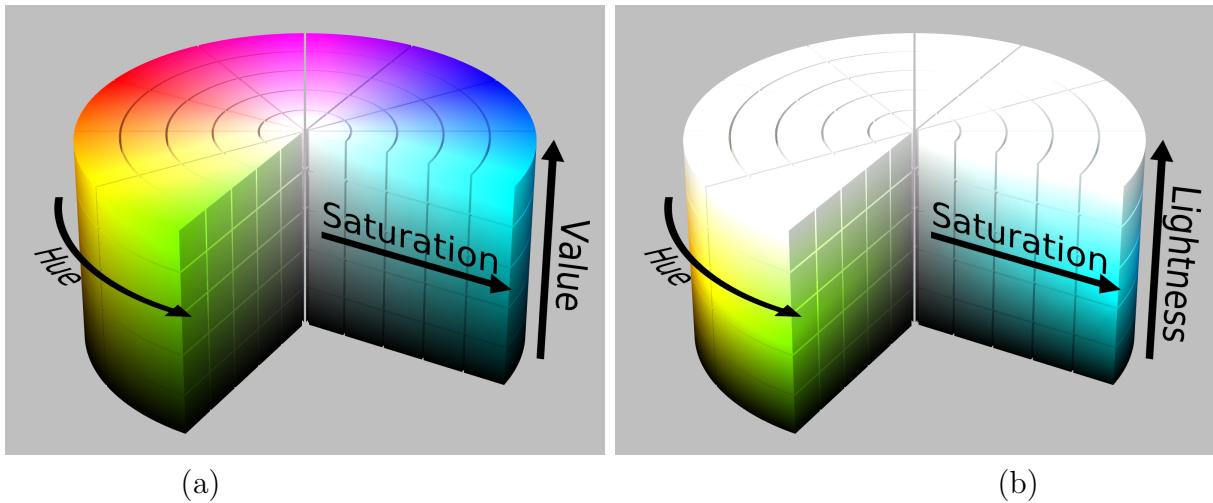


Figura 2.3: Exemplo do Modelo de Cor HSV(a) e HSL(b). Horvath(1)

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 cálculo de conversão.

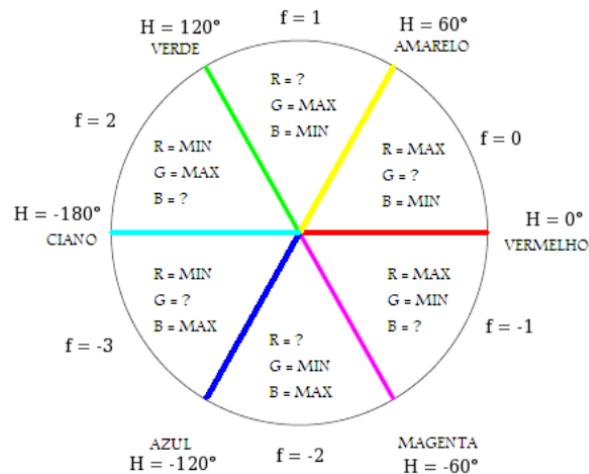


Figura 2.4: Exemplo do Modelo de Cor HPG. Mendes (2)

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(21), 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(21). 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.(22)

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(23) 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(24). 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 μ , 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 .

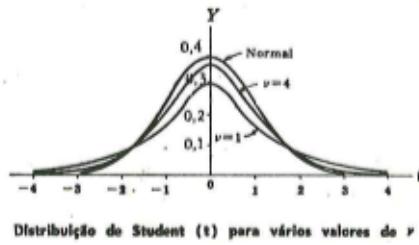


Figura 2.5: Distribuição de Student(t) para valores de v . Spiegel (3)

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 ν* .

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 μ 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 cálculo 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(23). Para os propósitos finais foi escolhido um nível de confiança de 95

2.5 Futebol de Robôs

Visto como um domínio bastante complexo, dinâmico e imprevisível(25), o futebol de robôs surgiu como uma tentativa de promover pesquisas nos campos de Inteligência Artificial e robótica, pela avaliação teórica, algoritmos e arquiteturas através dos problemas padrão(26). A Equipe Cedro se enquadra na categoria IEEE Very Small Size. Esta categoria é regulamentada pelo Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) e possui regras baseadas na MiroSot(5). O futebol de robôs se assemelha ao futebol humano onde o objetivo do jogo é fazer gols para vencer a partida, porém tendo regras adaptadas para o "ambiente" robótico. Rosa(5) em seu trabalho de graduação faz uma boa enumeração das regras básicas:

- 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;
- 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 cada início 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.

2.6 Trabalhos Relacionados

Para o tema específico deste trabalho, calibração de intervalos de cores para times de futebol de robôs da categoria very small size, não foram encontrados trabalhos relacionados, porém foram encontrados Team Description Papers e descrições de sistemas usados pelos times, onde consta sobre o processo de calibração e os métodos usados.

2.6.1 Calibra

O Centro Universitário da FEI, como visto em(27), utiliza em sua equipe Y04 uma solução desenvolvida denominada CALIBRA(4). Desenvolvida para sistemas Linux e com Graphical User Interface(4), o sistema de calibração possui um módulo chamado de MainWindow, que é responsável pela configuração de brilho, cor e contraste da imagem adquirida pela câmera e gera um arquivo que é analisado na hora da criação das cores padrão(27), onde cores-padrão são definidas como intervalos no espaço de cores HSI(27).



Figura 2.6: Sistema Calibra desenvolvido pelo Centro Universitário da FEI (4)

2.6.2 VSS-Vision

Em 2015 Rosa(5) descreve em seu Trabalho de Conclusão sobre a equipe de futebol de robôs Very Small Size, do Laboratório de Sistemas Inteligentes e Robótica, SIRLab(Faeterj-Petrópolis), o sistema de visão computacional da equipe, durante a competição do ano de 2014, que abrange inclusive a parte de calibração.

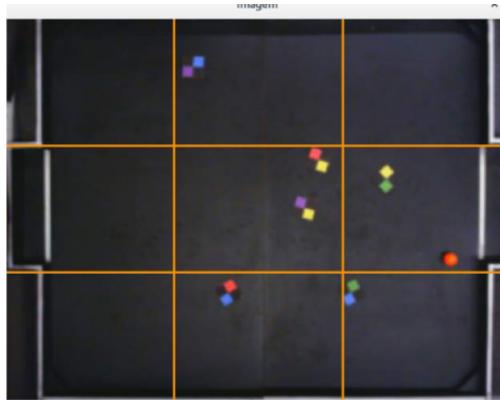


Figura 2.7: Sistema de calibracão desenvolvido peloSIRLab (5)

O autor menciona que a calibração de cores é feita calibrando obrigatoriamente laranja, amarelo e azul, e então as outras cores referentes aos jogadores em campo. Como visto na Figura 2.7 a imagem da camera é dividida em nove cantos, e para calibrar a cor o usuário deve clicar em cima da cor que gostaria de ser calibrada salvando um intervalo de cor tratado como RGB máximo daquela cor e o mínimo, a medida que vão havendo os cliques o sistema verifica para cada atributo 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(5) e esse processo deve ser feito em cada um dos nove cantos da imagem. Os valores HSV encontrados são ajustados manualmente com a ajuda de sliders, como visto na Figura 2.8. Este processo de calibração pode demorar entre cinco e dez minutos. O desenvolvimento do sistema utiliza para processamento de imagens a biblioteca OpenCV e para telas interativas a biblioteca ImGui.

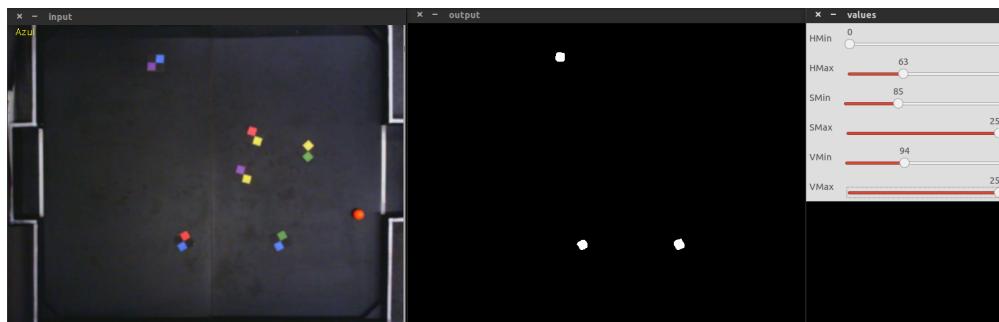


Figura 2.8: Sistema de calibracão desenvolvido peloSIRLab (6)

O atual sistema de visão computacional do SIRLab passou por algumas mudanças desde 2015 e conta com uma interface e método de calibração diferentes(6). Como disponível no repositório online do Laboratório, o atual sistema de calibração de cores utiliza o espaço de cores HSV, no lugar do RGB(5). A antiga interface do sistema, feita inicialmente em ImGui deu lugar à nova, desenvolvida em Qt, como mostrado na Figura 2.9.

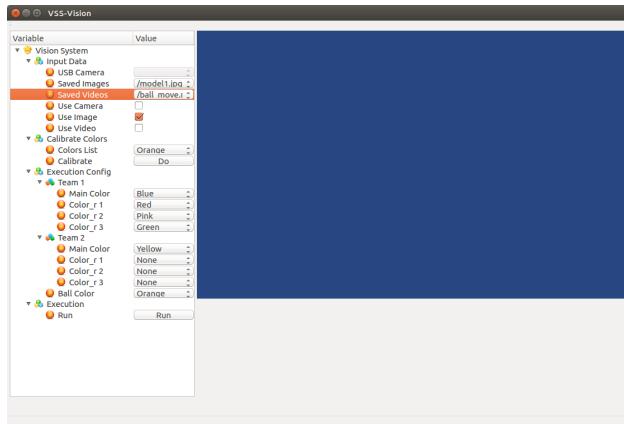


Figura 2.9: Nova interface do time da SIRLab (6)

O metodo de calibração de cores tambem foi modificado, segundo a equipe(6) o sistema possibilita a calibragem de 8 cores, Laranja, Amarelo, Azul, Vermelho, Verde, Rosa, Roxo, Marrom. Após o usuário escolher uma cor para calibrar o mesmo deve encontrar um intervalo de cor, no espaço de cores HSV, que represente-a. Ao clicar na tela com o botão direito o sistema da um zoom na área para ajuste fino. A figura 2.10 demonstra o novo metodo de calibração.

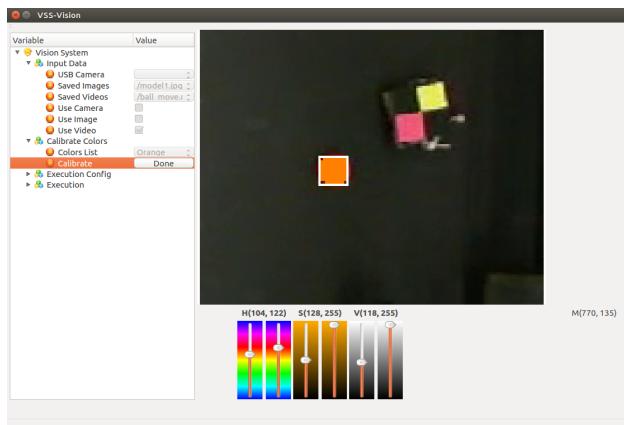


Figura 2.10: Calibração atual do time da SIRLab (6)

Capítulo 3

Metodologia e Desenvolvimento

Para o desenvolvimento foi escolhida a biblioteca OpenCV por ser OpenSource, 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 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 percorrida pixel a 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 foi desenvolvido seguindo o paradigma de programação conhecido como Orientação 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. 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) que é usado no projeto para ser a ponte entre o usuário e as funções do sistema.

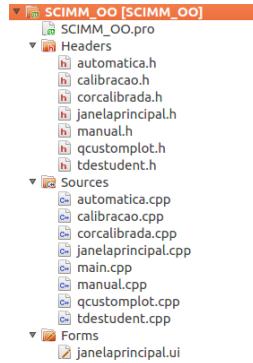


Figura 3.1: Organização das pastas do projeto

Cada arquivo de cabeçalho possui um arquivo fonte correspondente, formando assim uma Classe, com exceção do arquivo fonte main, pois para este arquivo não há a necessidade. As classes desenvolvidas no projeto são: calibracao, manual, automatica, corcalibrada, janela-principal e tstudent. Já a classe qcustomplot é um componente para auxílio em plotagem de gráficos e visualização de dados(28). Para melhor entendimento da interação entre as classes a figura 3.2 trás o diagrama de classes do projeto.

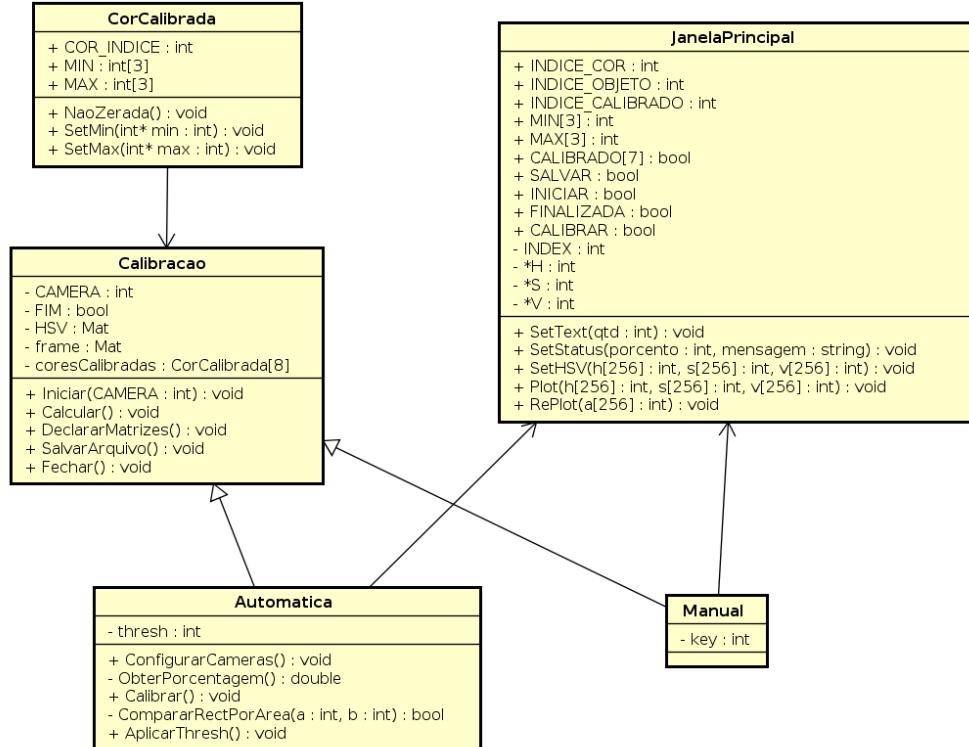


Figura 3.2: Diagrama de Classes do projeto

3.1.2 Classes

main esta é a classe executavel do sistema, ela inicia o programa e em seguida chama a classe de interação grafica **janelaprincipal**

janelaprincipal classe que faz a interação com o usuario e que de acordo com esta interação seleciona o tipo de calibração, e seus parametros, para então ser feita a analise dos pixeis

calibracao classe "pai" que contem os metodos e variaveis que virão a ser usadas por ambas as classes **manual** e **automatica**

manual classe que contem os metodos, cálculo e variaveis necessarias para a calibração manual

automatica classe que contem os metodos, cálculo e variaveis necessarias para a calibração automatica

corcalibrada classe que salva o indice da cor já calibrada e seu intervalo de valores

tdestudent esta é a classe que faz o cálculo probabilistico conhecido com TdeStudent

3.2 O Sistema

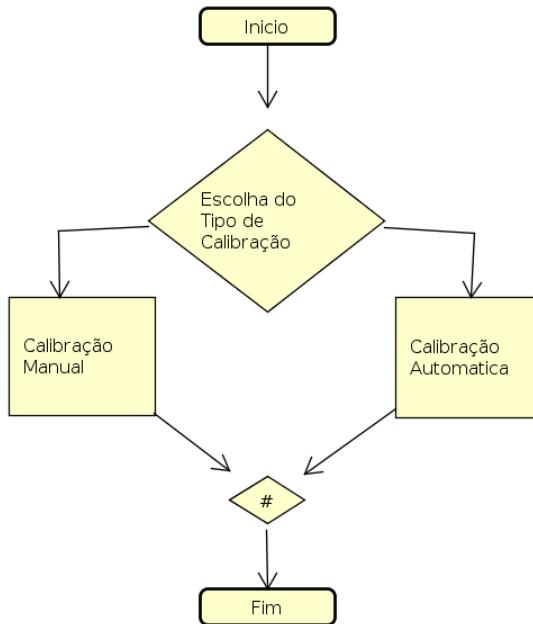


Figura 3.3: Diagrama de Fluxo

A Figura 3.3 mostra o diagrama de fluxo do sistema, este com duas possibilidades: Calibração Manual e Calibração Automatica. Ambas são independentes uma da outra.

O sistema consiste na apresentação da **interface gráfica** ao usuário. A **interface gráfica** que por sua vez oferece as duas possibilidades ao usuário, de acordo com o tipo de calibração escolhido o sistema inicia a rotina de calibração referente. Após a execução de toda o sistema é finalizada.

3.2.1 Calibração Automatica

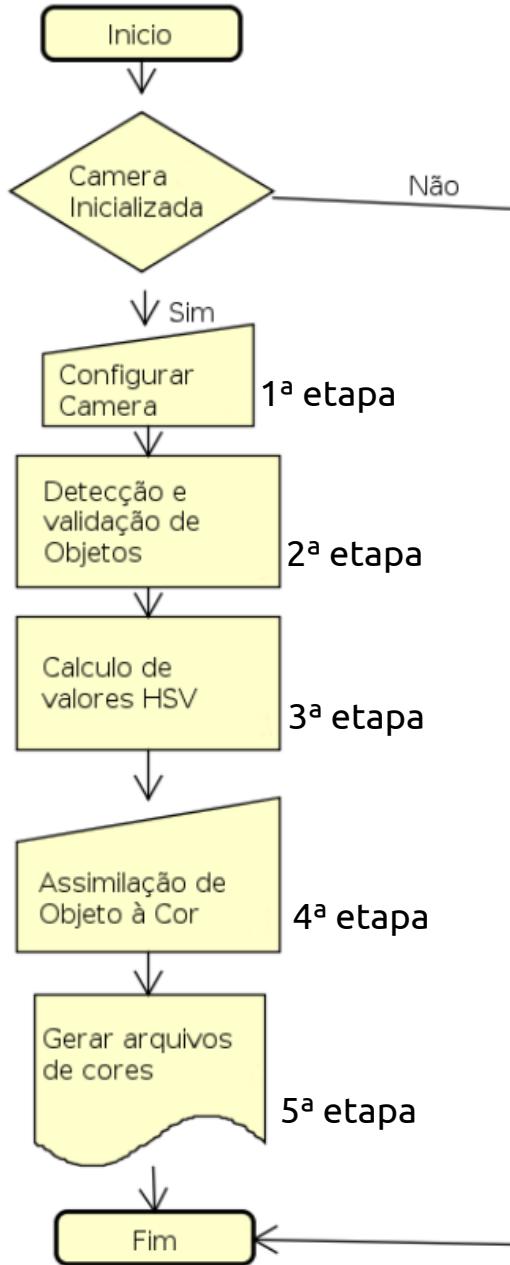


Figura 3.4: Diagrama de Fluxo Automatico

Conforme mostrado na Figura 3.4 a rotina de calibração automatica possui cinco etapas. Este tipo de calibração possui o mínimo possível de interação com o usuário. O sistema faz automaticamente a detecção de objetos e utilizando a probabilidade matemática T de Student, considerando o tamanho de todos os objetos encontrados para encontrar os tamanhos mínimo e máximo que os objetos desejados devem ter, nesse caso as etiquetas de cores dos robôs, e só após determinar quais objetos possuem o tamanho desejado, analisar os valores e calcular as ocorrências de cada um dos três valores do modelo HSV.

Nas sessões seguintes à seguir explicarei detalhadamente cada uma das etapas do processo de calibração automatica

1^a Etapa - Configuração de Camera

Antes de ser feito a calibração propriamente dita são necessarias duas configurações: Ajuste de Constraste e Brilho e Recorte de Imagem. A configuração de contraste e brilho utiliza o metodo *convertTo* da biblioteca *OpenCV* e é utilizada para o melhoramento da imagem antes da detecção dos objetos, a utilização completa fica da seguinte maneira:

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

Esta função recebe quatro parametros. O primeiro **frameA** informa aonde sera salvo o resultado da conversão. O segundo **-1** indica o tipo da matrix, ou numero de canais, da imagem a ser gerada, usa-se **-1** quando se deseja que se use os valores semelhantes aos da imagem da imagem original(29), O terceiro **contrast_value / 50.0** indica o valor de constraste, ou alpha, a ser usado para multiplicar os valores do pixel da imagem(29) e por ultimo **brightness_value** que é o valor do brilho, ou beta, a ser adicionado à imagem. Outra configuração feita é o Recorte de Imagem, onde utilizando a função *setMouseCallback* para possibilitar a interação do usuario na imagem por meio do mouse, sua utilização é dada da seguinte maneira:

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

Tem como primeiro parametro **src_windows** que indica a janela na qual a função recebera a interação, o segundo, **mouseHandler**, indica a função na qual esta implementada a interação e o ultimo parametro, **0**, indica parametros opcionais, neste caso não usaremos nenhum então foi usado o numero 0. Dentro da função **mouseHandler** são identificados os pontos inicio e final da seleção na tela e utilizada a função *rectangle* para demarcar a seleção na tela. A utilização da função *rectangle* completa fica da seguinte maneira:

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

A função rebece os parametros **frameA** indicando a imagem na qual será demarcada a area selecionada, depois o parametro **point1** que é o ponto incial 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 expressura da demarcação, **5** que significa o tipo de linha a ser utilizado na demarcação e **0** que é o numero de bits fracionarios. Após confirmada a escolha do tamanho da tela este é entao salvo na variavel nomeada *tamanho*, está entao sera usado durante todo o processo de calibração.

2^a Etapa - Detecção e validação de Objetos

A detecção dos objetos a serem calibrados é dada pelo algoritmo de detecção de bordas de Canny. Como mais um recurso para eliminação de ruidos e melhoria da imagem antes de ser executado a detecção de objetos através da detecção de bordas é utilizado desfoque na imagem. O algoritmo de Canny já está implementado dentro da biblioteca OpenCV e com a seguinte usagem:

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

O algoritmo de Canny utiliza por padrão imagem em padrões de cinza, sendo assim **src_gray** é a imagem original transformada para escala de cinza, esta é a imagem na qual o algoritmo será aplicado. **canny_output** será a imagem de saída da função. **thresh** e **thresh*3** são os limites mínimos e máximos para considerar uma borda. **3** é o valor de abertura ou kernel, o valor 3 é utilizado como padrão.

Apos o uso do algoritmo de Canny para detecção de bordas é necessário então fazer uso da função *findContours*, nativa no *OpenCV* para detecção de contornos.

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

O primeiro parametro, **canny_output**, é a imagem que o algoritmo de Canny gerou com as bordas encontrada na imagem, e é a imagem que o método *findContours* irá utilizar para detectar os contornos, **contours** é o parametro que indica onde serão salvos os contornos encontrados, cada contorno é armazenado como sendo um vetor de pontos (29). **hierarchy** é onde será salva um vetor de informações sobre a topologia da imagem, e terá como total de elementos o mesmo numero que o total de contornos encontrado(29). O quarto parametro, **CV_RETR_EXTERNAL** indica o modo de obtenção de contornos, nesse caso *CV_RETR_EXTERNAL* indica que o metodo só obterá os contornos exteriores(29). **CV_CHAIN_APPROX_SIMPLE** indica o metodo que sera usado para aproximação de contornos, o metodo *CV_CHAIN_APPROX_SIMPLE* comprime segmentos horizontais, verticais, diagonais e deixa apenas os seus pontos finais(29). E o ultimo parametro, **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 será necessário.

Uma vez obtidos os contornos é necessários que se faça a eliminação de vértices dos polígonos encontrados nos objetos deixando assim o objeto mais preciso. Isso é necessário para deixar a forma encontrada mais precisa da forma original. Para este ajuste foi usado o metodo *approxPolyDP*, já implementado dentro da biblioteca OpenCV. Esse metodo teve que ser aplicado em cada um dos contornos encontrados, e foi utilizado da seguinte maneira:

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

Onde o metodo inicia recebendo como parâmetro, **Mat(contours[i])** que é a criação de uma nova imagem, somente com aquele único objeto, que está sendo analisado. A seguir é informado no segundo parametro a variável de destino **contours_poly[i]**, onde será salvo o objeto com a eliminação dos vértices. O terceiro parametro indica o valor do *epsilon*, usado

o valor **3** que especifica a precisão da aproximação, a distância máxima entre a curva original e a sua aproximação(29). O ultimo parametro indica se a curva aparoximada sera fechada ou não, foi usado o valor **true** pois neste caso fechar um uma curva é necessário para que o objeto onde está a cor, seja idenficado e analisado na probabilidade.

Por ultimo os objetos possuem sua borda ignorada, sendo assim calculado o tamanho interior dele, para que por ventura não hajam pixeis de cor preta ou derivadas a serem calculadas.

A detecção de contorno detecta todos os contornos possíveis na imagem, isso inclue sombras, luzes entre outras coisas. Mas não são todos os objetos encontrados que deverão ser calculados, sendo assim foi usado o cálculo probabilístico T de Student. Foi implementado uma biblioteca para o uso da probabilidade voltada para objetos Rect, os objetos da biblioteca **OpenCV** obtidos na detecção. Essa biblioteca analisa a lista de objetos encontrados na detecção e faz o cálculo dos limites, tamanho mínimo e máximo, dos objetos. Apos a obtenção desse limite pela biblioteca são analisados todos os objetos encontrados na detecção e os cujo tamanho não esteja dentro deste limite são removidos da lista.

3^a Etapa - Calculo de valores HSV

Para que possam ser feitos os cálculo de valores HSV mínimo e máximos é necessário que se faça, primeiramente a conversão da imagem obtida pela camera, normalmente no espaço de cores RGB, para o espaço de cores HSV, pois a mesma lida melhor com diferenças de luminosidade. A biblioteca **OpenCV** converte o espaço de cor usando a função *cvtColor* que utiliza da imagem original, e de uma imagem vazia com memoria alocada para ser salva a imagem apos a conversão, além do parametro do tipo de conversão, Exemplo do uso do método:

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

Apos a conversão é necessário, então, ser feita uma analise dos objetos encontrados. Para cada objeto serão somadas as ocorrencias de cada um dos valores, HSV, e salvos separadamente em um vetor para cada, H, S e V com 256 posições, uma vez que se sabe que os valores de um pixel varial de 0 a 255. As ocorrencias são salvas se baseando no valor HSV do pixel. Para isso o pixel é separado em valores H, S e V este então terá seu valor correspondente a posição no vetor apropriado, por exemplo, se o valor de H em um determinado pixel for de 87, a posição de numero 87 no vetor ganhara uma ocorrencia a mais. No codigo, fica da seguinte maneira:

Algoritmo 1 Contagem dos Valores HSV

```
para todo objeto encontrado faça
    ValoresH[objeto.valorh]++
    ValoresS[objeto.valors]++
    ValoresV[objeto.valorv]++
fim para
```

Uma vez terminada a analise dos pixeis do objeto o sistema procura o valor mínimo e máximo, ou seja a primeira e ultima ocorrencia. Esses valores são encontrados procurando

o primeiro valor de pixel que não esta zerado e o ultimo valor que não esta zerado. Isto é feito para eliminar que durante a analise o numero de valores a serem comparados ao valor real do pixel, da imagem final, seja menor. Para exemplificar a Figura 3.6 tras o gráfico dos valores de H da calibração de um certo objeto antes de ser feia a eliminação dos valores. Se fosse analisado esses valores, não eliminando os valores zerados, o sistema buscara no valor de H do pixel desde o valor 1 ao 126 sem a necessidade, pois se não houve ocorrencia desses valores no objeto de referencia, o que foi analisado, então esses valores não fazem parte da cor desejada.

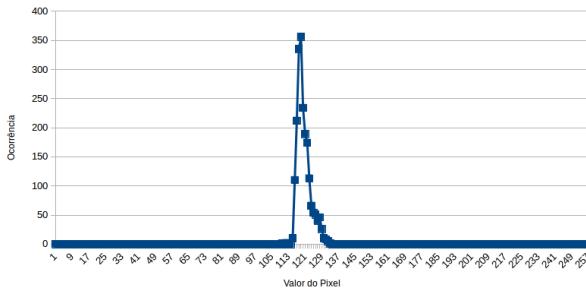


Figura 3.5: Grafico Exemplificando a Ocorrencia dos valores de H.

Apos ser feita a eliminação, Figura 3.7, de valos sabemos que somente a partir do pixel 110 houve ocorrencia, assim quando for necessario analisar pixel a pixel, os valores abaixo de 110 serão ignorados, ou seja não fazem parte da cor desejada.

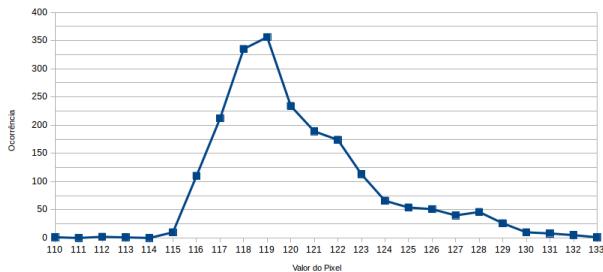


Figura 3.6: Grafico Exemplificando a Ocorrencia dos valores de H apóis eliminação.

Após serem encontrados os valores mínimo e máximos para H, S e V de cada objeto detectado, estes valores são salvos em uma lista de objetos do tipo CorCalibrada. Neste objeto são salvos os respectivos valores.

4^a Etapa - Assimilação de Objeto à Cor

Uma vez que os objetos já foram identificados, suas cores analisadas e gerada sua lista. O sistema, utilizando do recurso de interface gráfica disponivel pela biblioteca Qt, informa ao usuario uma lista com os intervalos de cores encontrados, e outra lista com as possiveis cores a serem assimiladas para eles. Assim o usuario pode analisar um por um dos objetos e escolher qual se assemelha a qual cor. Assim que assimilada a cor ao objeto este é salvo e seu objeto CorCalibrada na lista de objetos salva o valor da cor em seu atributo COR_INDICE.

5^a 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 o indice de cada cor calibrada e seus valores máximos e mínimos.

3.2.2 Calibração Manual

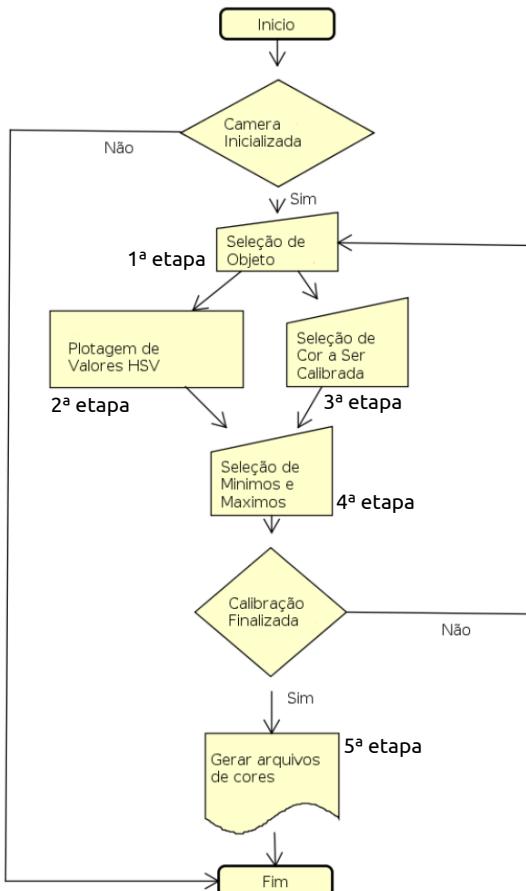


Figura 3.7: Diagrama de Fluxo Manual

A Figura 3.7 mostra o fluxo de calibração manual, o sistema possui cinco etapas que permanecem em repetição até que todos os objetos sejam calibrados. Como o próprio nome já indica, o método de calibração é manual, sendo assim é necessário que o usuário selecione o objeto que o mesmo quer calibrar.

1^a Etapa - Seleção de Objeto

O primeiro passo para ser feita a calibração manual é a seleção de Objeto. Com o auxílio da função `setMouseCallback`, disponível na biblioteca **OpenCV** e já explanada na sessão 3.2.1, é possível obter os pontos de uma seleção feita na imagem, seu uso é necessário, já que obtenção de um ou mais objetos com a **mesma cor** a serem analisados é feita somente com a interação do usuário. Neste seleção existe a possibilidade de escolher somente um objeto ou

mais de um objeto da mesma cor em diferentes pontos do campo, a seleção é feita somente um objeto por vez e cada objeto é analisado e seu valor e ocorrência calculado utilizando o mesmo método encontrado na sessão 3.2.1 subsessão **Calculo de valores HSV**, estes valores são armazenados em memória de execução e somente mostrados ao usuário quando requisitados.

2^a Etapa - Plotagem de Valores HSV

Uma vez que o usuário já selecionou todos os objetos e os mesmos já foram analisados, o processo de calibração encaminha à interface gráfica os valores encontrados e está plotando três gráficos, um para cada um dos valores HSV. Com a ajuda do componente auxiliar qcustomplot os gráficos são mostrados no sistema informando para o usuário quais foram os valores obtidos na seleção.

3^a Etapa - Seleção de Cor a ser Calibrada

A seleção da cor na lista de cores possíveis para ser assimilado, pode ser considerada uma ação concomitante à ação de plotagem de graficos sendo que uma não depende da outra, já que no momento da seleção do objeto o usuário já tinha em mente qual era sua cor, e a plotagem do gráfico não interfere nessa escolha, já a plotagem do gráfico é uma ação somente visual e não tem influencia da cor escolhida.

4^a Etapa - Seleção de Minimos e Maximos

Assim que os graficos são plotados, se torna então possível, utilizando o recurso Slider, de fazer a seleção dos valores, tanto mínimo, quanto máximo. Ao ser modificado, cada um dos valores, o gráfico se ajusta para melhor detalhar ao usuário a ocorrência dos pixels. Ao contrário do sistema automático, no sistema manual o objeto CorCalibrada só é criado após a seleção da cor e dos intervalos pelo usuário. Quando este salva a cor na interface gráfica, a mesma cor é encaminhada para o sistema manual e assim salva na lista.

5^a Etapa - Gerar Arquivo de Cores

Do mesmo modo que ocorre na calibração automática, uma vez que todas as cores já estiverem sido assimiladas pelo usuário, este então finaliza a calibração e um arquivo com os valores calibrados é salvo localmente.

Bibliografia

- 1 HORVATH, M.
- 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 PENHARBEL, E. A. et al. Filtro de imagem baseado em matriz rgb de cores-padrão para futebol de robôs. *Submetido ao I Encontro de Robótica Inteligente*, 2004.
- 5 ROSA, J. F. da. *Construção de um time de futebol de robôs para a categoria IEEE Very Small Size Soccer*. Dissertação (Mestrado) — FACULDADE DEE DUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO.
- 6 SIRLAB. In: . [S.l.: s.n.].
- 7 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.
- 8 GONZALEZ, R.; WOODS, R. *Digital Image Processing*. Pearson/Prentice Hall, 2008. ISBN 9780131687288. Disponível em: <<https://books.google.com.br/books?id=8uGOnjRGEzoC>>.
- 9 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.
- 10 NASCIMENTO, M. C. *Detecção de Objetos em Imagens*. Dissertação (Mestrado) — Centro de Informática, Universidade Federal de Pernambuco.
- 11 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.
- 12 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.
- 13 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>>.
- 14 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>.
- 15 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.
- 16 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.
- 17 SOUTO, R. P. *Segmentação de imagem multiespectral utilizando-se o atributo matiz*. Dissertação (Dissertação de Mestrado) — INPE, São José dos Campos, 2003.
- 18 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>>.
- 19 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.
- 20 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.
- 21 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.
- 22 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.
- 23 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>>.
- 24 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>>.
- 25 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.
- 26 KITANO, H. et al. Robocup: A challenge problem for ai. *AI magazine*, v. 18, n. 1, p. 73, 1997.
- 27 PENHARBEL, E. A. et al. Time de futebol de robôs y04 do centro universitário da fei.

- 28 EICHHAMMER, E. *QCustomPlot*. Acessado 28/06/2016 22:09. Disponível em: <<http://qcustomplot.com/>>.
- 29 ITSEEZ. *OpenCV*. Acessado 29/06/2016 16:53. Disponível em: <<http://docs.opencv.org/3.1.0/>>.