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

Coorientação: Prof. Me. Angelo Darcy

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

Capítulo 1

Introdução

1.1 Contexto

1.2 Justificativa e Objetivos

Este trabalho tem por objetivo principal automatizar o sistema de identificação de objetos coloridos em imagens provenientes de uma câmera em imagens de tempo real, fazendo a calibração dos valores HSV definindo seus limites mínimos e máximos. Se fez necessário um sistema para tal finalidade notando-se:

- 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.

Para alcançar o objetivo principal, foram propostos os seguintes 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.

1.3 Trabalhos Corelatos

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

1.3.1 Calibra

O Centro Universitário da FEI, como publicado em artigo do I Encontro de Robótica Inteligente(1), utiliza em sua equipe Y04 um sistema denominado CALIBRA(2). Desenvolvido para sistemas Linux e com Graphical User Interface(2), o sistema de calibração possui um módulo chamado de MainWindow, que é responsavel pela configuração de brilho, cor e contraste da imagem adquirida pela câmera e gera um arquivo que é analizado na hora da criação das cores padrão(1), onde cores-padrão são definidas como intervalos no espaço de cores HSI(1).

Figura 1.1: Sistema Calibra desenvolvido pelo Centro Universitário da FEI (2)

1.3.2 VSS-Vision

Em 2015 Rosa(3) descreveu 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.

O autor menciona que a calibração de cores e feita calibrando obrigatoriamente laranja, amarelo e azul, e então as outras cores referentes aos jogadores em campo. Como visto na Figura 2.8 a imagem da câmera é dividida em nove cantos, e para calibrar a cor o usuário deve clicar em cima da cor que gostaria de ser calibrada, assim salvando um intervalo de cor tratado como RGB máximo e o mínimo daquela cor , a medida que vão havendo os cliques o sistema verifica para cada atributo 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(3) e esse processo

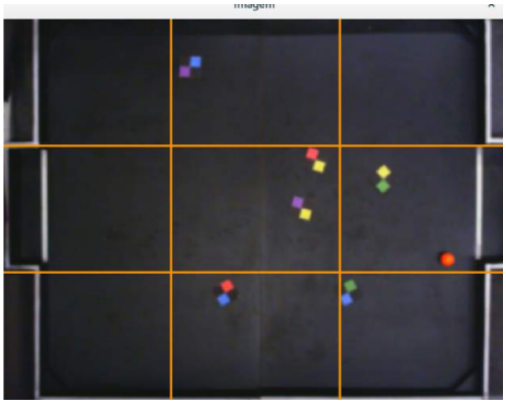

Figura 1.2: Sistema de calibracao desenvolvido peloSIRLab (3)

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 2.9. Este processo de calibração pode demorar entre cinco e dez minutos. O desenvolvimento do sistema utiliza para procesamento de imagens a biblioteca OpenCV e para telas interativas a biblioteca ImGui.

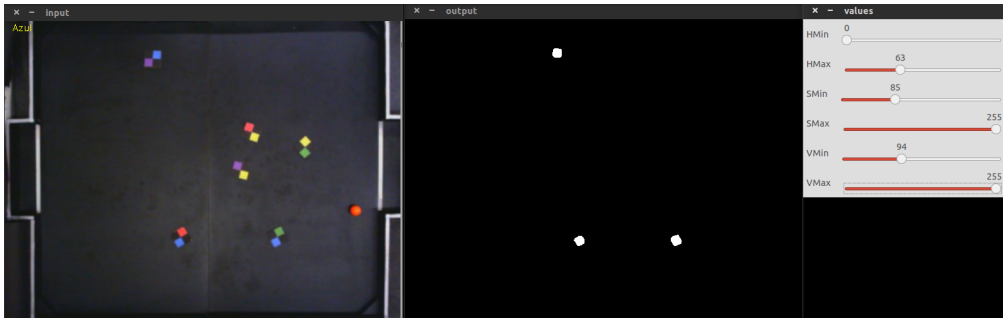

Figura 1.3: Sistema de calibracao desenvolvido peloSIRLab (4)

O atual sistema de visão computacional do SIRLab passou por algumas mudanças desde 2015 e conta com uma inteface e metodo de calibração diferentes(4). Como disponível no repositorio online do Laboratorio, o atual sistema de calibração de cores utiliza o espaço de cores HSV, no lugar do RGB(3). A antiga inteface do sistema, feita inicialmente em ImGui deu lugar a nova, desenvolvida em Qt, como mostra a Figura 2.10.

O método de calibração de cores também foi modificado, segundo a equipe(4) 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 2.11 demonstra o novo método de calibração.

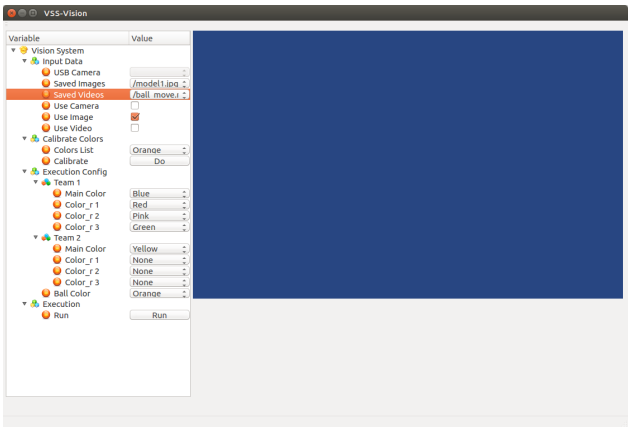

Figura 1.4: Nova interface do time da SIRLab (4)

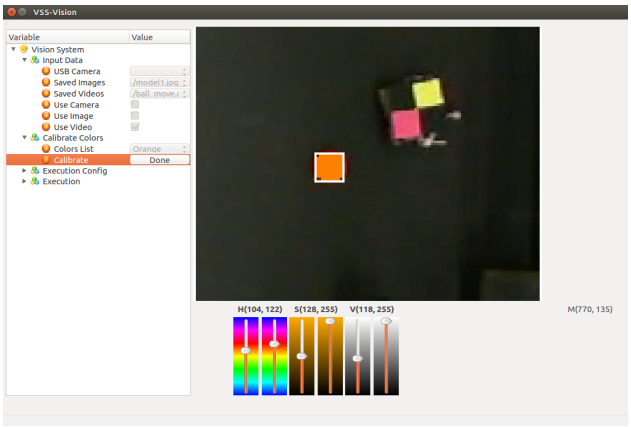

Figura 1.5: Calibração atual do time da SIRLab (4) após calibração

1.4 Organização da Trabalho

Este trabalho está divido em cinco capítulos, incluindo esta introdução como capítulo inicial.

No segundo capítulo encontra-se a fundamentação teórica do texto, contento informações sobre processamento de imagens referente à detecção de objetos e cores. A descrição da probabilidade usada no trabalho, na descoberta do tamanho desejavel dos objetos, uma breve descrição sobre o futebol de robôs.

No terceiro capítulo encontra-se todo o desenvolvimento do projeto, suas classes, a descrição das tecnologias utilizadas no projeto, e o detalhamento de cada um dos tipos de calibração.

No quarto capítulo são apresentados os testes e os resultados obtidos em cada um dos testes do projeto.

No quinto capítulo são feitas as conclusões do trabalho e considerações finais, bem como expostas melhorias que possam vir a ser implementadas em trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Processamento de Imagens

2.1.1 Detecção de Objetos

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 especificas. Quando essa técnica se junta ao processamento de imagens, onde são estas características são acentuadas em um determinado 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(5). 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(6), análise de componentes principais, análise independente de componentes, fatoração de matriz não-negativa, análise discriminativa linear, boosting(7), além da classificação binaria, onde se considera a detecção do objeto em tamanho fixo apenas variando na posição na imagem(8).

2.1.2 Detecção de Bordas

Para um objeto poder ser detectado por algum método de detecção a imagem passa por um processo de segmentação. A segmentação pode ser dita como o processo de divisão da imagem em objetos(9). De acordo com Wangenheim(10) 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 pixeis provenientes da descontinuidade são agrupados de acordo com a proximidade um dos outros para formar os objetos de interesse. De acordo com Canny(11) 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, este tambem é considerado um processamento de imagem de baixo nivel, uma vez que age diretamente na imagem original apenas melhorando-a. 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(12) 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(12), 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(11) para essa detecção que devem ser levados em consideração, Taxa de Erro e Localização(12).

Taxa de Erro É importante que as bordam 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.(10, 11, 12)

Localização A distância entre os pixels de borda encontradas pelo algoritmo e a borda atual deveriam ser o menor possível.(10)

Ao tentar aplicas 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. (10, 11, 12)

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(12).

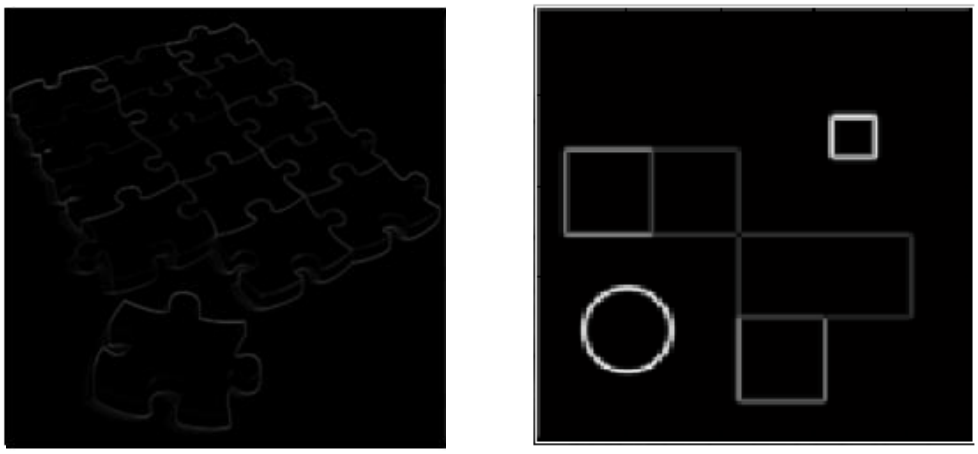

Figura 2.1: Detecçcao de Borda com Algoritmo de Canny (13)

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 responsaveis por distinguirem tons de cinza e pela visão periferica e tem como caracteristica serem sensiveis a baixo nivel de luminosidade(14), os cones, por sua vez, são sensiveis ao alto nivel de iluminação e responsaveis pela percepçao de cores(14). Segundo a teoria tricomática de Thomas Young e mais tarde estudada por Hermann von Helmholtz(14), a retina humana é formada por três tipos de fotopigmentos que seriam os três receptores de cor, ou seja respondiam ao comprimento de onda de apensa três cores: Vermelho, Verde e Azul. A informação obtida atraves do sistema visual humano é assimilada pelo nosso cérebro e ligando a cor a sua aparência levando em consideração o aprendizado que obtivemos sobre a mesma, já para uma máquina cores são números, códigos, cada cor contém um código especifico e cada uma de suas variâncias também. Para o nosso cérebro é muito fácil entender, exemplo, que o verde, verde lima, verde escuro são todos verde, apenas com tonalidades diferentes, já para o computador estas são: (0,255,0),(50,205,50),(0,128,0), no padrão de cor RGB. Mas se for aplicado luminosidade nessas cores, por exemplo, elas ainda se tornam outras diferentes cores, um código diferente para cada luminosidade possível.

Para poder explicar as propriedades e comportamentos das cores em determinadas circunstâncias, surgiram os **Sistemas de cores**. Devido a complexidade existente em explicar todos os aspectos relacionados às cores são utilizados diversos sistemas para descrever as mais diferentes caracteristicas das cores e sua percepção pelo ser humano(14). Dentro os sistemas mais conhecidos, estão o RGB, HSV E HSL, sistemas quais falaremos neste trabalho.

Segundo Azevedo(14) o universo de cores que podem ser reproduzidas por um sistem é chamado de **Espaço de Cores**. De acordo com Foley et. al citado por Souto(15) 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.

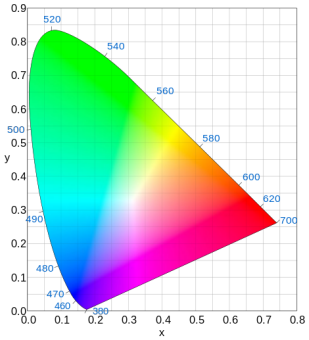

Figura 2.2: Universo de cores estabelecido pelo CIE.

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 conseguir cobrir todas as alterações

e caracteristicas as cores em 1921 a Comissão Internacional de Iluminação (CEI)(15) definiu três primarias(X, Y e Z) que podem ser combinadas para formarem todas as cores e 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. Assim entendeu-se então porque em seu experimento Newton não obteve sucesso para gerar a cor branca, pois parar gerar a cor branca é necessário a soma das três cores primarias azul, verde e vermelho, uma vez que seu experimento utilizava a reflexão e não emissão de cores.

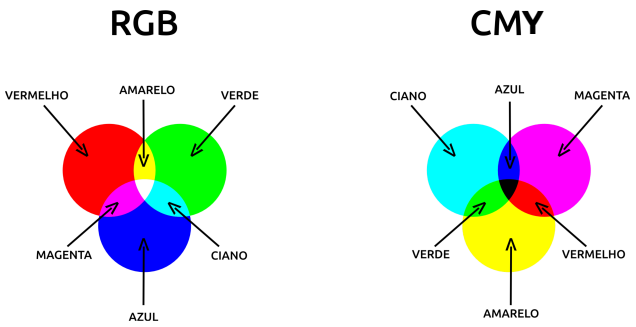

Figura 2.3: Exemplo dos espaços de cores RGB E CMY.

Para utilização pratica de sistemas e espaços de cores e descreverem as cores foram criados modelos de cores, neste trabalho falaremos sobre o RGB e HSV, pois são os modelos usados durante o desenvolvimento. Modelo de cores são modelos matemáticos utilizados para classificação das cores de acordo com sua tonalidade, saturação, luminosidade ou crominância na tentativa de conseguir cobrir o maior número de cores possíveis e assim simulando a visão. A representação da cor é definida por um único ponto em um modelo tridimensional. Os modelo de cores tem função definir as cores nos programas gráficos de computadores de forma que combine com a percepção das cores pelo sistema visual humano e utiliza três eixos similares para definirem a cor(16).

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 primarias, 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.

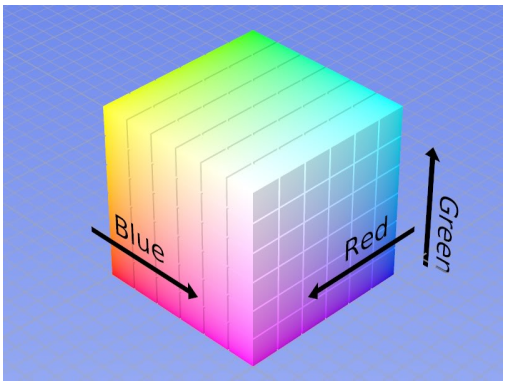

Figura 2.4: Exemplo do Modelo de Cor RGB. Horvath(17)

O modelo HSV define tonalidade (hue) que é a cor em si, variando de 0 a 360º, a saturação(saturnation) 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) que tenta fazer referência à percepção humana(16) que é a intensidade da cor, escala de tons de cinza(14), variando também de 0 a 1.

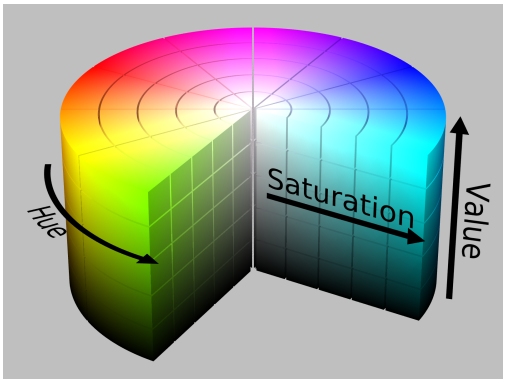

Figura 2.5: Exemplo do Modelo de Cor HSV, Horvath(17)

O sistema HSV ultiliza definiçoes de cor mais intuitivas que o conjunto de cores primarias, por isso são mais adequados quando se necessita obter varias tonalidades.

2.3 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 referencia t para encontrar a Tabela T(18) 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(19).

2.3.1 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(20). A teoria T de Student é um teoria usada em pequenas amostras, ou seja, amostras com tamanho menor que 30.

De acordo com Spiegel (19), a definição da distribuição de "Student"t é dada por:

$$t = \frac{\bar{X} - \mu}{s}\sqrt{N-1} = \frac{\bar{X} - \mu}{\hat{s}/\sqrt{N}}$$

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.

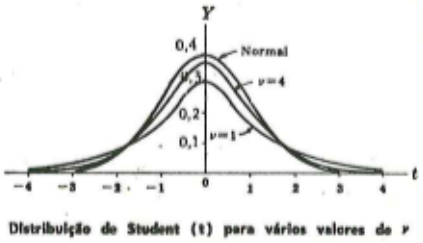

Figura 2.6: Distribuição de Student(t) para valores de v. Spiegel (19)

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 ν = (N - 1) é denominada *número de graus de liberdade* ν.

2.3.2 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, ν = 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, ν = N-1.

2.3.3 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 iniciai 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(18). Para os propósitos finais foi escolhido um nível de confiança de 95

2.4 Futebol de Robôs

Visto como um dominio bastente complexo, dinamico e imprevisivel(21), o futebol de robos surgiu como uma tentativa de promover pesquisas nos campos de Inteligencia Artificial e robotica, pela avaliacao teorias, algoritmos e arquiteturas atraves problemas padrao(22). 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(3). O futebol de robos se assemelha ao futebol humano onde o objetivo do jogo é fazer gols para vencer a partida, porem tendo regras adaptadas para o "ambito"robotico. Rosa(3) 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 inicio de partida ou gol feito a bola deve ser posicionada no centro do campo e os robôs devem ser posicionados de acordo com a posse de bola.

2.5 Trabalhos Relacionados

Para o tema especifico deste trabalho, calibração de intervalo de cores para times de futebol de robos da categoria very small size, não foram encontrados trabalhos relacionados, porém foram encotrados Team Discription Papers e descrições de sistemas usados pelos times, onde consta sobre o processo de calibração e os metodos usados.

2.5.1 Calibra

O Centro Universitário da FEI, como visto em(1), utiliza em sua equipe Y04 utiliza um sistema, desenvolvido denominado CALIBRA(2).Desenvolvido para sistemas Linux e com Graphical User Interface(2), o sistema de calibração possui um modulo chamado de MainWindow, que é responsavel pela configuracao de brilho, cor e contraste da imagem adquirida pela camera e gera um arquivo que é analizado na hora da criacao das cores padrao(1), onde cores-padrão são definidas como intervalos no espaço de cores HSI(1).

Figura 2.7: Sistema Calibra desenvolvido pelo Centro Universitário da FEI (2)

2.5.2 VSS-Vision

Em 2015 Rosa(3) 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-Rio), o sistema de visão computacional da equipe, durante a competição do ano de 2014, que abrange inclusive a parte de calibração.

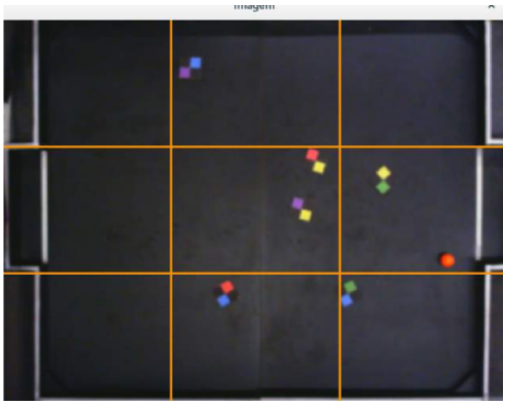

Figura 2.8: Sistema de calibracao desenvolvido peloSIRLab (3)

O autor menciona que a calibração de cores e feita calibrando obrigatoriamente laranja, amarelo e azul, e então as outas cores referentes aos jogadores em campo. Como visto na Figura 2.8 a imagem da camera é dividida em nove cantos, e para calibrar a cor o usuario 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(3) e esse processo deve ser feito em cada um dos nove cantos da imagem. Os valores HSV encontrados sao ajustados manualmente com a ajuda de sliders, como visto na Figura 2.9. Este processo de calibração pode demorar entre cinco e dez minutos. O desenvolvimento do sistema utiliza para procesamento de imagens a biblioteca OpenCV e para telas interativas a biblioteca ImGui.

O atual sistema de visão computacional do SIRLab passou por algumas mudanças desde

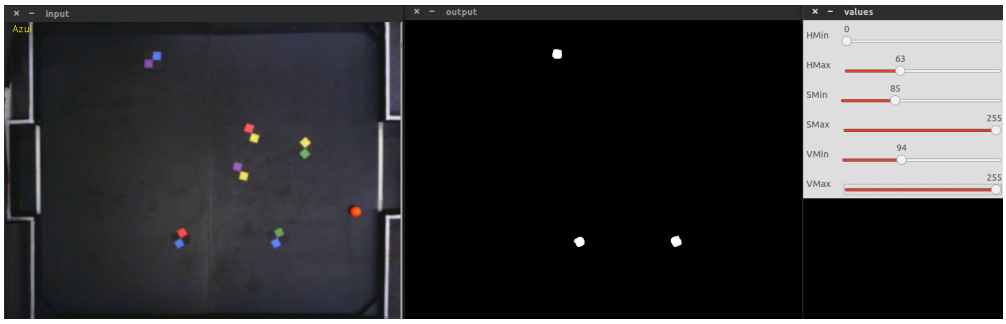

Figura 2.9: Sistema de calibracao desenvolvido peloSIRLab (4)

2015 e conta com uma inteface e metodo de calibração diferentes(4). Como disponivel no repositorio online do Laboratorio, o atual sistema de calibração de cores utiliza o espaço de cores HSV, no lugar do RGB(3). A antiga inteface do sistema, feita inicialmente em ImGui deu lugar à nova, desenvolvida em Qt, como mostrado na Figura 2.10.

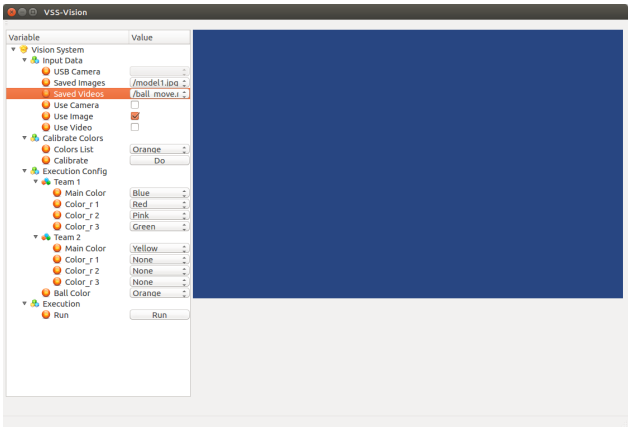

Figura 2.10: Nova interface do time da SIRLab (4)

O metodo de calibração de cores tambem foi modificado, segundo a equipe(4) 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 2.11 demonstra o novo metodo de calibração.

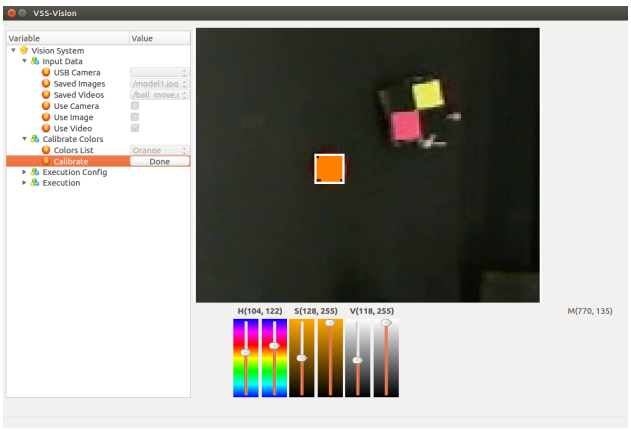

Figura 2.11: Calibração atual do time da SIRLab (4)

Capítulo 3

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 linguagem 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 imagem 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 feito a contagem de ocorrências de cada cor.

3.1 Tecnologias Usadas

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 estará presente no Capítulo de Metodologia.

OpenCV Lançado em 1999 pela Intel(23), com objetivo de ser otimizada, portável e com um grande número de funções, o Open Source Computer Vision Library,OpenCV, se

tornou se tornou uma ferramenta que possui mais de 2500 algoritmos e 40 mil pessoas em seu grupo de usuários(23). 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.(24)

3.2 Projeto

3.2.1 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 usuario e as funções do sistema.

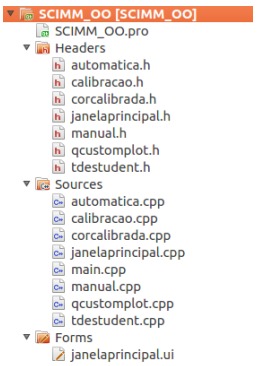

Figura 3.1: Organização das pastas do projeto

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, janelaprincipal e tdestudent. Já a classe qcustomplot é um componente para auxilio em plotagem

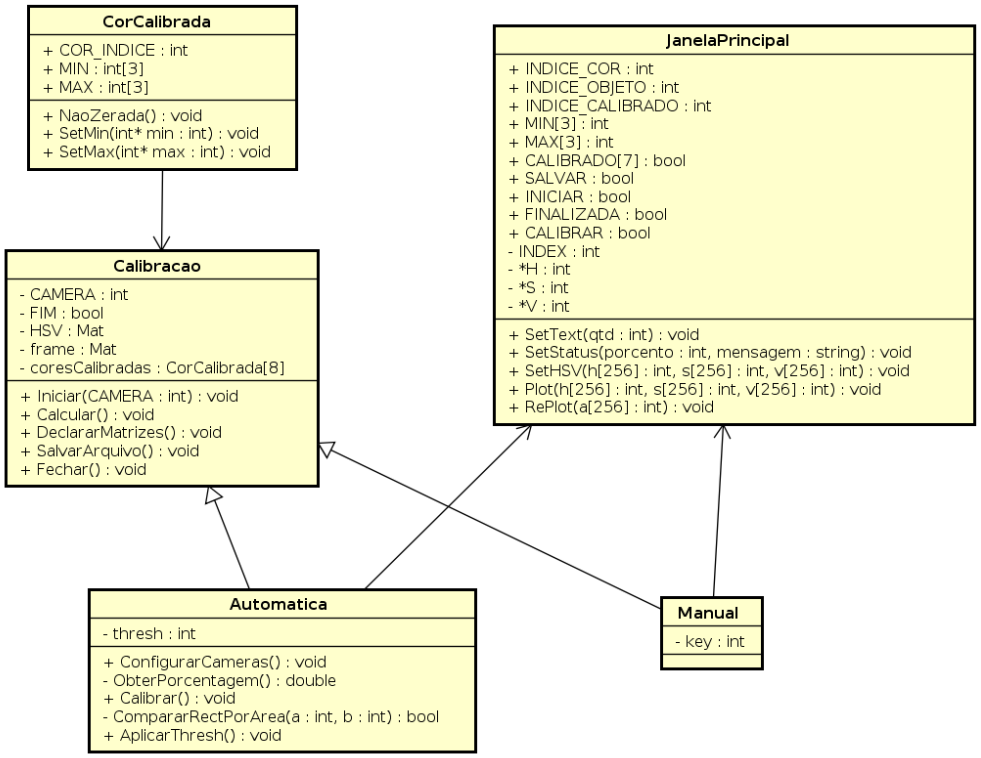

Figura 3.2: Diagrama de Classes do projeto

de gráficos e vizualização de dados(25). Para melhor entendimento da interação entre as classes a figura 3.2 trás o diagrama de classes do projeto.

3.2.2 Classes

main

Esta é o que se chama de *ponto de entrada* em programação.*Ponto de entrada* é onde o sistema operacional ira iniciar a execução do sistema desenvolvido. Esta classe possui somente o metodo **main** e este invoca a classe **JanelaPrincipal** que apresenta a interfaçe grafica para interação com o usuario, e de acordo com esta interação iniciar o processo de calibração.

JanelaPrincipal

É uma classe-objeto que utiliza da implementação de objetos QWidget e suas subclasses disponiveis pelo framework Qt, para a criação de uma interface grafica que faz a interação com o usuario e que seleciona o tipo de calibração a ser iniciada. Todos os metodos presentes nessa clase são para utilização grafica, ex. comportamento de botões, slider, abas, menus de seleção, gráficos, bem como a comunicação com as classes de calibração.

Calibracao

É classe base que contem a definição dos metodos e variaveis que virão a ser usadas por ambas as classes **Manual** e **Automatica**. Seus metodos são:

Iniciar: Método que inicia ambos os proceessos de calibração, invocando os metodos necessarios antes de se fazer a calibração em si.

Calcular: Método onde cada pixel é analizado e calculadas as ocorrências de valores HSV.

Fechar: Método que finaliza o processo de calibração, fechando todas as janelas.

SalvarArquivo: Método onde é criado o arquivo com intervalo de valores, através da lista de minimos e maximos.

DeclararMatrizes: Método que é invocado quando tem a necessidade de ultilizar novamente as matrizes, estas então precisar ser zeradas.

Manual

É classe que contem os metodos e variaveis oriundos da classe base **Calibracao** e os implementa de acordo com a necessidade da calibração manual, são eles:

Iniciar: Método onde a camera é inciada e caso esteja disponivel a rotina de calibraçao é iniciada. Ao ser iniciada a rotina entre em uma sequencia que se repete até que a calibração seja finalizada. Durante a sequencia o sistema recebe a seleção dos objetos e invoca o metodo **Calcular**. Utilizando a interface grafica o usuario informa ao sistema quando a selecao dos objetos foi finalizada. O passo a seguir é passar para a interface gráfica os valores encontrados e para plotagem do grafico

Calcular: Método onde cada pixel pertencente à um objeto é analisado e seus valores HSV adicionados a lista de ocorrencia. O algoritmo de contagem de ocorrencia em cada pixel é dado da seguinte maneira:

Algoritmo 1 Contagem dos Valores HSV - Classe manual

```
para todo pixel do objeto faça
    OcorrenciasDeH[pixel.h]++
    OcorrenciasDeS[pixel.s]++
    OcorrenciasDeV[pixel.v]++
fim para
```

DeclararMatrizes: Método que é invocado ao final da calibração de cada cor onde as listas de valores HSV seram zeradas para que os valores de uma calibração não influenciem na calibraçã seguinte.

Automatica

É classe que contem os metodos e variaveis proprios além dos oriundos da classe base **Calibracao** e os implementa de acordo com a necessidade da calibração manual:

Os métodos da propria classe são:

EliminarExcessos: É o metodo que usa a classe **TdeStudant** para, a partir da lista de todos os objetos encontrados, obter os limites de tamanho minimo e maximo de um objetos e diminuir a quantidade de objetos a serem alanisados por exclusão de tamanho dos objetos que não estão nos limites.

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

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

ConfigurarCamera: Método onde a imagem de exibição é ajustando utilizando configurações de brilho, contraste e tamanho de tela.

Calibrar: Método onde é exibido a imagem da camera e quando estiver de acordo com a exigencia do usuario o mesmo inicial a calibração. So apos ser iniciada a calibração o método **AplicarThresh** é invocado, apos os objetos de cores serem identificados, é retornada à tela do usuario a lista do objetos.

Os métodos oriundos da classe base são:

Iniciar: Método onde é o ID da camera é fixado dentro da classe, também é o método que faz a gerencia dos metodos inciais. Incovando primeiro o metodo de configuração de imagem, durante o método **ConfigurarCamera** para se configurar a imagem, que sera

analizada durante a calibração, onde é feita a tentativa de acesso a camera. Caso esta esteja disponivel a configuração finaliza e somente após a mesma finalizada se inicia a rotina de calibração, caso a camera esteja indiponivel a calibração fica impedida de iniciar.

Calcular: Método onde cada pixel pertencente à cada um dos objetos encontrados é analisado e seus valores HSV adicionados a lista de ocorrencia. O algoritmo de contagem de ocorrencia em cada pixel é dado da seguinte maneira:

Algoritmo 2 Contagem dos Valores HSV

```
para todo objeto encontrado faça
    para todo pixel do objeto faça
        OcorrenciasDeH[pixel.h]++
        OcorrenciasDeS[pixel.s]++
        OcorrenciasDeV[pixel.v]++
    fim para
fim para
```

3.3 O Sistema

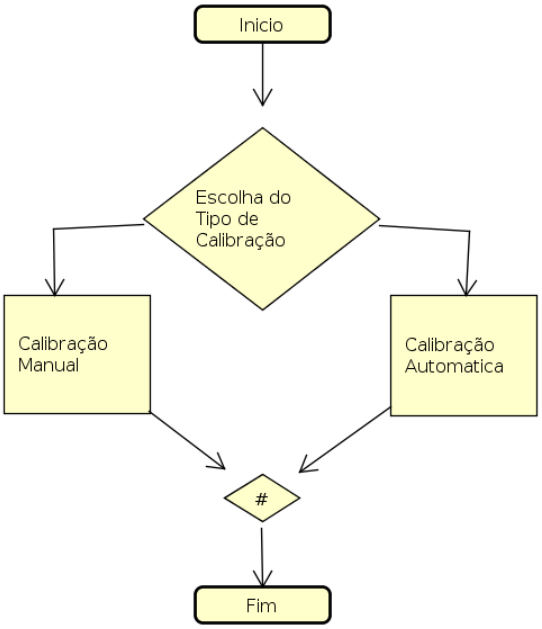

Figura 3.3: Diagrama de Fluxo

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 usuario. A **interface grafica** que por sua vez oferece as duas possibilidades ao usuario, de acordo com o tipo de calibração escolhido o sistema inicia a rotina de calibração referente. Apos a execução de toda o sistema é finalizada.

3.3.1 Calibração Automatica

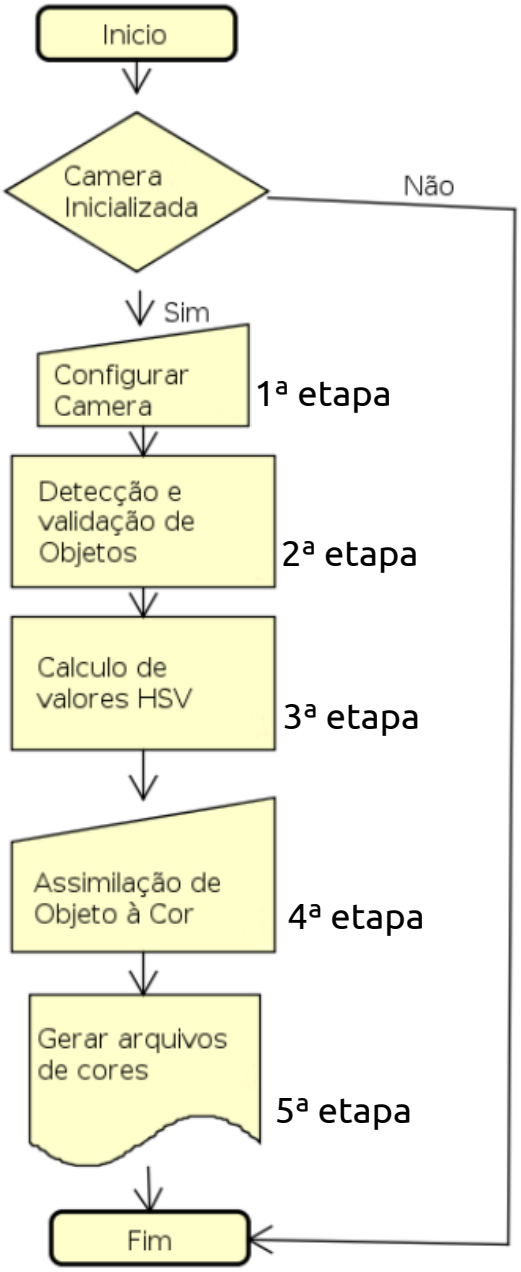

Figura 3.4: Diagrama de Fluxo Automatico

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

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

1ª Etapa - Configuração de Camera

Antes de ser feito a calibração propriamente dita são necessarias duas configurações: Ajuste de Constrate 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(26), O terceiro **contrast_value / 50.0** indica o valor de constraste, ou alpha, a ser usado para multiplicar os valores do pixel da imagem(26) 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 interção 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 iniciai 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 expessura da demarcação, **5** que significa o tipo de linha a ser utilizado na demarcação e **0** que é o numero de bits fracinarios. Após confirmada a escolha do tamanho da tela este é então salvo na variavel nomeada *tamanho*, está então sera usado durante todo o processo de calibração.

2ª 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 atravez da detecção de bordas é utilizado desfoque na imagem. O algoritmo de Canny já está implementado dentro da biblioteca OpenCV e com a seguite 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 orignal tranformada para escala de cinza, esta é a imagem na qual o algoritmo sera aplicado. **canny_output** será a imagem de saida da função. **thresh** e **thresh*3** são os limites mínimos e máximos para considerar uma borda. **3** é o valor de apertura ou kernel, o valor 3 é utilizado como padro.

Apos o uso do algoritmo de Canny para detecção de bordas é necessario 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* ira 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 (26). **hierarchy** é onde será salva um verot de informações sobre a topologia da imagem, e terá como total de elementos o mesmo numero que o total de contornos encontrado(26). O quarto parametro, **CV_RETR_EXTERNAL** indica o modo de obtenção de contornos, nesse caso *CV_RETR_EXTERNAL* indica que o metodo só obtera os contornos exteriores(26). **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(26). 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 sera necessario.

Uma vez obtidos os contornos é necessarios que se faça a eliminação de vertices dos polignos encontrados nos objetos deixando assim o objeto mais preciso. Isso é necessario para deixar a forma encontrada mais precisa dá forma original. Para este ajuste foi usado o metodo *approxPolyDP*, ja 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 paralametro, **Mat(contours[i])** que é a criação de uma nova imagem, somente com aquele unico objeto, que esta sendo analisado. A seguir é informado no segundo parametro a variavel de destino **contours_poly[i]**, onde sera salvo o objeto com a eliminação dos vertices. O terceiro parametro indica o valor do *epilson*, usado

o valor **3** que especifica a precisão da aproximação, a distância máxima entre a curva original e a sua aproximação(26). 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 necessario 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 possiveis 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 probabilistico 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.

3ª Etapa - Calculo de valores HSV

Para que possam ser feitos os cálculo de valores HSV mínimo e máximos é necessario 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 deferenças 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 necessario, 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 posiçoes, uma vez que se sabe que os valores de um pixel varial de 0 a 255. As ocorrençias são salvas se baseando no valor HSV do pixel. Para isso o pixel é separado em valores H, S e V este então tera seu valor correspondente a posição no vetor apropiado, 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.

Uma vez terminada a analise dos pixeis do objeto o sistema procupa 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 buscaria 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.

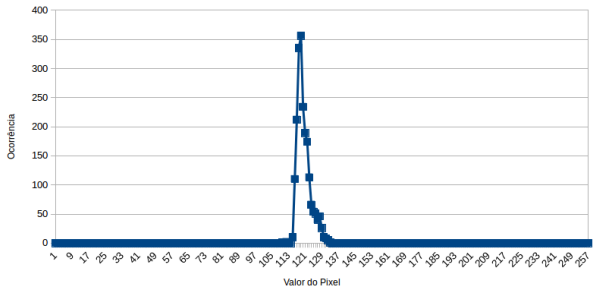

Figura 3.5: Grafico Exemplificando a Ocorrencia dos valores de H.

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.

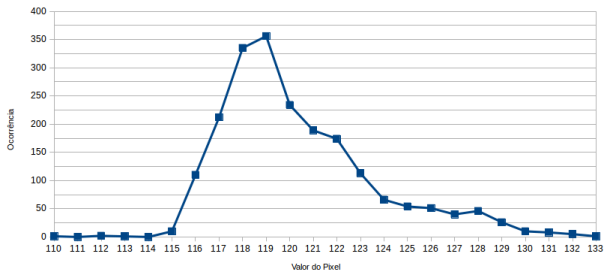

Figura 3.6: Grafico Exemplificando a Ocorrencia dos valores de H após eliminação.

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.

4ª 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.

5ª 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 calibraa e seus valores máximos e mínimos.

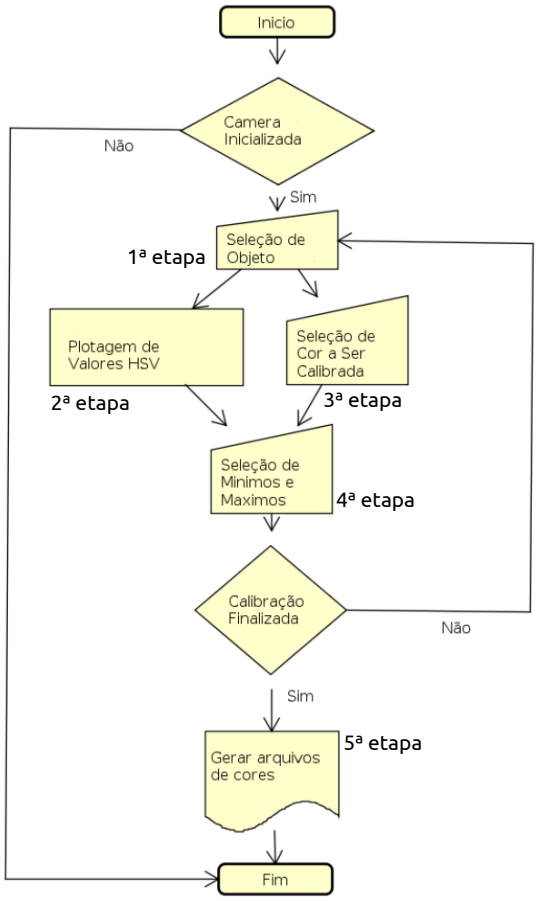

Figura 3.7: Diagrama de Fluxo Manual

3.3.2 Calibração 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 metodo de calibração é manual, sendo assim é necessario que o usuario selecione o objeto que o mesmo quer calibrar.

1ª Etapa - Seleção de Objeto

O primeiro passo para ser feita a calibração manual é a seleção de Objeto. Com o auxilio da função setMouseCallback, disponivel na biblioteca **OpenCV** e já explanada na subsessão 3.2.1, é possivel obter os pontos de uma seleção feita na imagem, seu uso é necessario, já que obtenção de um ou mais objetos com a **mesma cor** a serem analisados é feita somente com a interação do usuario. 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 ocorrencia calculado

utilizando o mesmo metodo encontrado na sessão 3.2.1 subsessão **Calculo de valores HSV**, estes valores são armazenados em memoria de execução e somente mostrados ao usuario quando requisitados.

2ª Etapa - Plotagem de Valores HSV

Uma vez que o usuario já selecionou todos os objetos e os mesmos já foram analisados, o processo de calibração encaminha à interface grafica os valores encontrados e está plota 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 usuario quais foram os valores obtidos na seleção.

3ª Etapa - Seleção de Cor a ser Calibrada

A seleção da cor na lista de cores possiveis para ser assimilado, pode ser considerada uma ação concomitante à ação de plotagem de graficos sendo que uma não depende dá outra, já que no momento da seleção do objeto o usuario 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.

4ª Etapa - Seleção de Minimos e Maximos

Assim que os graficos são plotados, se torna então possivel, 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 usuario a ocorrencia dos pixeis. Ao contrario do sistema automatico, no sistema manual o objeto CorCalibrada só é criado após a seleção da cor e dos intervalos pelo usuario. Quando este salva a cor na interface gráfica, a mesma cor é encaminhada para o sistema manual e assim salva na lista.

5ª Etapa - Gerar Arquivo de Cores

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

Capítulo 4

Resultados

Para obtenção dos resultados foram organizados dois tipos de testes: Teste por região e Teste geral.

No teste por região o campo foi dividido verticalmente em cinco faixas e cada faixa dividia em três partes horizontais. Cada uma das faixas possue ao seu centro um conjunto de cinco cores. Cada faixa horizontal possui vinte e nove centimetros, já as faixas verticais possuem quarenta e um centimetros com um intervalo de um centimetro entre elas.

Em cada ponto de teste estão disposta cinco cores: Vermelho e Verde na primeira linha, Amarelo, Azul e Laranja na segunda. As cores estão distantes verticalmente *7.5cm*, na primeria linha a distancia entre as cores é de *11 cm* e na segunda linha de *7.2 cm*. Um melhor detalhamento da disposiçao das cores é mostrado na Figura 4.1.

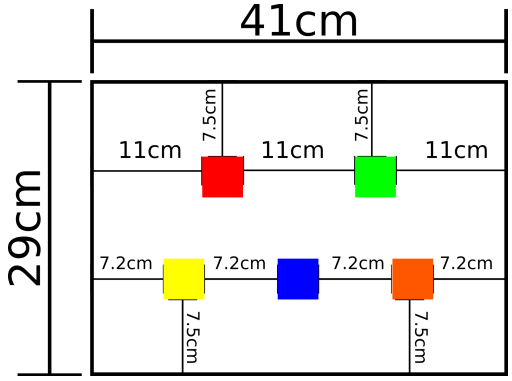

Figura 4.1: Disposição dos objetos coloridos dentro de cada uma das partes

No total estavam disposto no campo quinze pontos de cada cor, sendo as cores: . Sendo cinco areas horizontais e três verticais, a divisão do campo resulta em quinze partes, cada um das partes sera testada vinte e cinco vezes. Para cada uma das faixas foram testadas calibrações com cinco tipos diferentes de contraste e luminosidade, somando um total de vinte e cinco testes.

Tabela 4.1: Tabela de Testes

Tipo de Teste	Quantidade
Divisões no Campo	25
Constrate	5
Brilho	5
Total	260

Capítulo 5

Conclusão

Bibliografia

1 PENHARBEL, E. A. et al. Time de futebol de robôs y04 do centro universitário da fei.

2 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.

3 ROSA, J. F. da. *Construção de um time de futebol de robôs para a categoria IEEE Very Small Size Soccer*. Dissertação (Trabalho de Conclusão de Curso) — FACULDADE DEE DUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO, 2015. Disponível em: ⟨http://docplayer.com.br/11866286-Construcao-de-um-time-de-futebol-de-robos-para-a-categoria-ieee-very-small-size-soccer.html⟩.

4 SIRLAB. Imagens hsv, hsl e rgb. In: . [s.n.]. Acessado 06/06/2016 14:50. Disponível em: ⟨https://github.com/SIRLab/VSS-Vision/wiki⟩.

5 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.

6 NASCIMENTO, M. C. *Detecção de Objetos em Imagens*. Dissertação (Trabalho de Graduação) — Centro de Informática, Universidade Federal de Pernambuco, 2007. Disponível em: ⟨www.cin.ufpe.br/~tg/2007-2/mcn2.doc⟩.

7 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.

8 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.

9 GONZALEZ, R.; WOODS, R. *Digital Image Processing*. Pearson/Prentice Hall, 2008. ISBN 9780131687288. Disponível em: ⟨https://books.google.com.br/books?id=8uGOnjRGEzoC⟩.

10 WANGENHEIM, A. Von. encontrando a linha divisória: Detecção de borda. *Departamento de Informática e Estatística-Universidade Federal de Santa Catarina, 2013a*, v. 16, 2014. Disponível em: ⟨www.inf.ufsc.br/~visao/bordas.pdf⟩.

11 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.

12 VALE, G. M. do; POZ, A. P. D. O processo de detecção de bordas de canny: Fundamentos, algoritmos e avaliação exeperimental. In: BERGIN, T. J.; GIBSON, R. G. (Ed.). *Simpósio Brasileiro de Geomática*. [S.l.: s.n.], 2002. p. 292–303.

13 SAINI, S.; KASLIWAL, B.; BHATIA, S. Comparative study of image edge detection algorithms. *arXiv e-print service in the fields of physics, mathematics, computer science, quantitative biology, quantitative finance and statistics, Cornell University Library*, 2013. Disponível em: ⟨https://arxiv.org/pdf/1311.4963⟩.

14 AZEVEDO, E.; A., C. *Computação Gráfica: Geração de Imagens*. [S.l.]: Elsevier, 2003. ISBN 9878535212525.

15 SOUTO, R. P. *Segmentação de imagem multiespectral utilizando-se o atributo matiz*. Dissertação (Dissertação de Mestrado) — INPE, São José dos Campos, 2003.

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 HORVATH, M. *"Imagens HSV, HSL e RGB"*. Disponível em: ⟨http://isometricland.net/artwork/artwork.php⟩.

18 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⟩.

19 SPIEGEL, M. R. *Estatística*. [S.l.]: McGraw-Hill do Brasil, 1974.

20 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⟩.

21 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.

22 KITANO, H. et al. Robocup: A challenge problem for ai. *AI magazine*, v. 18, n. 1, p. 73, 1997.

23 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.

24 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.

25 EICHHAMMER, E. *QCustomPlot*. Acessado 28/06/2016 22:09. Disponível em: ⟨http://qcustomplot.com/⟩.

26 ITSEEZ. *OpenCV*. Acessado 29/06/2016 16:53. Disponível em: ⟨http://docs.opencv.org/3.1.0/⟩.