

Dissertação apresentada à Pró-Reitoria de Pós-Graduação e Pesquisa do Instituto Tecnológico de Aeronáutica, como parte dos requisitos para obtenção do título de Mestre em Engenharia no Curso de Mestrado Profissional em Engenharia Aeronáutica no Programa de Pós-Graduação em Engenharia Aeronáutica e Mecânica.

Felipe de Paula Lima

**SISTEMA DE VISÃO COMPUTACIONAL
APLICADO AO PROCESSO DE SELAGEM DE
ESTRUTURAS E COMPONENTES
AERONÁUTICOS**

Dissertação aprovada em sua versão final pelos abaixo assinados:


Prof. Dr. Luis Gonzaga Trabasso

Orientador


Eng. Msc. Elton Candia Cordeiro

Coorientador

Prof. Dr. Luiz Carlos Sandoval Góes

Pró-Reitor de Pós-Graduação e Pesquisa

Campo Montenegro
São José dos Campos, SP - Brasil
2017

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

de Paula Lima, Felipe
Sistema de Visão Computacional Aplicado ao Processo de Selagem de Estruturas e Componentes Aeronáuticos / Felipe de Paula Lima.
São José dos Campos, 2017.
79f.

Dissertação de Mestrado – Curso de Mestrado Profissional em Engenharia Aeronáutica. Área de Sistemas Aeroespaciais e Mecatrônica – Instituto Tecnológico de Aeronáutica, 2017. Orientador: Prof. Dr. Luís Gonzaga Trabasso. Coorientador: Eng. Msc. Elton Candia Cordeiro.

1. Asa. 2. Automação da Manufatura. 3. Filete. 4. OpenCV. 5. Prendedor. 6. Selagem. 7. Visão Computacional. I. Centro Técnico Aeroespacial. Instituto Tecnológico de Aeronáutica. Divisão de Engenharia Aeronáutica. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

DE PAULA LIMA, Felipe. Sistema de Visão Computacional Aplicado ao Processo de Selagem de Estruturas e Componentes Aeronáuticos. 2017. 79f.
Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Felipe de Paula Lima

TÍTULO DO TRABALHO: Sistema de Visão Computacional Aplicado ao Processo de Selagem de Estruturas e Componentes Aeronáuticos.

TIPO DO TRABALHO/ANO: Dissertação / 2017

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias desta dissertação e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta dissertação pode ser reproduzida sem a autorização do autor.

Felipe de Paula Lima
Rua Paulo Édson Blair, 65, Apt 91C
12.243-100 – São José dos Campos–SP

**SISTEMA DE VISÃO COMPUTACIONAL
APLICADO AO PROCESSO DE SELAGEM DE
ESTRUTURAS E COMPONENTES
AERONÁUTICOS**

Felipe de Paula Lima

Composição da Banca Examinadora:

Prof. Dr.	Luís Gonzaga Trabasso	Orientador/Presidente	-	ITA
Eng. Msc.	Elton Candia Cordeiro	Coorientador	-	Embraer
Profª. Dra.	Emilia Villani	Membro Interno	-	ITA
Eng. Msc.	Iván García Martinez	Membro Externo	-	Embraer

ITA

À minha família pelo enorme incentivo
e apoio durante toda a minha jornada
acadêmica.

Agradecimentos

Primeiramente, gostaria de agradecer ao meu pai, por ter me dado todo o suporte necessário para a realização deste mestrado.

À minha mãe e irmã por me receberem sempre de braços abertos e me apoiarem.

Aos meus familiares e amigos que se orgulham das minhas conquistas.

Aos amigos do PEE 22/ITA que estiveram presentes durante os longos dias e noites de estudo.

E por fim, mas não menos importante, ao Prof. Dr. Luís Gonzaga, pela orientação e confiança depositadas em mim no desenvolvimento deste trabalho.

*"If everything seems under control,
you're just not going fast enough."*

— MARIO ANDRETTI

Resumo

Com a crescente busca pela automação dos processos de manufatura, procurando sempre o aumento de eficiência, com soluções de baixo impacto ambiental, preservação do bem-estar dos seres humanos envolvidos na tarefa e mínimas mudanças nas instalações físicas já instauradas, está sendo executado pelo ITA em parceria com a Embraer, o projeto Automação da Montagem Estrutural de Asas (AME ASA). Dentro dele está inserido o sistema de selagem automática de asas, denominado sistema SELA, responsável pela automação do processo de selagem de prendedores e filetes em asas. Uma das funções necessárias para a consecução da selagem é a identificação de prendedores e filetes. É este o objetivo deste trabalho. Foi desenvolvido um sistema de visão computacional que identifica o tipo de componente aeronáutico (prendedor ou filete), calcula a posição de seu centro de área, para o caso dos prendedores, ou a posição de suas extremidades, para o caso dos filetes, e envia esta informação para o controlador do robô que, por sua vez, executa as tarefas programadas de selagem, com baixa intervenção do operador. Os equipamentos, métodos e software utilizados no sistema computacional estão detalhados no desenvolvimento dessa dissertação. Um método indicador de desempenho do sistema de visão, capaz de avaliar a taxa de acerto dada uma referência conhecida, foi criado. Ou seja, sabendo-se a quantidade de prendedores no painel enquadrados pela câmera, o indicador retorna a taxa de acerto, considerando prendedores não identificados e falsas identificações. Testes com simulação da variação de luminosidade e seu efeito na taxa de acerto tiveram resultados satisfatórios, mostrando robustez adequada a possíveis variações na iluminação. O sistema SELA prevê um sistema dedicado de luz artificial para aumentar ainda mais a robustez na identificação dos componentes e auxiliar a inspeção de qualidade do selante.

Abstract

With the growing search for the automation of manufacturing processes, always seeking to increase efficiency developing solutions with low environmental impact, preservation of the well being of humans involved in the task and small changes in the facilities already installed, it's being performed by the ITA in partnership with the Embraer, the Automation of Structural Assembly of Wings project (AME ASA). Inside it is inserted the automatic system of wings sealing, denominated SELA system, responsible for the automation of the sealing process of fasteners and fillets in wings. One of the necessary functions to achieve the sealing is the identification of fasteners and fillets. This is the purpose of this work. A computer vision system has been developed, it identifies the type of aeronautical component (fastener or fillet), calculates the position of its area center, for the case of fasteners, or the position of its ends, in the case of fillets, and sends this information for the robot controller which, in turn, performs the programmed sealing tasks, with low operator intervention. The equipment, methods and software used in the computational system are detailed in the development of this dissertation. An indicator method of performance of the vision system, capable of evaluating the hit rate given a known reference, was created. In other words, knowing the amount of fasteners in the panel framed by the camera, the indicator returns the hit rate, considering unidentified fasteners and false identifications. Tests with simulation of the luminosity variation and its effect on the hit rate had satisfactory results, showing adequate robustness to possible variations in the illumination. The SELA system foresees a dedicated artificial light system to further increase the robustness in identifying the components and to assist in the quality inspection of the sealant.

Listas de Figuras

FIGURA 1.1 – Exemplo de sistema de montagem com robô comercial (DEVLIEG; FEIKERT, 2008)	19
FIGURA 2.1 – Caminho do feixe de luz através da lente na câmera. Figura modificada de (MUSSABAYEV, 2015)	24
FIGURA 2.2 – <i>Pinhole</i> , modelo de câmera simplificado. Figura modificada de (MUSSABAYEV, 2015)	24
FIGURA 2.3 – Relação entre a imagem e o sistema de coordenadas padrão da câmera. Figura modificada de (MUSSABAYEV, 2015)	25
FIGURA 2.4 – Representação da imagem (BEVILAQUA, 2011)	28
FIGURA 2.5 – Representação da imagem em RGB. Figura extraída de (WIKIPEDIA, 2017)	29
FIGURA 2.6 – Representação da imagem em HSV. Figura extraída de (WIKIPEDIA, 2017)	29
FIGURA 2.7 – Sistema digital de imagem (GAMAL; ELTOUKHY, 2005)	30
FIGURA 2.8 – Esquemático de funcionamento dos sensores CCD e CMOS (LITWILER, 2005)	31
FIGURA 2.9 – Desfoque gaussiano em um vetor de pixels (BRADSKI; KAEHLER, 2008) .	34
FIGURA 2.10 – Aplicação do desfoque gaussiano (BRADSKI; KAEHLER, 2008)	34
FIGURA 2.11 – Resultados da aplicação do método de detecção de bordas Canny quando os limites superior e inferior foram ajustados para 50 e 10, respectivamente (BRADSKI; KAEHLER, 2008)	35
FIGURA 2.12 – Resultados da aplicação do método de detecção de bordas Canny quando os limites superior e inferior foram ajustados para 150 e 100, respectivamente (BRADSKI; KAEHLER, 2008)	36

FIGURA 2.13 – Aplicação da transformada de Hough para linhas. Muitas linhas não esperadas são encontradas (BRADSKI; KAEHLER, 2008)	37
FIGURA 2.14 – Um ponto (x_0, y_0) no plano da imagem (a) implica em várias linhas parametrizadas por diferentes ρ e θ (b), os diferentes pontos (ρ, θ) formados pelo conjunto dessa linhas correspondem a uma curva característica (c) (BRADSKI; KAEHLER, 2008)	37
FIGURA 2.15 – Detector de bordas Canny (limite inferior=50 e limite superior=150) aplicado primeiro, com os resultados em cinza, e a transformada de Hough probabilística progressiva aplicada posteriormente, com os resultados sobrepostos em branco. (BRADSKI; KAEHLER, 2008)	38
FIGURA 2.16 – Transformada de Hough para círculos encontrando um círculo no padrão de testes (BRADSKI; KAEHLER, 2008)	39
FIGURA 3.1 – Câmera XIMEA - MQ013CG-E2 (Color) - Descrição (XIMEA, 2014).	41
FIGURA 3.2 – Lente FUJINON - HF16HA-1B (FUJIFILM CORPORATION, 2009) . .	42
FIGURA 3.3 – (A)- Imagem distorcida antes da calibração. (B)- Imagem sem distorção após a calibração e correção. (DOCS.OPENCV, 2011-2014) . .	44
FIGURA 3.4 – O cabo <i>NULL Modem</i> , conhecido também como cabo <i>Crossover</i> , é usado para permitir que dois dispositivos seriais <i>Data Terminal Equipment</i> (DTE) comuniquem entre si, sem usar um modem ou um dispositivo <i>Data Communications Equipment</i> (DCE).	44
FIGURA 3.5 – Fluxograma do código desenvolvido para o sistema de visão.	45
FIGURA 3.6 – Tela de seleção de qual fonte de imagem será utilizada.	46
FIGURA 3.7 – Prendedores na asa. Imagem da câmera XIMEA.	47
FIGURA 3.8 – Prendedores na asa. Imagens da câmera XIMEA com filtros vermelho, verde e azul.	48
FIGURA 3.9 – Janelas de saída para as imagens, processada e com componentes identificados.	49
FIGURA 3.10 – Fluxograma da estrutura funcional do SELA.	50
FIGURA 3.11 – Demonstração do robô KUKA LBR iiwa em cooperação com o operador (KUKA, 2014)	51
FIGURA 3.12 – Conjunto atuador responsável pela selagem acoplado ao robô LBR iiwa.	52

FIGURA 4.1 – Imagens do software de calibração após encontrar os pontos no padrão adotado.	54
FIGURA 4.2 – Aplicação do filtro <code>cvSmooth()</code> à imagem (a). Logo em seguida aplicação da função <code>cvCanny()</code> para identificação de bordas (b).	55
FIGURA 4.3 – Prendedores na asa. Imagem da câmera XIMEA com aplicação de filtros.	56
FIGURA 4.4 – Prendedores na asa. Imagem da câmera XIMEA com a execução do software identificador de prendedores e filetes.	57
FIGURA 4.5 – Janela do programa em execução, ao medir o tempo de processamento para cada <i>frame</i>	58
FIGURA 4.6 – Histograma do canal verde no <i>frame</i> capturado da câmera XIMEA com indicação do percentual de intensidade do pixel de maior valor na imagem.	60
FIGURA 4.7 – Gráfico da taxa de acerto pelo percentual de luminosidade no <i>frame</i> para uma configuração de limite inferior de 50 e superior de 100 no algoritmo de Canny.	60
FIGURA 4.8 – Gráfico da taxa de acerto pelo percentual de luminosidade no <i>frame</i> para uma configuração de limite inferior de 25 e superior de 50 no algoritmo de Canny.	61
FIGURA 4.9 – <i>Frame</i> com 76% de luminosidade e taxa de acerto de 100%. Algoritmo de Canny configurado com limite inferior em 25 e superior em 50.	62
FIGURA 4.10 – <i>Frame</i> com 14% de luminosidade e taxa de acerto de 0%. Algoritmo de Canny configurado com limite inferior em 25 e superior em 50.	62
FIGURA 4.11 – <i>Frame</i> com 93% de luminosidade e taxa de acerto de 0%. Algoritmo de Canny configurado com limite inferior em 25 e superior em 50.	63
FIGURA 4.12 – Variação do limite inferior do algoritmo de Canny.	63
FIGURA 4.13 – Gráfico da taxa de acerto pelo percentual de luminosidade no <i>frame</i> para uma configuração de limite inferior do algoritmo de Canny variando conforme a Figura 4.12 e um relação 2:1 entre limites superior:inferior.	64
FIGURA 4.14 – <i>Frame</i> com 14% de luminosidade e taxa de acerto de 71%. Algoritmo de Canny configurado com limite inferior variando conforme a Figura 4.12 e um relação 2:1 entre limites superior:inferior.	64

FIGURA 4.15 – <i>Frame</i> com 93% de luminosidade e taxa de acerto de 71%. Algoritmo de Canny configurado com limite inferior variando conforme a Figura 4.12 e um relação 2:1 entre limites superior:inferior	65
FIGURA 4.16 –Esquemático do sistema de comunicação via serial entre os computadores de visão e de controle do robô.	65
FIGURA 4.17 –Terminal do software de identificação indicando a abertura da porta serial, o envio dos <i>bytes</i> contendo as coordenadas dos prendedores e o fechamento da porta.	66
FIGURA 4.18 –Terminal recebendo as coordenadas dos prendedores em tempo real.	66
FIGURA A.1 –Imagens de 1 a 6 usadas para calibração, com identificação das arestas.	73
FIGURA A.2 –Imagens de 7 a 12 usadas para calibração, com identificação das arestas.	74
FIGURA A.3 –Imagens de 13 a 18 usadas para calibração, com identificação das arestas.	75
FIGURA A.4 –Dados de saída do programa de calibração.	76
FIGURA A.5 –Gráfico da taxa de acerto pelo percentual de luminosidade no <i>frame</i> para uma configuração de limite inferior de 5 e superior de 10 no algoritmo de Canny.	76
FIGURA A.6 –Gráfico da taxa de acerto pelo percentual de luminosidade no <i>frame</i> para uma configuração de limite inferior de 15 e superior de 30 no algoritmo de Canny.	77
FIGURA A.7 –Gráfico da taxa de acerto pelo percentual de luminosidade no <i>frame</i> para uma configuração de limite inferior de 20 e superior de 40 no algoritmo de Canny.	77
FIGURA A.8 –Gráfico da taxa de acerto pelo percentual de luminosidade no <i>frame</i> para uma configuração de limite inferior de 75 e superior de 150 no algoritmo de Canny.	78
FIGURA A.9 –Desenho 3D do robô KUKA LBR iiwa 14 R820 com o conjunto atuador responsável pela selagem.	79

Lista de Tabelas

TABELA 3.1 – Câmera XIMEA - MQ013CG-E2 (Color) - Especificações Técnicas (XIMEA, 2014)	41
TABELA 3.2 – Especificações técnicas da lente HF16HA-1B (FUJIFILM CORPORA- TION, 2009)	42
TABELA 3.3 – Especificações técnicas do robô LBR iiwa 14 R820 (KUKA, 2014). . .	52

Listas de Abreviaturas e Siglas

ADC	<i>Analog Digital Converter</i> (Conversor analógico-digital)
AME ASA	Projeto de Automação da Montagem Estrutural de Asas
CAD	Computer Aided Design (Desenho Assistido por Computador)
CCD	<i>Charge Coupled Device</i> (Dispositivo de carga acoplada)
CMOS	<i>Complementary Metal Oxide Semiconductor</i> (Semicondutor de metal-óxido complementar)
DCE	<i>Data Communications Equipment</i> (Equipamento de comunicação de dados)
DOF	<i>Degrees Of Freedom</i> (Graus de Liberdade)
DTE	<i>Data Terminal Equipment</i> (Equipamento terminal de dados)
FIRE	Ferramental Inteligente Reconfigurável
FPS	<i>Frames Per Second</i> (Quadros por segundo)
HSV	<i>Hue, Saturation and Value</i> (Matiz, Saturação e Valor)
ITA	Instituto Tecnológico de Aeronáutica
LAM	Laboratório de Automação da Manufatura
MINT	Montagens Internas
OpenCV	<i>Open Source Computer Vision</i> (Visão computacional com código aberto)
PPHT	<i>Progressive Probabilistic Hough Transform</i> (Transformada de Hough Probabilística Progressiva)
RGB	<i>Red, Green and Blue</i> (Vermelho, Verde e Azul)
SELA	Automação do Processo de Selagem
SHT	<i>Standard Hough Transform</i> (Transformada de Hough Padrão)
USB	<i>Universal Serial Bus</i> (Barramento serial universal)
VLSI	<i>Very Large Scale Integration</i> (Integração de larga escala)

Listas de Símbolos

d_0	Distância do objeto à lente
d_i	Distância da lente ao plano da imagem
f	Distância focal
f_x	Distância focal pela largura do pixel
f_y	Distância focal pela altura do pixel
h_0	Altura do objeto
h_i	Altura do objeto projetado na imagem
$I(x, y)$	Intensidade de luz refletida em cada pixel, de coordenadas (x, y)
$I(k, j)$	Intensidade de luz refletida em cada pixel, de coordenadas (k, j)
k_1, k_2, k_3	Parâmetros de distorção radial
m'	Vetor de coordenadas homogêneas do ponto na imagem
M'	Vetor de coordenadas homogêneas do ponto no sistema de coordenadas real
n_x	Largura da janela do filtro gaussiano
n_y	Altura da janela do filtro gaussiano
P_{ident}	Número de prendedores identificados pelo software
P_{ref}	Número de prendedores na imagem
p_1, p_2	Parâmetros de distorção tangencial
r	Raio
s_x	Largura dos elementos sensores
s_y	Altura dos elementos sensores
σ_x	Sigma na direção horizontal
σ_y	Sigma na direção vertical
θ	Ângulo de visão
(ρ, θ)	Parametrização em coordenadas polares
(u_0, v_0)	Coordenadas do ponto principal
(x_0, y_0)	Ponto no plano da imagem

Sumário

1	INTRODUÇÃO	18
1.1	Contextualização e Motivação	18
1.2	Objetivos	20
1.2.1	Objetivo Geral	20
1.2.2	Objetivos Específicos	20
1.3	Justificativas	20
1.4	Recursos e Métodos	21
1.5	Delimitações	21
1.6	Estrutura da Dissertação	22
2	FUNDAMENTAÇÃO TEÓRICA	23
2.1	Câmeras e Imagens Digitais	23
2.1.1	Modelo de Câmera <i>Pinhole</i>	23
2.1.2	Representação de Imagens	27
2.1.3	Sensores de Imagem	30
2.2	Calibração de Câmeras	31
2.3	Plataforma de Desenvolvimento e Ferramentas	32
2.3.1	Filtro para Suavização	33
2.3.2	Método para Detecção de Bordas	34
2.3.3	Transformadas de Hough	35
3	O SISTEMA DE IDENTIFICAÇÃO	40
3.1	Equipamentos para Captura de Imagens	40
3.1.1	Câmera XIMEA	40

3.1.2	Lente FUJINON	41
3.2	Software para Calibração da Câmera	43
3.3	Comunicação Serial	44
3.4	Descrição do Programa de Visão	45
3.4.1	Captura da Imagem (Dados de Entrada)	45
3.4.2	Identificação dos Prendedores	48
3.4.3	Identificação dos Filetes	49
3.4.4	Imagens e Dados de Saída	49
3.5	Contextualização do Trabalho no SELA	50
4	IMPLEMENTAÇÃO E TESTES DO SISTEMA DE VISÃO	53
4.1	Resultados e Análise	53
4.1.1	Calibração	53
4.1.2	Identificação de Prendedores e Filetes	55
4.1.3	Testes de Desempenho	57
4.1.4	Envio de Informações via Serial	65
5	CONCLUSÃO E SUGESTÕES PARA TRABALHOS FUTUROS	68
5.1	Conclusão	68
5.2	Sugestões para Trabalhos Futuros	69
	REFERÊNCIAS	71
	APÊNDICE A – INFORMAÇÕES ADICIONAIS	73
A.1	Calibração	73
A.2	Testes com Variação de Iluminação	76
A.3	Robô SELA	79

1 Introdução

1.1 Contextualização e Motivação

Há algum tempo a automação está presente na indústria. Na maioria dos casos para realizar tarefas simples e/ou repetitivas, com trajetórias pré-definidas e peças previamente posicionadas, por exemplo. Atividades mais complexas, que requerem identificação de objetos para a execução de alguma tarefa, ainda não são automatizadas com tanta facilidade.

Um projeto do ITA em parceria com a Embraer está em andamento. Denominado AME ASA (Automação da Montagem Estrutural de Asas), ele tem por objetivo geral adquirir a capacitação tecnológica em automação de montagem estrutural de asas. O projeto visa adaptar o Laboratório de Automação da Manufatura (LAM) para o desenvolvimento das atividades. Ele foi dividido em quatro sistemas:

- FIRE - Ferramental Inteligente Reconfigurável. Sistema responsável por posicionar os revestimentos de asas;
- MINT - Montagens Internas. Sistema automático de furação e cravação de estruturas aeronáuticas internas ou de ligação;
- REVESTE - Sistema robótico para a execução do processo *one-up-assembly* de asas;
- SELA - Sistema automático responsável pela execução do processo de Selagem de asas.

Como explicitado por (DEVLIEG; FEIKERT, 2008), para justificar os altos custos de automação e para maximizar a eficiência da fabricação, a indústria está se esforçando em desenvolver montagens "*one-up*", onde o produto é montado de uma só vez - perfurado, inspecionado e, em última instância, fixado - sem remoção de rebarbas, limpeza, vedação, por exemplo. Nesse contexto, que o AME ASA está inserido.

O objetivo final do projeto é integrar e testar mecânica, eletrônica e computacionalmente os módulos FIRE, MINT, REVESTE e SELA para trabalharem em conjunto autonomamente, resultando no sistema robotizado *One-up-asa*.



FIGURA 1.1 – Exemplo de sistema de montagem com robô comercial (DEVLIEG; FEIKERT, 2008).

O foco dessa dissertação está no sistema SELA. Nota-se que o processo de selagem é muito importante dentro do ramo aeronáutico, pois é responsável por impedir a passagem de combustível para o meio externo nas aeronaves que possuem tanques de combustível integrais (as estruturas internas das semi-asas são utilizadas como tanques), impermeabilizando o interior das semi-asas com a aplicação de selante sobre prendedores e junções estruturais (fletes).

A tarefa de selagem é realizada de forma manual, demandando um trabalho minucioso e repetitivo para o aplicador de selante. Existe o interesse por parte das fabricantes de aviões na automatização desse processo, com o objetivo de diminuir a exposição do operador a esse trabalho, além de melhorar a eficiência na manufatura, visando qualidade e aumento da velocidade de selagem. Para a consecução dessa tarefa autonomamente, algumas opções de robôs foram estudadas pelos integrantes do projeto, entre elas: humanoides e manipuladores. Câmeras e outros sensores também foram estudados para a aplicação.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo dessa dissertação de mestrado é aplicar técnicas de visão computacional para auxiliar no desenvolvimento de uma solução de automação do processo de selagem em montagem de asas de aviões, identificando e localizando componentes aeronáuticos, tais como filetes e prendedores, para a aplicação de selante.

1.2.2 Objetivos Específicos

- a) Escolher uma plataforma de programação com suporte ao processamento de imagens e funções específicas para aplicações, que envolvem visão computacional;
- b) Encontrar algoritmos e métodos de visão, que auxiliem no reconhecimento de objetos de formatos previamente conhecidos, como prendedores e filetes;
- c) Desenvolver um programa capaz de reconhecer e localizar os componentes aeronáuticos, além de enviar suas coordenadas para o robô que realizará a selagem;
- d) Testar a aplicação desenvolvida em situações adversas, que podem ser encontradas em uma operação real.

1.3 Justificativas

Segundo (SCAFE, 2012), o processo de selagem na indústria aeroespacial é quase 100% executado manualmente. As especificações de selagem, em conjunto com os requisitos de mistura e aplicação do material vedante são escritos para a execução manual da tarefa. Este tem sido o método utilizado no setor aeroespacial por muitos anos. Desenvolver a automatização desse processo demanda a adequação desses guias para a realidade da automação. A tarefa de selagem automática pode ser relativamente simples, quando utilizam-se locais fixos para os componentes envolvidos e asas pré-programadas, porém a sua automatização para torná-lo adaptável aos diferentes tipos de estruturas e posicionamentos levemente distintos, o transforma em um problema bem mais complexo, envolvendo processamento de imagens avançado (visão computacional) e sensores de alta precisão.

Para automatizar o processo de selagem, um diferencial seria a interferência mínima no trabalho como é feito atualmente, ou seja, automatizar o processo sem demandar

grande espaço adicional na linha de montagem ou mudanças significativas no ferramental utilizado.

A aplicação de visão computacional associada a um robô manipulador com dimensões reduzidas e vários graus de liberdade ou um robô humanoide, por exemplo, permite desenvolver uma solução facilmente adaptável ao processo atual de selagem. Isso se dá pelas dimensões reduzidas do sistema e pelo seu alto nível de adaptabilidade.

1.4 Recursos e Métodos

Entre as plataformas de desenvolvimento com recursos para aplicações de visão computacional, as que mais se destacam são: Visual Studio da Microsoft com a programação em linguagem C/C++ e biblioteca de visão computacional OpenCV; LabVIEW da National Instruments com os módulos *NI Vision Development*; e MATLAB & Simulink da MathWorks com as ferramentas do pacote *Computer Vision System Toolbox*.

O algoritmo de Canny (CANNY, 1986), para detecção de bordas, e os algoritmos derivados do método de Hough (HOUGH, 1959), para identificação de formas simples em imagens, são aplicados em diversas funções dentro das plataformas supracitadas que podem auxiliar no programa de visão desenvolvido nesse trabalho.

Foi escolhido desenvolver um software independente do sistema de controle do robô, assim uma comunicação entre os dois sistemas faz-se necessária. Para isso, é preciso lançar mão do uso de uma interface de comunicação entre eles, seja rodando no mesmo computador, ou em máquinas distintas, um exemplo seria a comunicação serial. Considerando as várias ferramentas disponíveis para a realização do software de visão e as possibilidades de comunicação entre a aplicação e o robô, pode-se perceber a viabilidade do desenvolvimento desse trabalho.

Alguns problemas de processamento em tempo real de imagens podem afetar o desempenho do sistema de visão, tais como: o custo computacional para realizar a tarefa, a variação da luminosidade no ambiente de aplicação, a velocidade de processamento de imagem em função da resolução aplicada, entre outros. Testes serão feitos, com o objetivo de identificar problemas e aumentar a robustez do sistema.

1.5 Delimitações

As delimitações consideradas na solução desenvolvida são:

- a) O sistema de visão possui iluminação artificial dedicada, porém pode sofrer algumas

- variações devido à luz ambiente não controlada;
- b) É considerado um painel aberto de asa para a identificação e localização dos componentes aeronáuticos;
 - c) O auxílio de sensores, além da câmera, é necessário para a identificação da pose dos componentes no sistema de coordenadas do mundo real;
 - d) Prendedores circulares e filetes em linha reta são considerados no desenvolvimento da aplicação.

1.6 Estrutura da Dissertação

Esta dissertação é composta por cinco capítulos, incluindo esta introdução, cujas seções descrevem e guiam o desenvolvimento do trabalho. O Capítulo 2 contém o embasamento teórico para a dissertação, desenvolvendo sobre os fundamentos e características dos equipamentos e ferramentas de visão, de forma abrangente. Descreve o funcionamento de câmeras e a necessidade do processo de calibração, além da teoria por trás das principais funções computacionais que serão usadas. No Capítulo 3, as informações sobre a câmera, lente e outros equipamentos e software utilizados serão detalhadas. O dispositivo para comunicação serial, o software para calibração da câmera, a descrição do programa de visão desenvolvido, e ao final, uma contextualização do trabalho dentro do sistema SELA serão vistos nesse capítulo também. O Capítulo 4 explicita os resultados e análises dos testes feitos para validar o funcionamento e a robustez do sistema de visão. Detalhes da calibração da câmera e do programa rodando serão mostrados aqui, além de testes com variação de iluminação e sua influência na taxa de acerto, que também será descrita. No Capítulo 5 são apresentadas as conclusões a partir dos resultados obtidos, as dificuldades encontradas no desenvolvimento do trabalho, e também sugestões para trabalhos futuros. Informações adicionais podem ser vistas no Apêndice dessa dissertação.

2 Fundamentação Teórica

Como citado no capítulo anterior, o trabalho se trata da identificação e localização de componentes aeronáuticos, utilizando algoritmos de processamento de imagem em tempo real. Para atingir esse objetivo, o uso de equipamentos de captura e processamento de imagens, além de software e bibliotecas específicas, faz-se necessário. Esse capítulo serve de base teórica para o trabalho e detalha os fundamentos e características dos equipamentos e ferramentas que foram usados.

2.1 Câmeras e Imagens Digitais

Uma câmera pode ser modelada, simplificadamente, a partir de uma entrada de luz por um orifício do tamanho de um buraco de agulha, onde do lado de dentro está uma placa fotossensível (ou matriz de sensores), que fica marcada devido à incidência da luz. Os detalhes sobre modelagem de câmeras e representação de imagens são encontrados a seguir.

2.1.1 Modelo de Câmera *Pinhole*

A luz refletida em um objeto passa através da lente, que concentra todos os feixes de luz provenientes da cena, em seguida, atinge o plano da imagem localizado na parte de trás da câmera (veja a Figura 2.1), produzindo uma imagem invertida e reduzida do objeto (MUSSABAYEV, 2015).

A relação matemática dos parâmetros mostrados na Figura 2.1 é conhecida como **equação da lente fina** (2.1), ela relaciona as medidas da seguinte forma:

$$\frac{1}{f} = \frac{1}{d_0} + \frac{1}{d_i} \quad (2.1)$$

Onde f é o comprimento focal, d_0 é a distância do objeto à lente e d_i é a distância da lente ao plano da imagem.

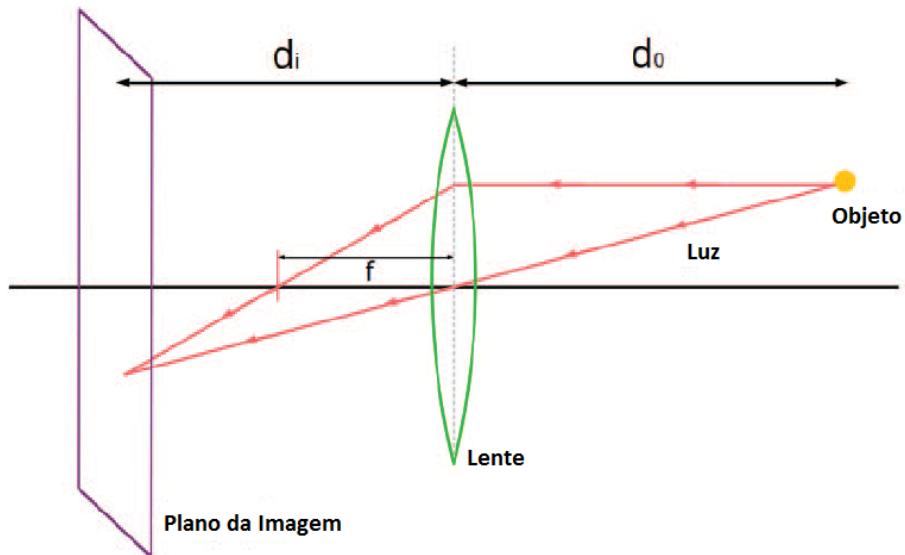


FIGURA 2.1 – Caminho do feixe de luz através da lente na câmera. Figura modificada de (MUSSABAYEV, 2015).

Em visão computacional, tornou-se habitual empregar um modelo simplificado de captura de imagens. A primeira simplificação é ignorar o efeito da lente e substituir a câmera com lente por uma abertura infinitesimal. Em teoria isso não afeta a imagem resultante. Outra simplificação é supor que o plano da imagem é invertido, devido ao fato de que na maioria dos casos têm-se $d_0 \gg d_i$. E por fim, mudar a localização do plano da imagem para antes do centro óptico do sistema, de modo a obter uma imagem vertical e não invertida, como mostra a Figura 2.2. Este modelo é conhecido como modelo de câmera ***pinhole***.

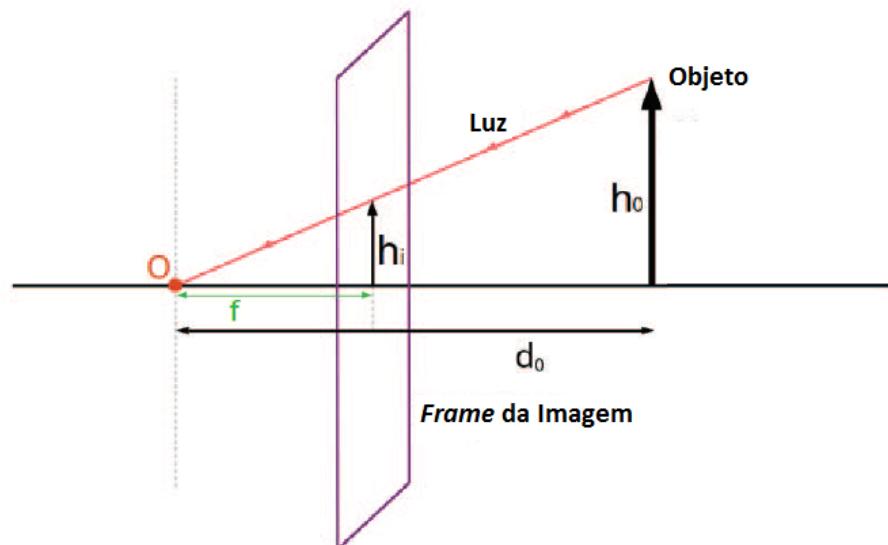


FIGURA 2.2 – *Pinhole*, modelo de câmera simplificado. Figura modificada de (MUSSABAYEV, 2015).

Usando semelhança de triângulos, pode-se deduzir a equação 2.2.

$$h_i = f \frac{h_0}{d_0} \quad (2.2)$$

Onde h_i é a altura do objeto projetado na imagem, d_0 é a distância do objeto ao centro óptico da câmera e h_0 é a altura real do objeto.

Para relacionar um objeto na imagem com a sua localização no mundo real, em três dimensões, é preciso ir um pouco mais a fundo. Considere um sistema de referência tridimensional, cuja origem está localizada no centro óptico do sistema da câmera e o eixo Z (eixo óptico) passa ortogonalmente pelo plano da imagem em um ponto chamado ponto principal. Este sistema é conhecido como **sistema de coordenadas padrão da câmera** (veja a Figura 2.3).

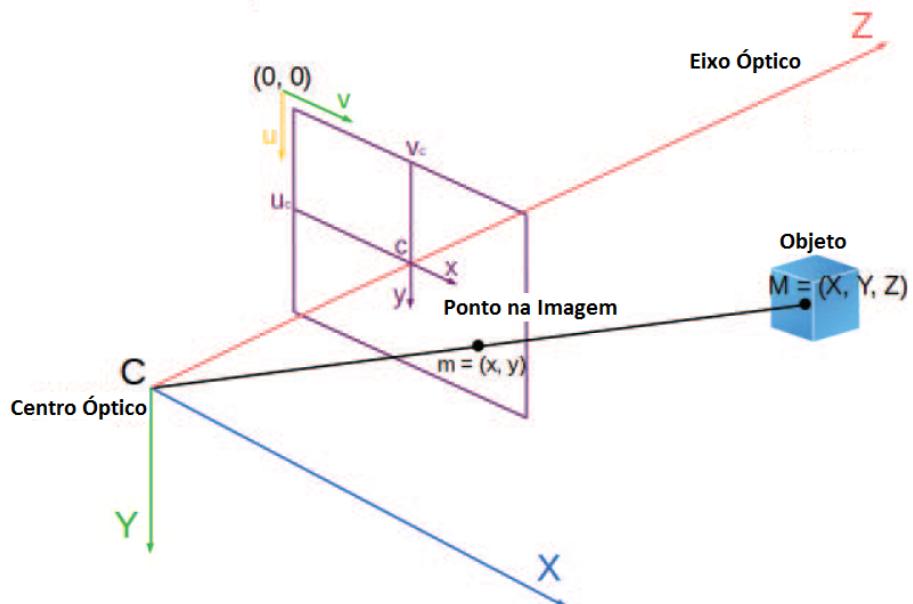


FIGURA 2.3 – Relação entre a imagem e o sistema de coordenadas padrão da câmera. Figura modificada de (MUSSABAYEV, 2015).

Na Figura 2.3, M é um ponto pertencente ao objeto, com coordenadas (X, Y, Z) , relativas ao sistema de coordenadas padrão. Um raio de luz refletido dele intercepta o plano da imagem, gerando o ponto projetado m com coordenadas (x, y) .

Novamente, por semelhança de triângulos, obtêm-se as equações 2.3 e 2.4.

$$x = f \frac{X}{Z} \quad (2.3)$$

$$y = f \frac{Y}{Z} \quad (2.4)$$

Que podem ser reescritas em coordenadas homogêneas como:

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2.5)$$

Onde s é um fator escalar, diferente de zero, que indica que as coordenadas homogêneas foram usadas.

Possuindo a distância focal e a distância em Z , pode-se utilizar a equação 2.5 para calcular quanto um pixel (unidade de medida de (x, y) na imagem projetada) representa no espaço real, e então estimar o tamanho de objetos. A distância focal é um parâmetro que pode ser medido e depende somente da câmera, de acordo com o modelo *pinhole*. Já a distância em Z , conhecida como profundidade, é uma das informações perdidas na conversão 3D para 2D. É possível recuperá-la, utilizando relações encontradas entre duas imagens através da geometria epipolar (geometria da visão estéreo), ou medi-la, usando um sensor de distância infravermelho por exemplo.

A origem no *frame* da imagem está localizada no canto superior esquerdo e todas as medidas são expressas em pixels. Assim, obtém-se as equações 2.6 e 2.7.

$$u = u_0 + \frac{x}{\text{largura do pixel}} \quad (2.6)$$

$$v = v_0 + \frac{y}{\text{altura do pixel}} \quad (2.7)$$

Onde u_0 e v_0 são as coordenadas do ponto principal c , em relação ao canto superior esquerdo, e (u, v) são as coordenadas resultantes da projeção, em pixels.

Combinando as equações 2.3, 2.4, 2.6 e 2.7 obtém-se:

$$Zu = Zu_0 + f_x X \quad (2.8)$$

$$Zv = Zv_0 + f_y Y \quad (2.9)$$

$$f_x = \frac{f}{\text{largura do pixel}} \quad (2.10)$$

$$f_y = \frac{f}{\text{altura do pixel}} \quad (2.11)$$

Em coordenadas homogêneas, o sistema de equações é representado conforme mostra a equação matricial 2.12.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2.12)$$

Na forma reduzida, têm-se:

$$sm' = AM' \quad (2.13)$$

Onde m' e M' são vetores homogêneos de coordenadas dos pontos na imagem e no sistema de coordenadas real, respectivamente. E A é a matriz dos parâmetros intrínsecos da câmera, ou ***matriz da câmera***, ela pode ser obtida com o processo de calibração, que será detalhado na seção 2.2.

2.1.2 Representação de Imagens

Uma foto pode ser definida como uma representação visual em um plano bidimensional de cenas do mundo real, ou seja, é uma projeção ou mapeamento de um espaço 3D num plano 2D. Ao realizar essa projeção uma informação importante é perdida (ou escondida), a profundidade. Conhecendo toda a teoria por trás da captura e dessa transformação é possível realizar uma reconstrução aproximada do espaço 3D, a partir das imagens de uma câmera sob várias perspectivas.(LIMA, 2013)

Para o processamento de imagens, é necessário que a informação esteja em formato digital. Uma imagem no mundo virtual pode ser representada de várias maneiras, a escolha dessa representação deve ser feita para simplificar a aplicação em estudo. As representações mais comuns são:

- a) Preto e branco;
- b) Níveis de cinza;
- c) RGB;
- d) HSV.

Uma imagem em escala de cinza é representada por uma função contínua $I(x, y)$, que indica a intensidade de luz refletida em cada pixel, de coordenadas (x, y) , da imagem. Dada uma câmera que possui uma matriz de $m \times n$ elementos sensores, de largura s_x e altura s_y , as imagens capturada por essa câmera são amostradas com resolução espacial $m \times n$ pixels e função intensidade dada por $I(k, j)$ (veja Figura 2.4), quantizada em intensidade com uma precisão de 2^b , onde b representa o número de *bits* utilizados para armazenar o valor de intensidade. A imagem em preto e branco é um caso particular da representação em escala de cinza, quando se utiliza um *bit* para armazenar o valor de intensidade, assim, tem-se apenas duas possibilidades, branco ou preto.

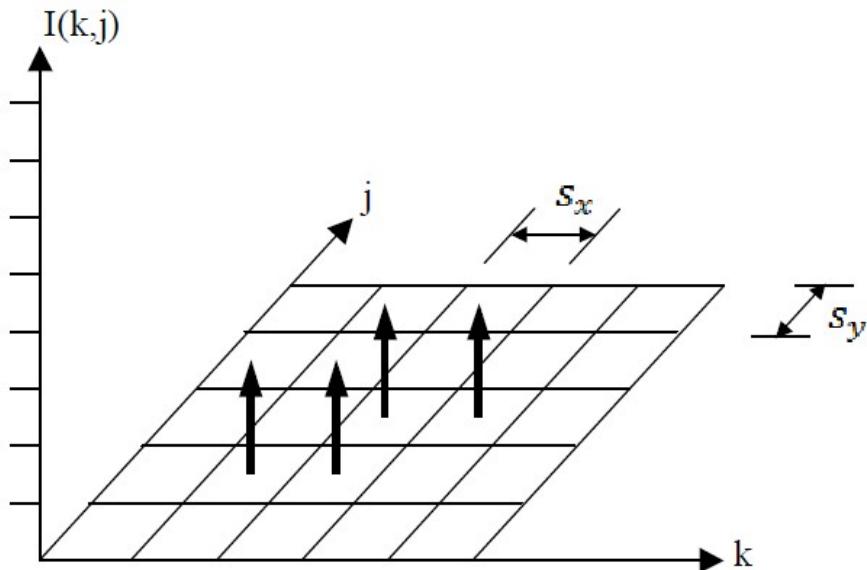


FIGURA 2.4 – Representação da imagem (BEVILAQUA, 2011).

As imagens coloridas (RGB ou HSV) se diferenciam por possuírem função intensidade com três dimensões. No caso da representação em RGB, a base é o sistema visual do ser humano, que através de cones e bastonetes (células localizadas na retina do olho) são capazes de detectar três níveis diferentes de cores, são elas: vermelho, verde e azul (*Red, Green, Blue*). Combinando os níveis identificados para cada uma dessas três cores, variando continuamente a intensidade, obtém-se infinitas tonalidades (GONZALES, 2002). O modelo RGB é um dos métodos mais utilizados, sua representação é um cubo, usando coordenadas RGB em vez de (x, y, z) (SCHELEYER, 2016), como observado na Figura 2.5.

No modelo HSV (*Hue, Saturation, Value*), a cor é representada por três quantidades fundamentais: tonalidade, saturação e o valor de intensidade. A tonalidade identifica a cor, a saturação estabelece a quantidade de cor presente em cada pixel e finalmente, seu valor (SCHELEYER, 2016). O modelo é representado por um subconjunto de um sistema de coordenadas cilíndricas (veja a Figura 2.6). Cada parâmetro é detalhado como:

- **Matriz (*Hue*)** - Especifica uma base de cor através de um ângulo em torno do eixo

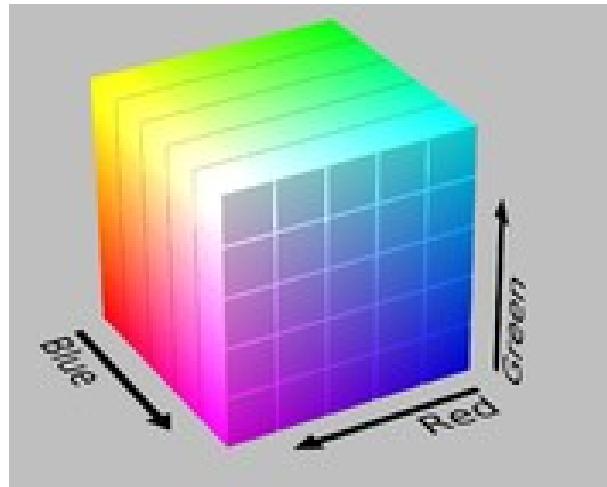


FIGURA 2.5 – Representação da imagem em RGB. Figura extraída de (WIKIPEDIA, 2017).

vertical do cilindro, variando de 0° a 360° ;

- **Saturação (Saturation)** - É a medida ao longo do eixo radial. Especifica a pureza relativa (diluição com a cor branca) da cor, varia entre 0 e 1;
- **Valor (Value)** - Medida ao longo do eixo do cilindro, que possui valor 0 para o preto e 1 para o branco. Especifica a intensidade de luz na cor.

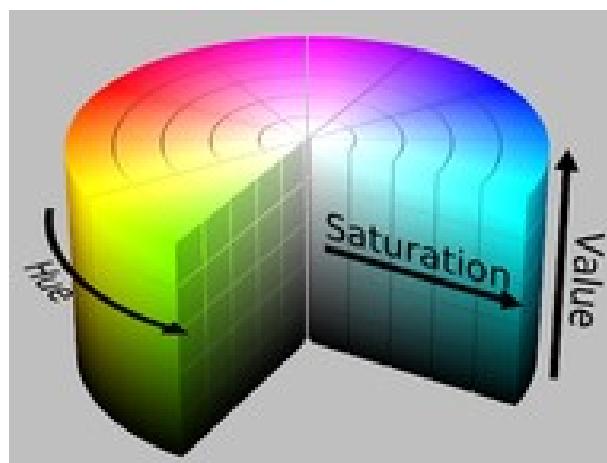


FIGURA 2.6 – Representação da imagem em HSV. Figura extraída de (WIKIPEDIA, 2017).

A representação de imagens em HSV permite a identificação de cores mais diretamente, independente da iluminação. Porém, por facilidade na aplicação de algoritmos de processamento de imagens, a maioria das funções das bibliotecas de software de visão utilizam a representação RGB.

2.1.3 Sensores de Imagem

Adicionando um sensor computacional no plano de formação da imagem, como o mostrado no modelo *pinhole*, a intensidade de luz é transformada em valores digitais através de um conversor analógico-digital e assim a imagem é criada virtualmente. Um sistema de processamento de visão é composto por módulos discretos, um sensor de imagem (normalmente uma câmera com sensor CCD ou CMOS), um sistema de conversão analógico-digital e um sistema de processamento digital (ROSSI, 2012).

A tecnologia dos sensores de imagem CCD e CMOS foi inventada no final dos anos 60. Na época, a performance dos sensores CMOS era limitada pela tecnologia de litografia (técnica associada à deposição de elétrons em circuitos integrados) disponível, o que permitiu que os sensores CCDs dominassem o mercado. Com o passar dos anos e a evolução do processo de produção de circuitos integrados, os sensores CMOS com pixels ativos estão emergindo como uma tecnologia competitiva (ROSSI, 2012).

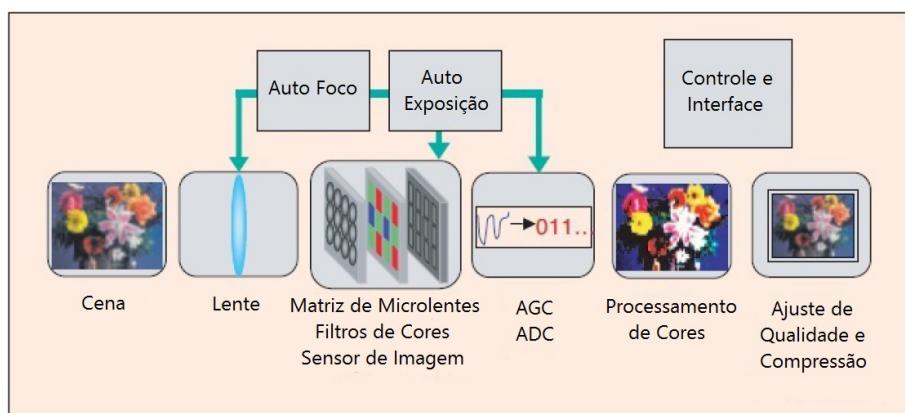


FIGURA 2.7 – Sistema digital de imagem (GAMAL; ELTOUKHY, 2005).

Um diagrama de blocos de um sistema de imagem digital é mostrado na Figura 2.7. Primeiro, a cena é focada no sensor através das lentes. O sensor, composto por uma matriz de pixels, converte a luz incidente na sua superfície em uma matriz de sinais elétricos. Esses sinais são lidos e digitalizados por um conversor analógico-digital (ADC). Um processamento de sinais é empregado para processar as cores, qualidade e compressão da imagem. Outros tipos de processamentos são empregados para realizar o auto-foco, a auto-exposição e controles gerais da câmera. Mesmo cada componente mostrado na Figura 2.7 sendo importante, o sensor é o componente chave para determinar o limite de desempenho final (GAMAL; ELTOUKHY, 2005).

A Figura 2.8 ilustra as principais características dos sensores CCD e CMOS, observa-se que os sensores CCD movem a carga gerada pelos fótons de pixel a pixel até convertê-la em tensão na saída, já os sensores CMOS convertem a carga em tensão dentro de cada pixel (LITWILLER, 2005).

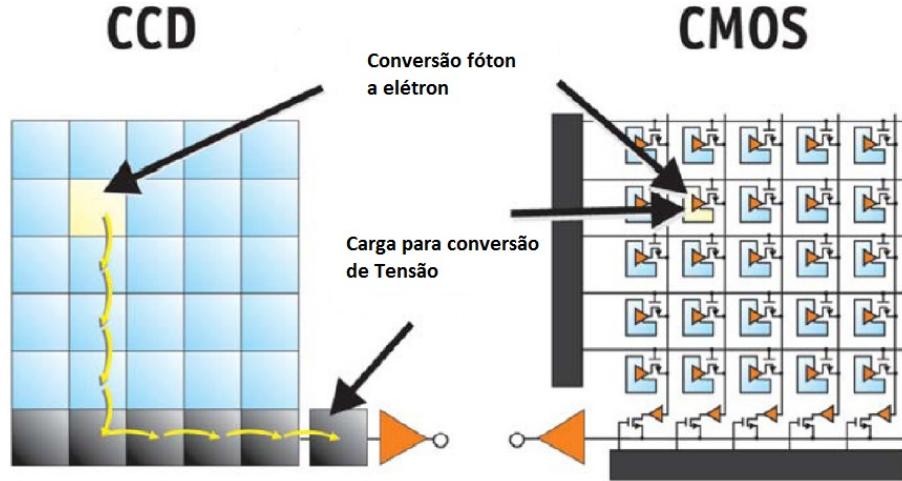


FIGURA 2.8 – Esquemático de funcionamento dos sensores CCD e CMOS (LITWILLER, 2005).

2.2 Calibração de Câmeras

Com o passar dos anos as câmeras e lentes vêm se tornando cada vez mais populares e acessíveis, graças aos processos de fabricação em massa de lentes e sensores. Associadas a essa popularização estão as distorções nas imagens das câmeras, cuja fabricação não garante uma qualidade aceitável para algumas aplicações. Felizmente, essas distorções são constantes, na maioria dos casos, e podem ser corrigidas utilizando algoritmos de calibração e remapeamento. Com a calibração pode-se determinar também a relação entre a unidade natural da câmera (pixel) e uma unidade real (milímetros, por exemplo).

Para encontrar as distorções nas imagens são levados em conta os fatores radial e tangencial, como explicita (DOCS.OPENCV, 2011-2014). A presença da distorção radial é identificada pelo efeito barril ou olho de peixe. Já a distorção tangencial, ocorre devido ao não paralelismo entre a lente e o plano da imagem. O fator radial é obtido através das equações 2.14 e 2.15.

$$x_{corrigido} = x(1 + k_1r^2 + k_2r^4 + k_3r^6) \quad (2.14)$$

$$y_{corrigido} = y(1 + k_1r^2 + k_2r^4 + k_3r^6) \quad (2.15)$$

Assim, para um pixel de coordenadas (x, y) na imagem de entrada, sua posição na imagem de saída corrigida será $(x_{corrigido}, y_{corrigido})$.

Para identificar a distorção tangencial as equações 2.16 e 2.17 são aplicadas.

$$x_{corrigido} = x + [2p_1xy + p_2(r^2 + 2x^2)] \quad (2.16)$$

$$y_{corrigido} = y + [p_1(r^2 + 2y^2) + 2p_2xy] \quad (2.17)$$

Logo, temos cinco parâmetros de distorção, que podem ser representados pela **matriz de distorção** 2.18.

$$D = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \quad (2.18)$$

O processo que determina as matrizes de distorção e da câmera é chamado de **calibração**. O cálculo desses parâmetros é feito usando geometria básica.

2.3 Plataforma de Desenvolvimento e Ferramentas

Dentre as várias plataformas de desenvolvimento para aplicações em visão computacional, a biblioteca OpenCV associada ao ambiente de programação Visual Studio foi escolhida. A OpenCV é uma biblioteca *open source* com ferramentas e funções focadas em processamento de imagens. A sua vantagem em relação às outras ferramentas, para se trabalhar com visão computacional, é sua ampla comunidade. Para a maioria dos problemas que você se deparar, existe alguém que já passou pelo mesmo e o resolveu dentro de fóruns, que podem ser facilmente encontrados na internet. Composta por algoritmos otimizados, suas funções procuram sempre minimizar o custo computacional, uma grande vantagem quando se trabalha com processamento em tempo real. Como detalhado (BRADSKI; KAEHLER, 2008), essa biblioteca pode ser usada para diversas aplicações, dentre elas se destacam:

- a) Operações entre imagens;
- b) Filtros;
- c) Transformações morfológicas;
- d) Calibração de câmeras;
- e) *Tracking*;
- f) Estimação de pose;
- g) Reconhecimento e identificação de faces, gestos e objetos.

A biblioteca OpenCV foi desenvolvida para as linguagens C, C++ e Python, roda nos sistemas operacionais Linux, Windows e Mac OS.

Nesta seção do capítulo serão detalhados os principais métodos e funções aplicados no trabalho.

2.3.1 Filtro para Suavização

Conforme explicita (BRADSKI; KAEHLER, 2008), a suavização, também chamada de desfoque, é uma operação de processamento de imagem simples e usada frequentemente. Há muitas razões para suavizar uma imagem, mas geralmente isso é feito para reduzir ruído ou anomalias da câmera. A biblioteca OpenCV oferece cinco operações diferentes de suavização. Todas elas são aplicadas através da função `cvSmooth()`, que leva a forma desejada de suavização como um argumento.

A função `cvSmooth()`, além dos argumentos de imagem: fonte e destino. Possui quatro parâmetros, denominados: param1, param2, param3 e param4. Os significados desses parâmetros dependem da operação de suavização escolhida, que pode ser:

- a) CV_BLUR – Desfoque simples;
- b) CV_BLUR_NO_SCALE – Desfoque simples sem escala;
- c) CV_MEDIAN – Desfoque mediano;
- d) CV_GAUSSIAN – Desfoque gaussiano;
- e) CV_BILATERAL – Filtro bilateral.

O desfoque gaussiano é o mais utilizado. Esta filtragem é feita convoluindo cada ponto na matriz de entrada com um núcleo gaussiano e, em seguida, somando para produzir a matriz de saída.

Para o desfoque Gaussiano (veja a Figura 2.9), os dois primeiros parâmetros da função `cvSmooth()` são a largura e a altura da janela do filtro, o terceiro parâmetro (opcional) indica o valor sigma (metade da largura na metade do máximo) do núcleo gaussiano. Se o terceiro parâmetro não for especificado, o filtro gaussiano será automaticamente determinado a partir do tamanho da janela usando as equações 2.19 e 2.20:

$$\sigma_x = 0,30 \left(\frac{n_x}{2} - 1 \right) + 0,80 \quad (2.19)$$

$$\sigma_y = 0,30 \left(\frac{n_y}{2} - 1 \right) + 0,80 \quad (2.20)$$

Onde n_x e n_y são o primeiro e o segundo parâmetro, respectivamente.

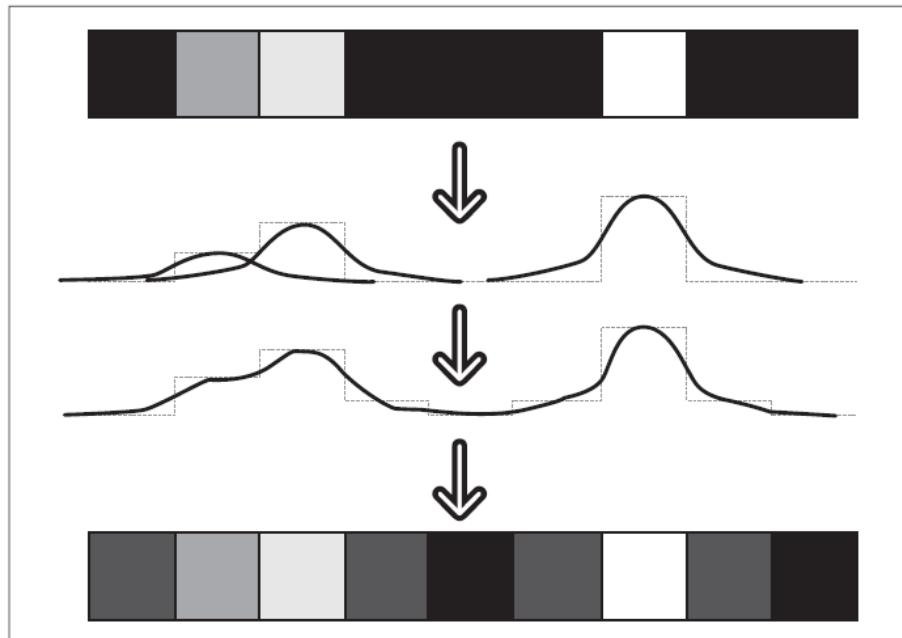


FIGURA 2.9 – Desfoque gaussiano em um vetor de pixels (BRADSKI; KAEHLER, 2008).

O quarto parâmetro é opcional e deve ser fornecido caso deseje que o núcleo seja assimétrico, neste caso, o terceiro e quarto parâmetros serão os valores de sigma nas direções horizontal e vertical, respectivamente. Se esses parâmetros forem dados, mas os dois primeiros forem definidos como 0, então o tamanho da janela será determinado automaticamente a partir do valor de sigma. Os resultados do desfoque gaussiano podem ser vistos na Figura 2.10.



FIGURA 2.10 – Aplicação do desfoque gaussiano (BRADSKI; KAEHLER, 2008).

2.3.2 Método para Detecção de Bordas

De acordo com (BRADSKI; KAEHLER, 2008) o método da função Laplaciano, que pode ser utilizado como um detector de arestas, foi refinado por J. Canny em 1986, criando

assim o Detector de bordas Canny (CANNY, 1986). Uma das principais diferenças entre o algoritmo de Canny e o de Laplace é que no algoritmo de Canny as primeiras derivadas são computadas em x e y , e então combinadas em quatro derivadas direcionais. Os pontos onde estas derivadas direcionais são máximos locais tornam-se candidatos a bordas.

O algoritmo de Canny tenta montar os pixels candidatos a borda em contornos. Estes contornos são formados aplicando um limiar de histerese aos pixels, ou seja, existem dois limites, um superior e um inferior. Se um pixel tem um gradiente maior do que o limite superior, então ele é aceito como um pixel de borda, já se ele estiver abaixo do limite inferior, ele é rejeitado. Caso o gradiente do pixel esteja entre as bordas, então ele será aceito somente se estiver conectado a um pixel que está acima do limite superior. Canny recomendou uma relação de limite superior:inferior entre 2:1 e 3:1. As Figuras 2.11 e 2.12 mostram os resultados da aplicação de `cvCanny()` a uma fotografia usando relações 5:1 e 3:2, respectivamente.

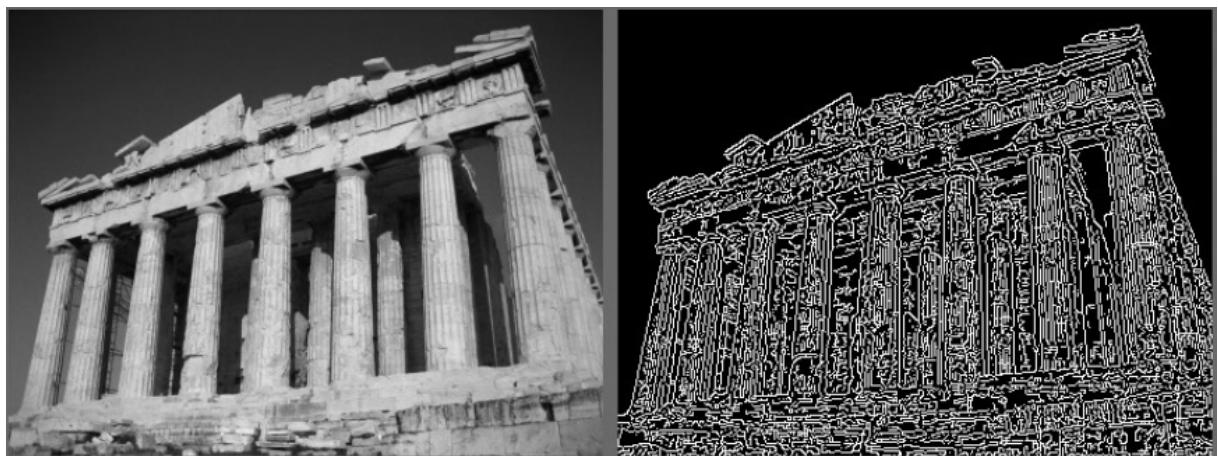


FIGURA 2.11 – Resultados da aplicação do método de detecção de bordas Canny quando os limites superior e inferior foram ajustados para 50 e 10, respectivamente (BRADSKI; KAEHLER, 2008).

A função `cvCanny()` espera uma imagem de entrada, que deve ser em tons de cinza. Os próximos dois argumentos são os limites inferior e superior. O último argumento é uma abertura, esta é usada pelos operadores gradientes de Sobel que são chamados dentro da implementação de `cvCanny()`, assim como no algoritmo de Laplace (BRADSKI; KAEHLER, 2008). A imagem de saída, também será em tons de cinza, porém se parecerá mais com uma imagem booleana, ou seja, em preto e branco.

2.3.3 Transformadas de Hough

Hough desenvolveu essa transformada para ser usada em experimentos físicos (HOUGH, 1959), seu uso em visão computacional foi introduzido por Duda e Hart (DUDA; HART, 1972). A transformada original, é uma maneira relativamente rápida de procurar em uma



FIGURA 2.12 – Resultados da aplicação do método de detecção de bordas Canny quando os limites superior e inferior foram ajustados para 150 e 100, respectivamente (BRADSKI; KAEHLER, 2008).

imagem binária por linhas retas. Com a possibilidade de ser generalizada para casos em que não sejam apenas linhas, ela foi aprimorada para encontrar também círculos e outras formas simples. A seguir estão detalhadas as transformadas utilizadas nesse trabalho, as informações foram extraídas de (BRADSKI; KAEHLER, 2008).

2.3.3.1 Identificação de Linhas

A transformada de linha de Hough teoriza que qualquer ponto em uma imagem binária pode fazer parte de um conjunto de possíveis linhas. Se parametrizarmos cada linha, por uma inclinação a e uma interceptação b , então um ponto na imagem original é transformado em um conjunto de pontos no plano (a, b) , correspondente a todas as linhas que passam por esse ponto (veja a Figura 2.13). Se convertermos cada pixel diferente de zero na imagem de entrada (de coordenadas (x, y)) em um conjunto de pontos na imagem de saída (de coordenadas (a, b)) e somarmos todas essas contribuições, então as linhas na imagem de entrada aparecerão como máximos locais na imagem de saída. Como estamos somando as contribuições de cada ponto, o plano (a, b) é comumente chamado de *plano acumulador*.

A parametrização real utilizada na computação numérica é um pouco diferente. Ela representa cada linha como um ponto em coordenadas polares (ρ, θ) , sendo a linha sugerida a linha que passa pelo ponto indicado, mas perpendicular à radial desde a origem até aquele ponto (veja a Figura 2.14). Essa linha é dada pela equação 2.21.

$$\rho = x \cos \theta + y \sin \theta \quad (2.21)$$

O algoritmo da transformada de Hough do OpenCV retorna os máximos locais no



FIGURA 2.13 – Aplicação da transformada de Hough para linhas. Muitas linhas não esperadas são encontradas (BRADSKI; KAEHLER, 2008).

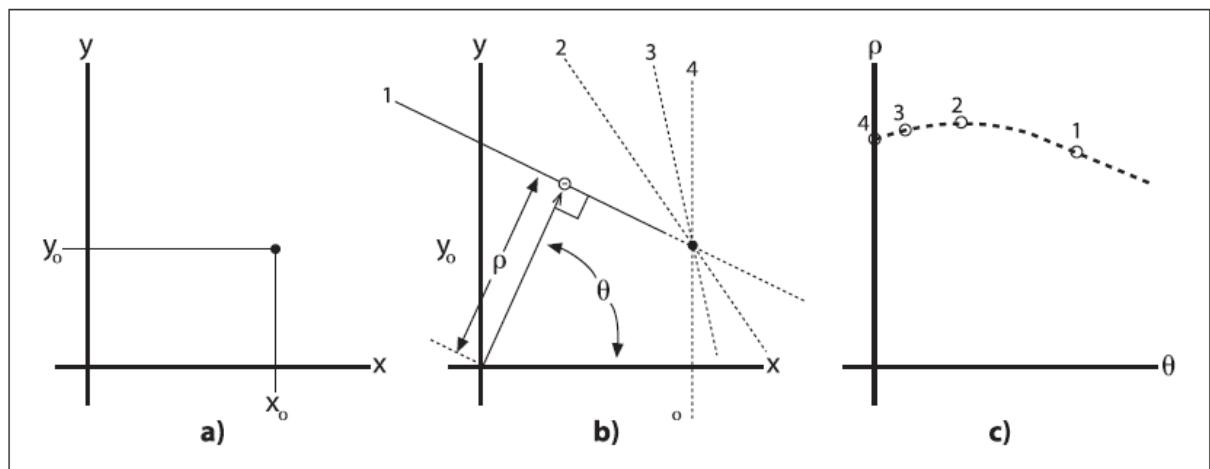


FIGURA 2.14 – Um ponto (x_0, y_0) no plano da imagem (a) implica em várias linhas parametrizadas por diferentes ρ e θ (b), os diferentes pontos (ρ, θ) formados pelo conjunto dessa linhas correspondem a uma curva característica (c) (BRADSKI; KAEHLER, 2008).

plano (ρ, θ) . O OpenCV suporta dois tipos diferentes de transformada de linha de Hough: a Transformada de Hough Padrão (SHT) e a Transformada de Hough Probabilística Progressiva (PPHT). O SHT é o algoritmo que acabamos de ver. O PPHT é uma variação deste algoritmo que, entre outras coisas, calcula uma extensão para linhas individuais além da orientação, como mostrado na Figura 2.15. É probabilístico porque, ao invés de acumular todos os pontos possíveis no plano do acumulador, ele acumula apenas uma fração deles. A ideia é que se o pico vai ser suficientemente alto de qualquer maneira, identificar isso em uma fração do tempo é mais eficiente. O resultado desta conjectura pode ser uma redução substancial no tempo de computação. Ambos os algoritmos são acessados com a mesma função do OpenCV (`cvHoughLines2()`), porém os significados de alguns dos argumentos dependem de qual método está sendo usado.

O primeiro argumento da função é a imagem de entrada, essa deve ser em escala de

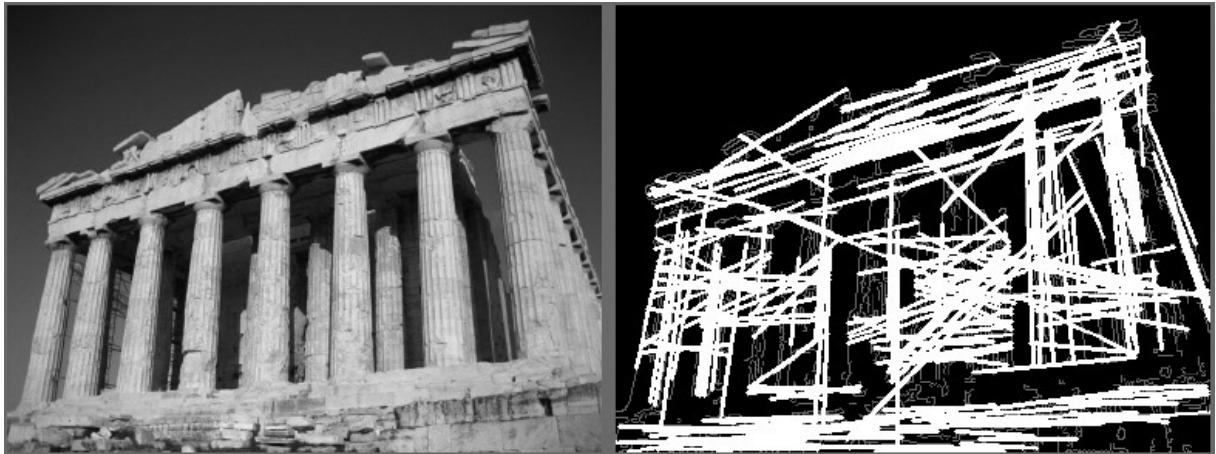


FIGURA 2.15 – Detector de bordas Canny (limite inferior=50 e limite superior=150) aplicado primeiro, com os resultados em cinza, e a transformada de Hough probabilística progressiva aplicada posteriormente, com os resultados sobrepostos em branco. (BRADSKI; KAEHLER, 2008).

cinza, porém é tratada como imagem binária, ou seja, todos os pixels diferentes de zero (preto) são considerados equivalentes. O segundo é um ponteiro, que aponta para um local onde os resultados podem ser armazenados na memória. O próximo argumento é o método, que pode ser:

- a) CV_HOUGH_STANDARD;
- b) CV_HOUGH_PROBABILISTIC;
- c) CV_HOUGH_MULTI_SCALE.

Onde o terceiro método é uma variante de escala múltipla da SHT.

Os dois argumentos seguintes, *rho* e *theta*, definem a resolução desejada para as linhas, isto é, a resolução do plano acumulador. A unidade de *rho* é pixel e a de *theta* é radiano. O plano acumulador pode ser imaginado como um histograma bidimensional com células de dimensões *rho* pixels por *theta* radianos. O sexto argumento (valor limite) é o valor no plano acumulador que deve ser atingido para que o algoritmo registre a identificação de uma linha.

Para o método PPHT os dois últimos parâmetros definem o comprimento mínimo de um segmento de linha que será retornado e a separação entre os segmentos colineares, necessária para que o algoritmo não os junte em um único segmento mais longo, respectivamente.

A função `cvHoughLines2()` retorna um ponteiro, que aponta para uma estrutura de sequência de objetos do OpenCV, `CvSeq`, contendo as linhas encontradas pelo algoritmo.

2.3.3.2 Identificação de Círculos

A transformada de círculo de Hough funciona de maneira análoga à transformada de linha. Porém, usa um método um pouco mais complexo chamado método de gradiente Hough, para evitar grande uso de memória e perda de desempenho. Esse método utiliza as derivadas de Sobel em sua implementação.

A função do OpenCV que aplica o algoritmo da transformada de círculo de Hough é a `cvHoughCircles()`, ela possui argumentos semelhantes à da função de transformada de linha. A imagem de entrada é uma imagem em escala de cinza. O argumento armazenador de círculos será um ponteiro novamente. Os círculos serão transformados em uma sequência do OpenCV e um ponteiro para essa sequência será retornado por `cvHoughCircles()`. O argumento do método deve ser definido como `CV_HOUGH_GRADIENT`.

O quarto parâmetro é a resolução da imagem do acumulador usada. Este parâmetro permite criar um acumulador com uma resolução inferior à da imagem de entrada. Se for definido como 1, então as resoluções serão as mesmas, se definido para um número maior, então a resolução do acumulador será menor por esse fator. O valor do argumento não pode ser menor que 1. O parâmetro seguinte é a distância mínima que deve existir entre os centros de dois círculos para que o algoritmo os considere círculos distintos. O sexto e o sétimo argumentos são o limite de borda (Canny) e o limite do acumulador, respectivamente. Como o identificador de bordas possui dois limites diferentes, quando a função `cvCanny()` é chamada internamente, o primeiro limite (superior) é definido como o valor do parâmetro passado em `cvHoughCircles()` e o segundo limite (inferior) é definido como a metade desse valor. O sétimo parâmetro é aquele usado para limitar o acumulador e é análogo ao argumento de limite da função `cvHoughLines()`. Os dois parâmetros finais são os raios mínimo e máximo dos círculos que serão procurados.

A Figura 2.16 mostra a função `cvHoughCircles()` em funcionamento.

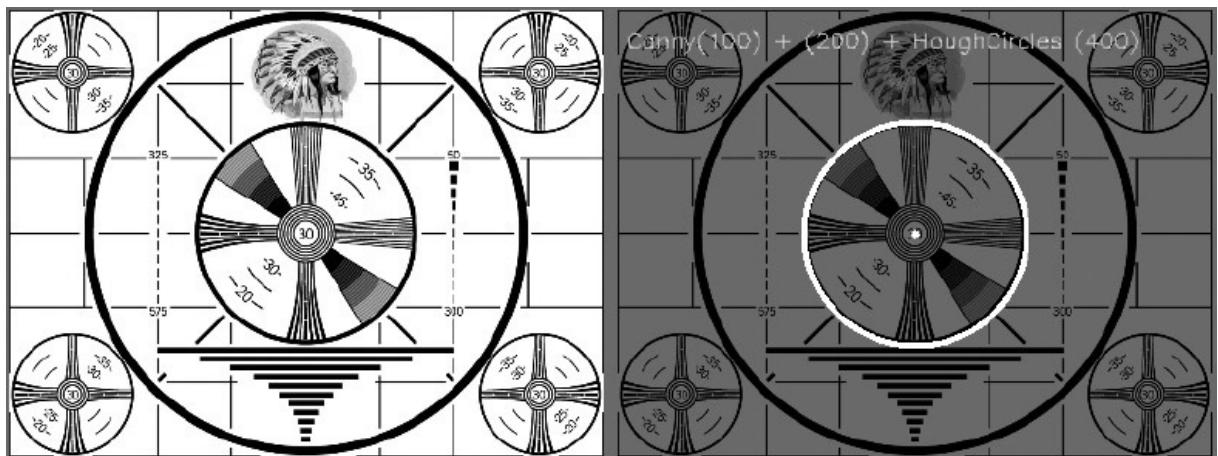


FIGURA 2.16 – Transformada de Hough para círculos encontrando um círculo no padrão de testes (BRADSKI; KAEHLER, 2008).

3 O Sistema de Identificação

Para realizar o processamento de imagens em tempo real foi utilizada a biblioteca de visão computacional *open source*, OpenCV (versão 2.4.10), associada à linguagem de programação C, dentro do ambiente Visual Studio 2013 rodando no Windows 10 em um laptop com processador Intel core i7 e 8GB de memória RAM.

Informações sobre a câmera e a lente utilizadas são encontradas nesse capítulo, além de detalhes sobre outros equipamentos e o código em C desenvolvido. Ao final, uma contextualização do trabalho dentro do SELA é feita, visando deixar claro o papel do sistema de visão computacional aqui desenvolvido dentro do sistema de selagem automático de asas.

3.1 Equipamentos para Captura de Imagens

Para identificar os objetos de interesse, um conjunto formado por uma câmera XIMEA e uma lente FUJINON foi usado. As seções a seguir detalham cada um dos componentes.

3.1.1 Câmera XIMEA

Uma câmera da série *USB3 Vision* fabricada pela XIMEA, mais precisamente a câmera da família xiQ, modelo MQ013CG-E2, ilustrada pela Figura 3.1, foi a escolhida para o sistema de visão.

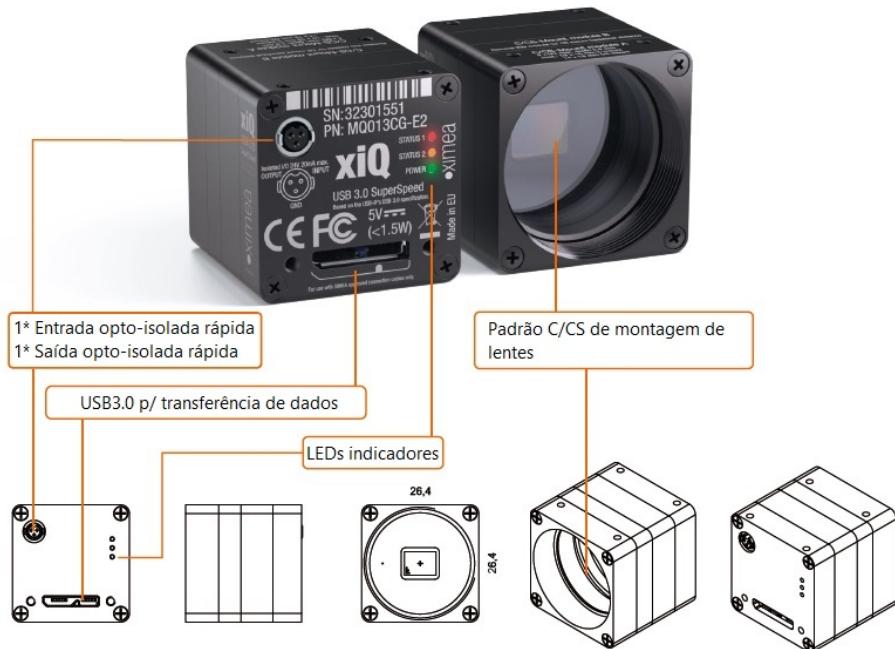


FIGURA 3.1 – Câmera XIMEA - MQ013CG-E2 (Color) - Descrição (XIMEA, 2014).

A família xiQ se trata de câmeras com sensores CMOS de última geração, com leitura *global shutter* (leitura quadro a quadro). Mais detalhadamente, as especificações dessa câmera podem ser encontradas na tabela 3.1.

TABELA 3.1 – Câmera XIMEA - MQ013CG-E2 (Color) - Especificações Técnicas (XIMEA, 2014)

Sensor	Resolução	Tamanho Pixel [μm]	ADC [bits]	Abertura	Tamanho Sensor [mm]	FPS
E2V EV76C560	1280x1024	5,3	10	1/1,8"	6,9 x 5,5	60

3.1.2 Lente FUJINON

O modelo HF16HA-1B (conforme mostrado na Figura 3.2) é da família FUJINON CCTV de lentes, desenvolvida para aplicações em automação da manufatura e visão de máquina. Seu projeto permite o uso em câmeras com resolução de até 1.5 megapixel. Projetada para baixa distorção, essa lente garante fidelidade na imagem de entrada. Ela é compacta, leve, robusta e possui trava para o ajuste de foco e abertura da iris. Essa lente suporta aplicações em ambientes com vibração (FUJIFILM CORPORATION, 2009).

HF16HA-1B

FIXED **1.5**
Mega **MANUAL** **C-mt** **METAL** **F1.4**

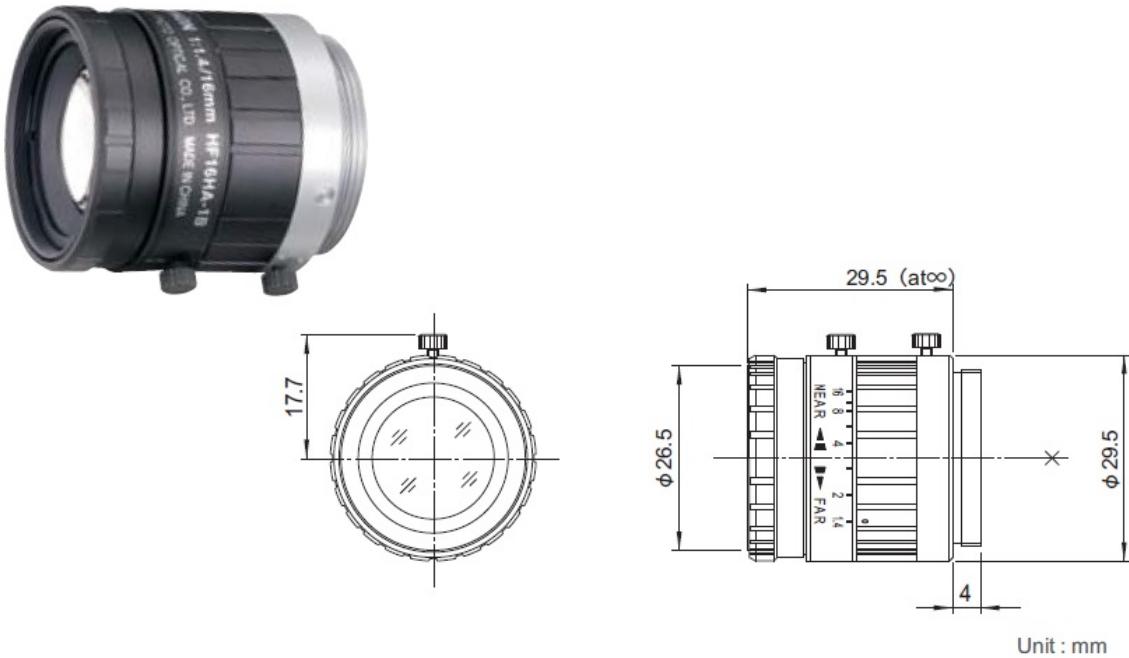


FIGURA 3.2 – Lente FUJINON - HF16HA-1B (FUJIFILM CORPORATION, 2009)

A lente se acopla perfeitamente à câmera XIMEA escolhida, com bom ângulo de visão para a aplicação em estudo e distorção máxima de aproximadamente 1%, de acordo com o fabricante.

TABELA 3.2 – Especificações técnicas da lente HF16HA-1B (FUJIFILM CORPORATION, 2009)

	HF16HA-1B
Distância Focal (mm)	16
Abertura	F1.4 - F16
Operação Foco/Iris	Manual
Ângulo de visão (HxV)	2/3"/ 30°45' x 23°18' 1/2"/ 22°37' x 17°04' 1/3"/ 17°04' x 12°50'
Foco	∞ - 0.1
Massa (g)	45

As especificações técnicas da lente FUJINON adotada estão detalhadas no manual do fabricante (FUJIFILM CORPORATION, 2009), a tabela 3.2 mostra os dados principais.

3.2 Software para Calibração da Câmera

Em (DOCS.OPENCV, 2011-2014), pode-se encontrar um programa desenvolvido em C++/OpenCV para calibração de câmeras. Ele nos permite escolher um entre três tipos de padrões diferentes para calibrar o conjunto câmera-lente, são eles:

- a) Tabuleiro de xadrez;
- b) Padrão de círculo simétrico;
- c) Padrão de círculo assimétrico.

Para calibrar a câmera é necessário tirar algumas fotos de um desses padrões com a câmera e deixar que o software os encontre. Cada padrão identificado resulta em uma equação. Para resolver o sistema de equações é preciso de um número mínimo de fotos para que se tenham equações linearmente independentes suficientes. Basicamente o programa de calibração realiza os seguintes passos:

- 1) Coleta das imagens pela câmera diretamente, por um vídeo ou por uma lista de imagens;
- 2) Determina a matriz de distorção;
- 3) Determina a matriz da câmera;
- 4) Salva os resultados em um arquivo XML;
- 5) Calcula o erro de re-projeção médio.

A Figura 3.3 mostra o programa coletando os pontos de interesse na imagem de uma câmera com distorção radial (A) em comparação com outra imagem (B) da mesma câmera após realizar o processo de correção utilizando as matrizes obtidas na calibração. O tabuleiro de xadrez foi adotado como padrão na imagem exemplo e na aplicação desenvolvida.

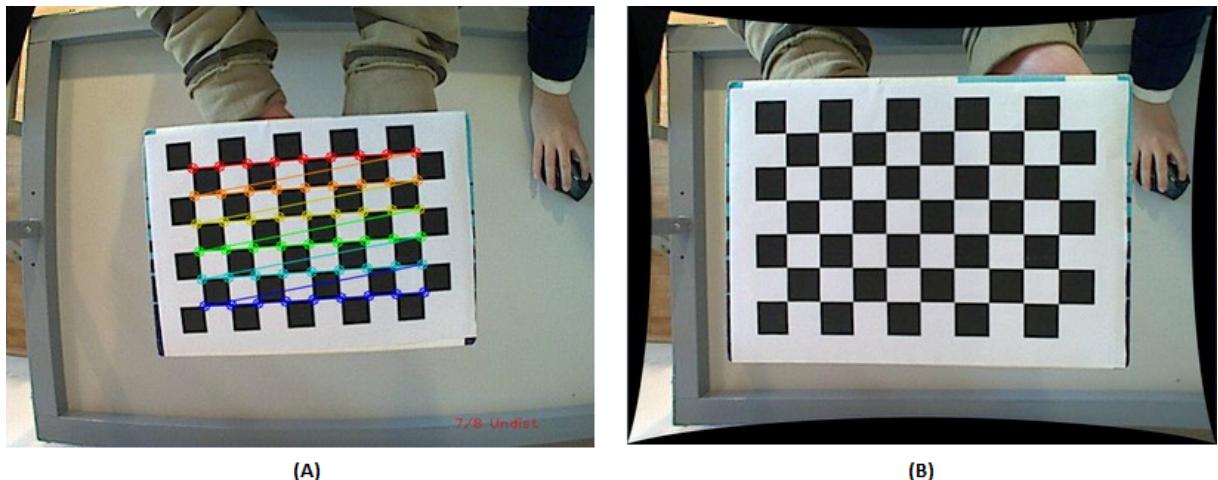


FIGURA 3.3 – (A)- Imagem distorcida antes da calibração. (B)- Imagem sem distorção após a calibração e correção. (DOCS.OPENCV, 2011-2014)

3.3 Comunicação Serial

Com o objetivo de permitir flexibilidade na utilização de plataformas diferentes para controlar o robô e para processar as imagens coletadas, foi desenvolvida uma *feature* capaz de enviar os dados obtidos (as coordenadas dos prendedores, por exemplo) em tempo real para um outro computador, esse responsável pelo controle do robô, realimentando-o com as informações do sistema de visão.

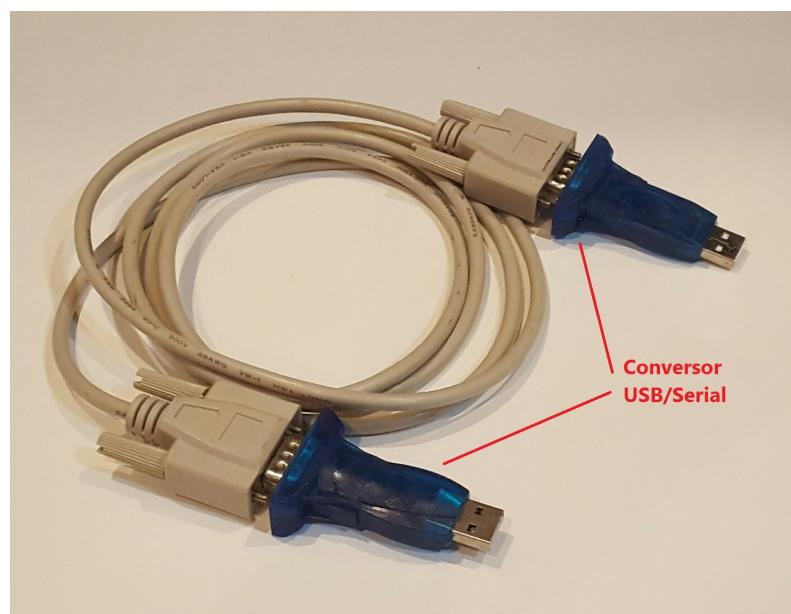


FIGURA 3.4 – O cabo *NULL Modem*, conhecido também como cabo *Crossover*, é usado para permitir que dois dispositivos seriais *Data Terminal Equipment* (DTE) comuniquem entre si, sem usar um modem ou um dispositivo *Data Communications Equipment* (DCE).

A comunicação serial com protocolo RS232 foi escolhida devido à sua ampla utilização em aplicações diversas. Para garantir ainda mais flexibilidade na comunicação, como poucos computadores possuem uma porta serial já incorporada, foram adicionados conversores USB/Serial em cada uma das pontas de um cabo *NULL Modem* (veja a Figura 3.4). Isso permite que dois computadores com interface USB se comuniquem facilmente via serial. Com um simples terminal de leitura, você pode visualizar as informações processadas pelo software de visão.

3.4 Descrição do Programa de Visão

Essa seção detalha o código desenvolvido em linguagem C, responsável por identificar os componentes aeronáuticos. O fluxograma da Figura 3.5 mostra as principais etapas envolvidas e a sequência em que elas são executadas no código.

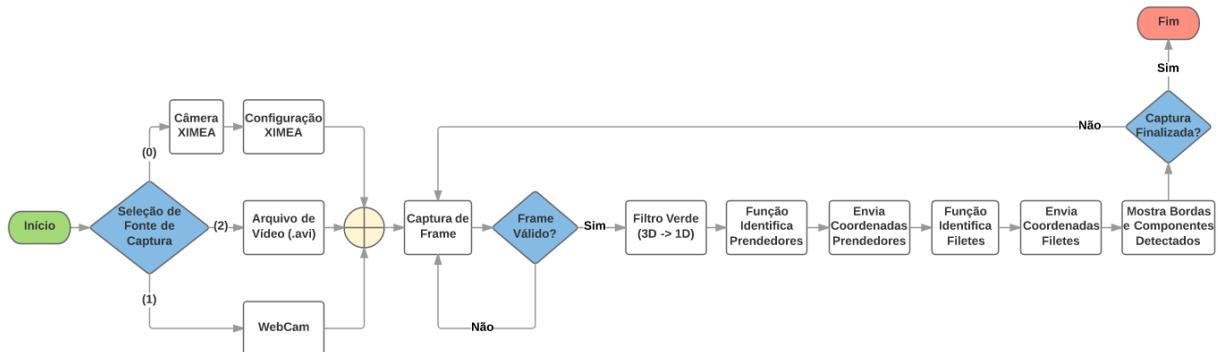


FIGURA 3.5 – Fluxograma do código desenvolvido para o sistema de visão.

Os módulos contendo as funções da biblioteca OpenCV que são utilizadas no programa foram incluídos no código, são eles: `imgproc_c.h`, `highgui_c.h` e `core_c.h`, além da biblioteca `xiApi.h` do pacote de software da câmera XIMEA e as bibliotecas `stdio.h` e `windows.h`, que contêm as principais funções padrões da linguagem C incluindo as utilizadas para a comunicação serial.

3.4.1 Captura da Imagem (Dados de Entrada)

Foi criada uma função para que o usuário selecione qual a fonte de captura de imagem será utilizada. Como a Figura 3.6 mostra, essa seleção é feita entrando com um número de 0 a 2 no terminal. A opção 0 é o programa de identificação de fato, enquanto as outras opções servem para testes, ela corresponde à seleção da câmera XIMEA com uma resolução de 640x512 pixels. Escolhendo a opção 1 o usuário seleciona a *webcam* padrão instalada no computador, com uma resolução de 640x480 pixels. Por fim, selecionando

a opção 2, o usuário pode rodar um vídeo, gravado na resolução 640x512 pixels, com a câmera XIMEA de preferência, para assim simular a identificação dos prendedores e fletes, sem precisar estar realmente no local realizando a filmagem. As opções 1 e 2 foram criadas para simular as situações reais e assim poder desenvolver o programa sem precisar estar necessariamente no laboratório.

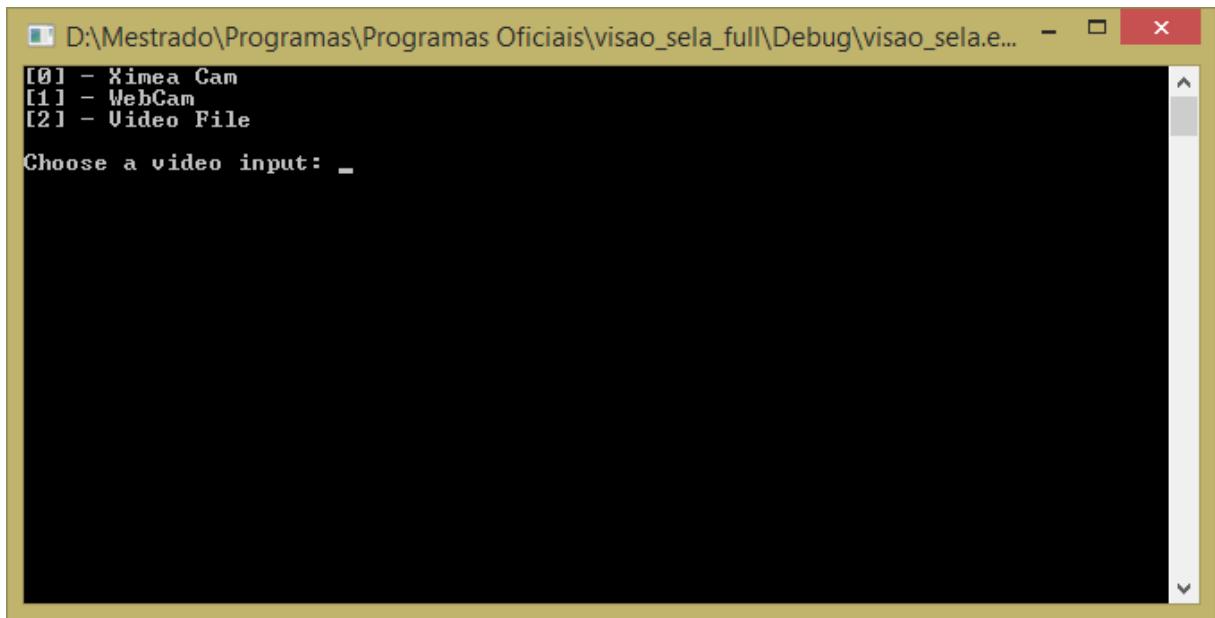


FIGURA 3.6 – Tela de seleção de qual fonte de imagem será utilizada.

A Figura 3.7 mostra a imagem de um vídeo, gravado com a câmera XIMEA, do protótipo utilizado para estudo sendo executado no ambiente do programa de visão desenvolvido.

Quando a câmera XIMEA é selecionada, o programa precisa chamar o procedimento de configuração da câmera e preparar para a captura dos *frames*. Logo após, são criadas as matrizes da câmera e de distorção, ambas obtidas após o procedimento de calibração. Essas matrizes são usadas para corrigir a imagem em cada novo *frame* que chega, essa correção é feita utilizando a função `cvUndistort2()` da biblioteca OpenCV.

Se o *frame* capturado não for vazio, o programa realiza os procedimentos de identificação posteriores. Essa condição foi desenvolvida pois, em caso de perda de *frames*, o programa apresenta erro na execução.

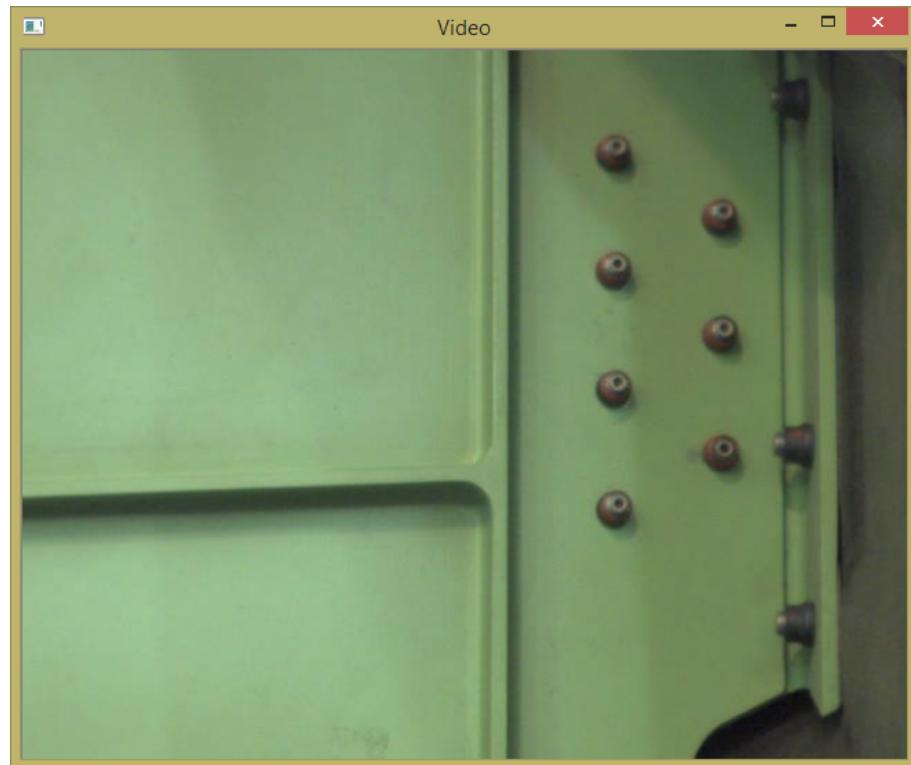


FIGURA 3.7 – Prendedores na asa. Imagem da câmera XIMEA.

A representação de imagens em RGB foi escolhida, pois a maioria das funções e métodos esperam imagens nesse formato, como parâmetros de entrada. Foram criadas três imagens com dimensões um (escala de cinza) de oito *bits* para cada uma das componentes do *frame*. A função `cvSplit()` que separa o *frame* de três dimensões (colorido, RGB) em três componentes de cores (vermelha, verde e azul) foi utilizada em seguida para obter cada imagem separada.

Observou-se que a componente com maior contraste era a com filtro verde, conforme mostrado na Figura 3.8, ela foi a usada nos processos seguintes de detecção, dado que essa característica na imagem facilita a detecção das bordas e por conseguinte, dos prendedores e filetes. Uma filtragem mais complexa poderia ter sido aplicada, levando em consideração as três componentes RGB, porém seria oneroso para o desempenho do programa e a utilização da componente verde foi suficiente para essa aplicação em específico.

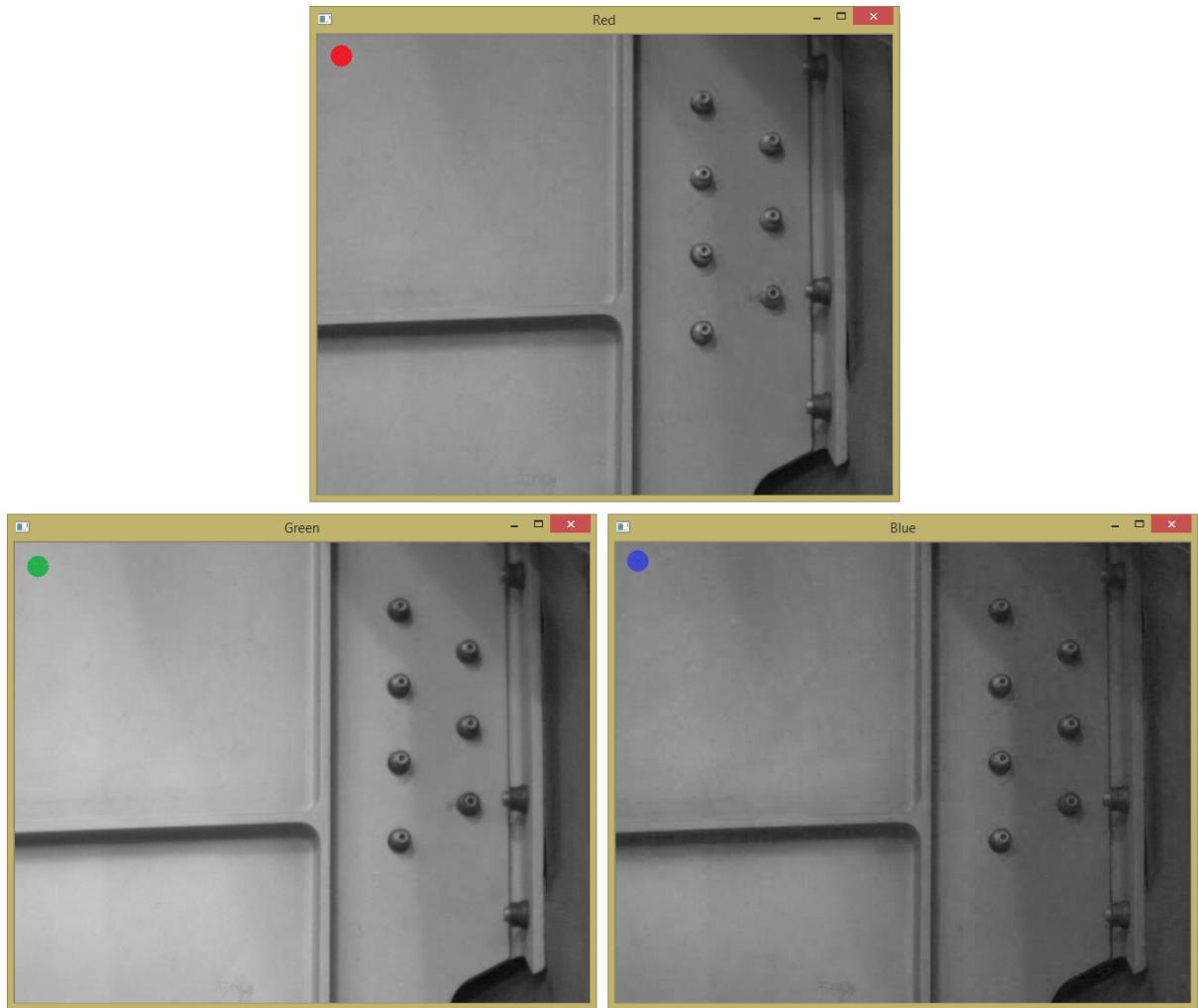


FIGURA 3.8 – Prendedores na asa. Imagens da câmera XIMEA com filtros vermelho, verde e azul.

3.4.2 Identificação dos Prendedores

Foi criada uma função para encontrar os prendedores na imagem atual que é chamada a cada novo *frame* capturado da câmera (ou do vídeo). Essa função retorna a imagem (*frame*) com os círculos desenhados onde os prendedores foram encontrados, além das coordenadas dos centros de cada círculo. O diâmetro de cada círculo reflete o tamanho do prendedor.

Primeiramente, a função aplica o filtro `cvSmooth()` na imagem em escala de cinza recebida como parâmetro. Logo em seguida, o método `cvCanny()` é aplicado na imagem realçando todas as bordas da imagem. Como resultado tem-se uma imagem binária, onde as bordas identificadas estão em branco e o resto da imagem em preto.

Já com a imagem previamente processada é realizada a chamada da função `cvHoughCircles()`, que encontra os círculos e retorna as coordenadas do centro e o raio de cada

um. Em seguida todos os círculos encontrados e as coordenadas dos seus centros são desenhados no *frame* original de três dimensões (RGB).

3.4.3 Identificação dos Filetes

Para a identificação dos filetes em cada *frame* da imagem foi criada também uma função específica, que recebe como entrada a imagem em escala de cinza já processada pela função que encontra os prendedores e a imagem colorida já com os prendedores identificados, ela retorna a imagem colorida com as retas desenhadas onde os possíveis filetes foram encontrados, juntamente com as coordenadas dos pontos de início e fim de cada filete.

A função que encontra as linhas (possíveis filetes) na imagem é a `cvHoughLines2()`, ela identifica as retas e retorna dois pontos para cada uma, um de início e outro de fim. Logo após a identificação, as linhas encontradas na imagem em escala de cinza e as coordenadas dos pontos de início e fim de cada uma são desenhadas na imagem colorida.

É importante ressaltar que, nessa função de identificação de filetes, não foi necessário aplicar novamente o filtro Smooth nem a detecção de bordas Canny, pois as imagens de entrada já herdam esses tratamentos na imagem da função de detecção de prendedores.

3.4.4 Imagens e Dados de Saída

A imagem em escala de cinza processada é mostrada na mesma janela que as barras de ajustes dos filtros utilizados, conforme visto na Figura 3.9. Já a imagem colorida original, com os prendedores e filetes encontrados, é mostrada em outra janela.

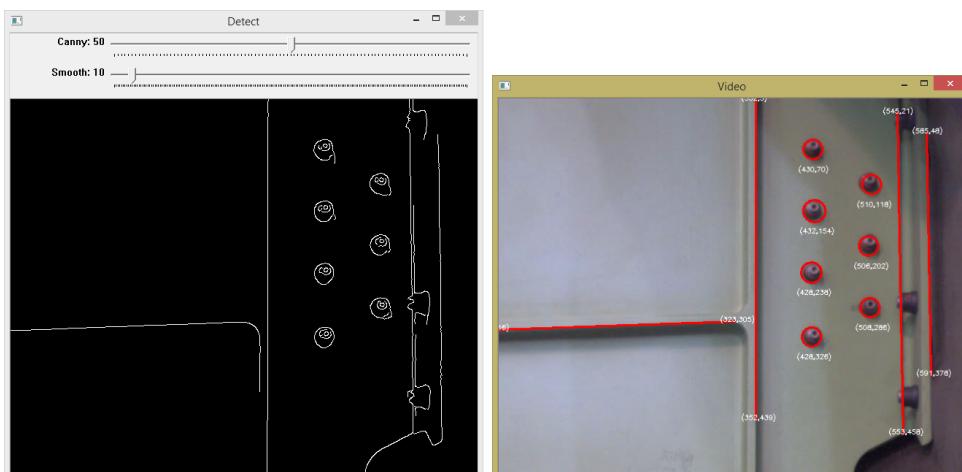


FIGURA 3.9 – Janelas de saída para as imagens, processada e com componentes identificados.

O programa envia as coordenadas dos centros dos prendedores e das extremidades dos filetes retos via serial com a função `serial_comm()`, criada no programa. Elas servem para realimentar o sistema de controle do robô que realiza a tarefa de selagem. É importante ressaltar, que o envio das coordenadas dos filetes foi desativado no decorrer dos testes, maiores detalhes são expostos no próximo capítulo.

3.5 Contextualização do Trabalho no SELA

O escopo do sistema SELA dentro do projeto AME ASA é a realização de forma automática do processo de selagem de prendedores e filetes nos painéis de asas.

Dentro da estrutura funcional do sistema, esse trabalho está inserido entre as fases de localizar painel e posicionar efetuador de selagem, como mostra a Figura 3.10. A tarefa de localizar prendedor ou filete é abordada nessa pesquisa, porém o programa pode ser adaptado para auxiliar nos processos de localização do painel e posicionamento do efetuador de selagem.

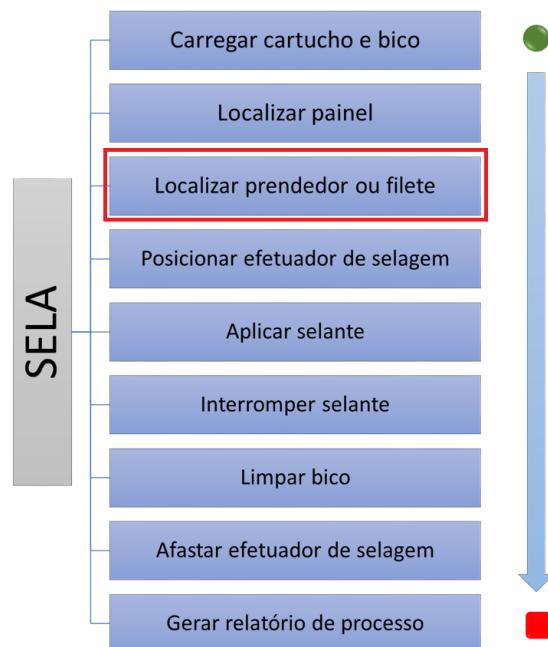


FIGURA 3.10 – Fluxograma da estrutura funcional do SELA.

Para realizar as etapas do processo indicadas no fluxograma, o robô é na maioria delas a ferramenta mais importante. O sistema de visão é o que guiará os principais movimentos dele. Escolher um robô que melhor atende aos requisitos do sistema foi uma das atividades do SELA.

No início havia várias possibilidades de robôs para atuarem no SELA, entre eles huma-

noides e manipuladores. Após um estudo, onde foram ponderadas várias características do robô, tais como: carga útil, capacidade de reproduzir trajetórias, peso, resistência ao movimento, maturidade do fornecedor, facilidade de programação, comunicação com protocolo Ethernet, adaptação de *end effector*, suporte técnico, entre outras, o robô manipulador escolhido foi o LBR iiwa da fabricante KUKA (KUKA, 2014). LBR significa "Leichtbauroboter" (robô leve, em alemão) e iiwa é a sigla para "Assistente de Trabalho Industrial Inteligente" em inglês. Esse é um robô manipulador de sete graus de liberdade produzido em série, altamente sensível, permitindo que ser humano e robô realizem tarefas com estreita cooperação, como pode ser visto na Figura 3.11. O iiwa está disponível em duas versões com capacidades de carga útil de 7 e 14 quilogramas, o maior será utilizado no SELA, denominado KUKA LBR iiwa 14 R820.

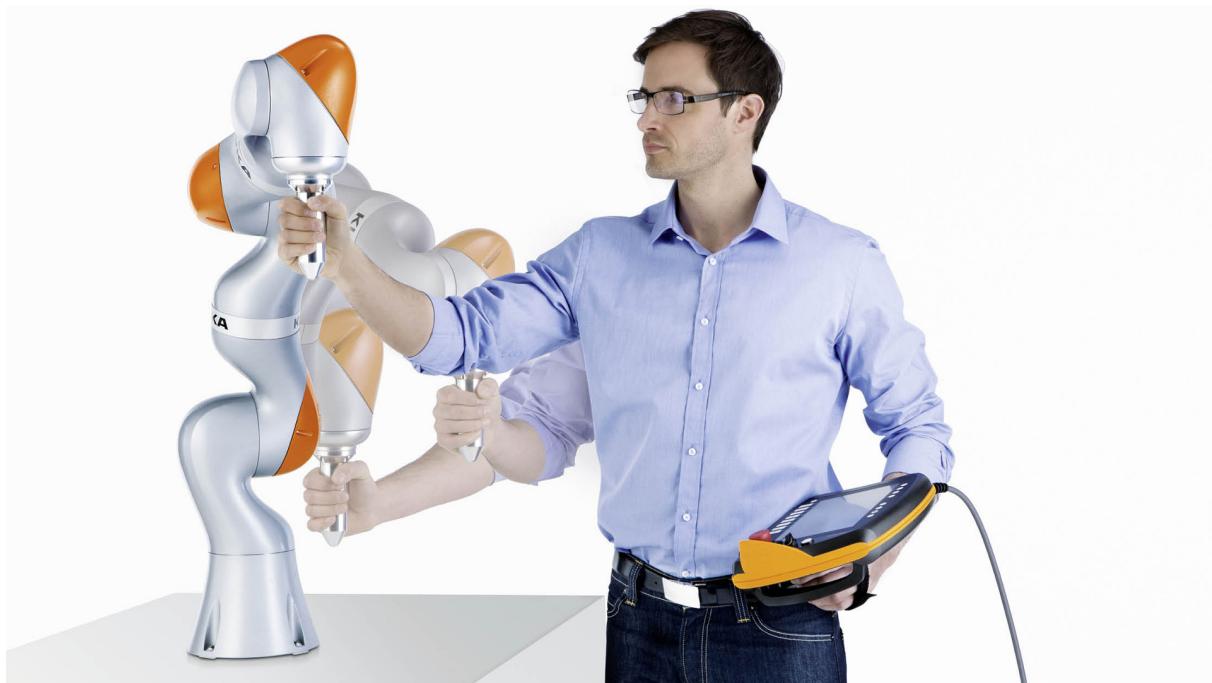


FIGURA 3.11 – Demonstração do robô KUKA LBR iiwa em cooperação com o operador (KUKA, 2014).

A tabela 3.3 mostra as especificações técnicas do robô. Devido aos seus sete graus de liberdade pode alcançar locais de difícil acesso.

TABELA 3.3 – Especificações técnicas do robô LBR iiwa 14 R820 (KUKA, 2014).

	LBR iiwa 14 R820
Carga Útil	14 kg
Máximo Alcance	820 mm
Número de Eixos (DOF)	7
Repetibilidade de Posição	$\pm 0,15$ mm
Peso	30 kg
Temperatura Ambiente	+5 a +45 °C
Classe de Proteção	IP 54

A tarefa de selagem deve ser efetuada por um *end effector* associado à extremidade do robô. Ele é necessário para acomodar as principais ferramentas e sensores utilizados.

Um esboço inicial do *end effector* do robô foi projetado no computador. Como mostra a Figura 3.12 o conjunto é composto por três sensores de distância infra-vermelho, a câmera XIMEA e o sistema de aplicação de selante. Uma imagem com o robô iiwa acoplado ao *end effector* pode ser encontrada no Apêndice A.3.

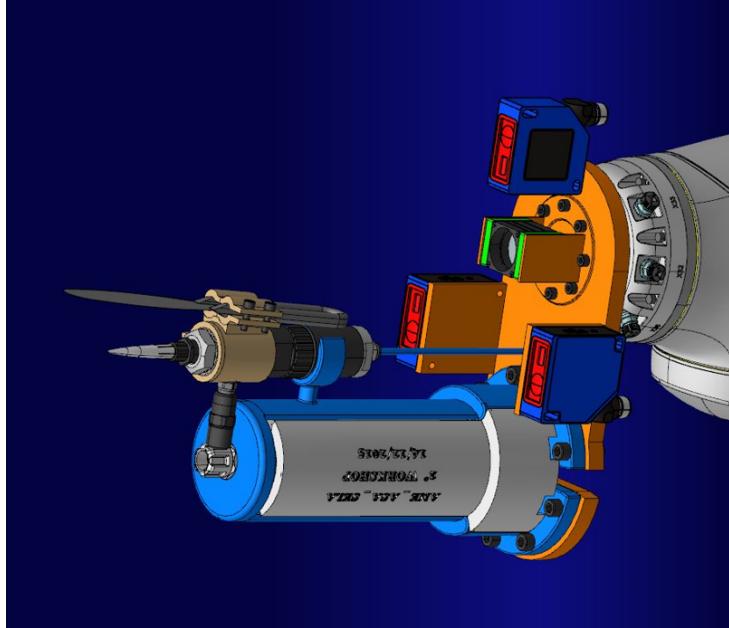


FIGURA 3.12 – Conjunto atuador responsável pela selagem acoplado ao robô LBR iiwa.

O software de visão pode ser executado no mesmo computador que controla os movimentos do robô, ou em um computador diferente. Nesse último caso, a interface serial já criada para o envio das informações obtidas deve ser utilizada. No primeiro, será preciso rodar um software de interface que fará a comunicação em tempo real entre os programas.

4 Implementação e Testes do Sistema de Visão

Nesse capítulo encontram-se todos os resultados obtidos a partir da aplicação do desenvolvimento citado no capítulo anterior para atingir os objetivos desse trabalho. Os detalhes da calibração da câmera e do programa em funcionamento são mostrados aqui. Além de testes com variação de iluminação e sua influência na taxa de acerto, que também é detalhada.

4.1 Resultados e Análise

A calibração da câmera foi feita para garantir uma imagem sem distorções para ser processada e analisada. Foram utilizadas fotos tiradas com a câmera XIMEA para realizar todo o procedimento. Em seguida, os dados da calibração realimentaram o software de identificação de prendedores e filetes, para corrigir possíveis defeitos.

Foi feita uma gravação de vídeo, com a câmera XIMEA, do protótipo do laboratório. Esse vídeo foi utilizado para testar o funcionamento do programa. Os resultados apresentados nesse capítulo são imagens do programa nas diversas fases de testes.

4.1.1 Calibração

Foram utilizadas dezoito imagens da câmera com resolução de 640x512 pixels. Uma amostra do programa de calibração identificando as quinas no padrão do tabuleiro de xadrez pode ser vista na Figura 4.1. Mais detalhes sobre os resultados do procedimento de calibração podem ser encontrados no Apêndice A.1.

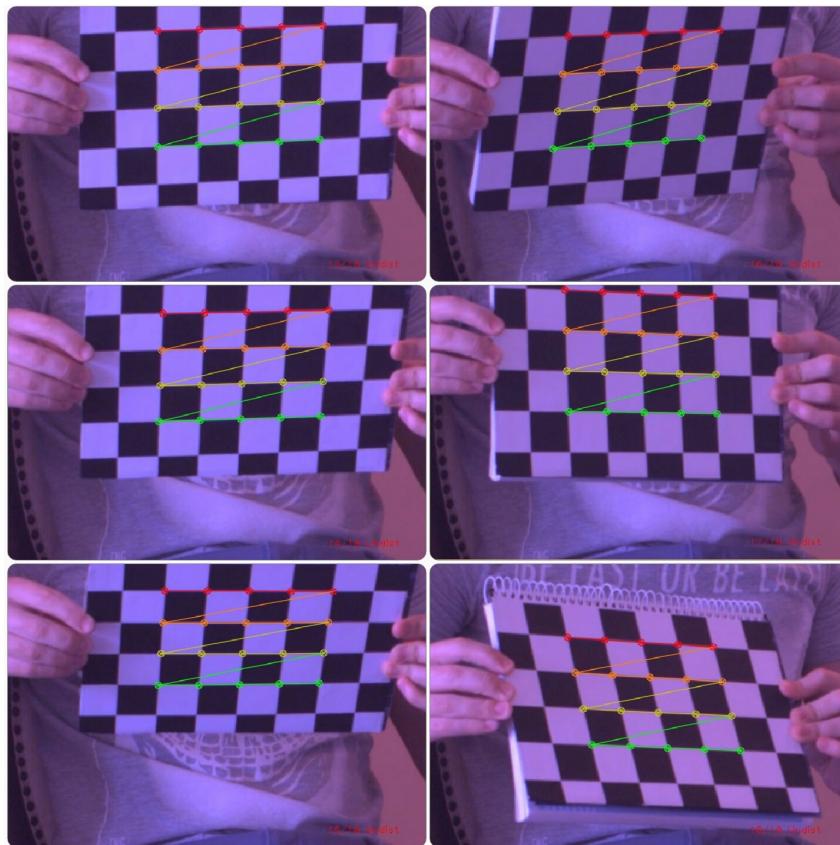


FIGURA 4.1 – Imagens do software de calibração após encontrar os pontos no padrão adotado.

A calibração foi realizada e um erro de re-projeção médio de 0,361 foi obtido, isso significa que na média um ponto está deslocado 0,361 pixels de onde deveria estar caso não houvesse distorções. Quando se divide esse valor pela resolução da imagem encontramos um erro médio inferior à 0,1%, bem abaixo da distorção máxima de 1%, que consta no manual do fabricante da lente, como citado no capítulo anterior.

A matriz de distorção (4.1) e a matriz da câmera (4.2) foram obtidas.

$$D = \begin{bmatrix} -0,64462 & 14,88971 & 0 & 0 & 122,22013 \end{bmatrix} \quad (4.1)$$

$$A = \begin{bmatrix} 3105,17546 & 0 & 319,50000 & 0 \\ 0 & 3105,17546 & 255,50000 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.2)$$

Observou-se que os coeficientes de distorção p_1 e p_2 (veja a equação 2.18) encontrados, referentes à distorção tangencial, são nulos, confirmando os dados do fabricante da lente que indica apenas a distorção radial.

As matrizes de distorção e da câmera foram inseridas como parâmetros na função `cvUndistort2()` da biblioteca OpenCV, para se obter a imagem corrigida. Na prática, não se observou diferenças a "olho nu" entre a imagem da câmera e a imagem corrigida, isso se deu pela alta qualidade da câmera e da lente utilizadas. Mesmo com a falta de variação, a correção foi mantida para cada *frame* capturado da câmera para garantir maior qualidade na estimativa de posição dos objetos de estudo. Retirar essa correção pode ser uma opção viável para melhorar o desempenho do programa.

4.1.2 Identificação de Prendedores e Filetes

Como já descrito no capítulo anterior o filtro para suavização `cvSmooth()` foi utilizado inicialmente. Os argumentos da função foram: o método do desfoque gaussiano, uma janela de filtro de 5x5 pixels, e terceiro e quarto parâmetros (*sigma*) iguais, devido à simetria do núcleo (janela) do filtro. A melhor configuração observada foi para *sigma* igual a 1.

A função responsável pela detecção de bordas `cvCanny()` foi alimentada com a imagem já suavizada e uma relação de limite superior:inferior de 2:1, que se mostrou melhor para a detecção, com o limite inferior em 50 e o superior em 100.

Os resultados da aplicação das funções de suavização e detecção de borda são mostrados pela Figura 4.2. Como pode-se observar, após os tratamentos consecutivos, os prendedores e arestas se destacam facilmente.

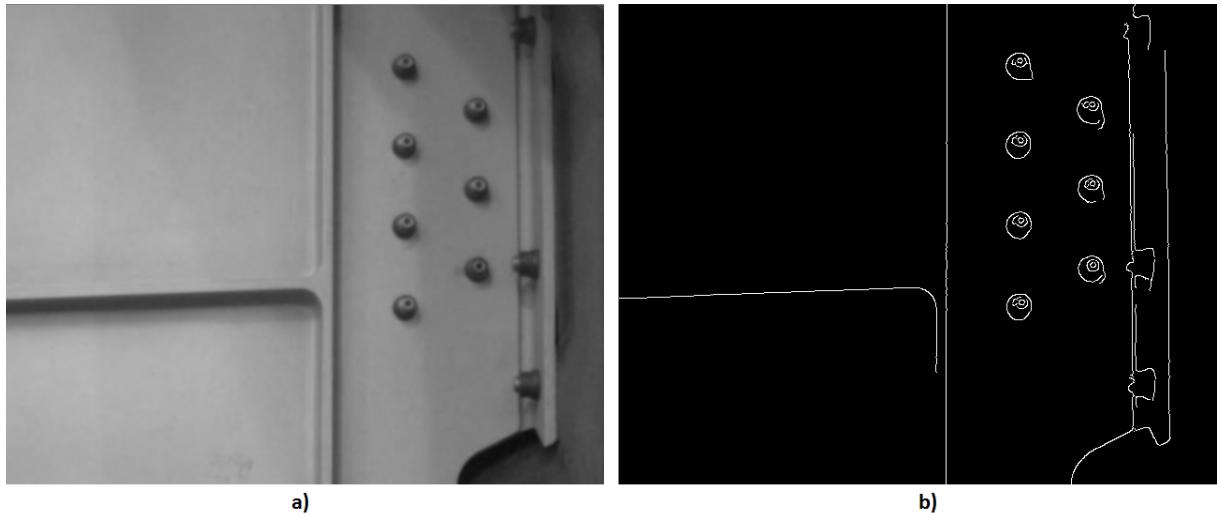


FIGURA 4.2 – Aplicação do filtro `cvSmooth()` à imagem (a). Logo em seguida aplicação da função `cvCanny()` para identificação de bordas (b).

Os ajustes dos parâmetros *sigma* para a função `cvSmooth()` e limite inferior e superior para a função `cvCanny()` foram realizados empiricamente através das barras de ajuste

Canny e *Smooth* mostradas na Figura 4.3. Os valores das barras de ajuste observados são os padrões.

Para a função de suavização o valor da barra *Smooth* é dividido por dez para entrar como parâmetro sigma. Assim, tem-se sigma igual a 1 de acordo com o valor observado na Figura 4.3.

O valor *Canny* de 50 se refere ao limite inferior do método de detecção de bordas. Dada a relação de 2:1 entre limite superior:inferior, tem-se limite superior de 100.



FIGURA 4.3 – Prendedores na asa. Imagem da câmera XIMEA com aplicação de filtros.

Após os detalhes da imagem serem destacados, os prendedores e arestas no caso, a aplicação das funções `cvHoughCircles()` e `cvHoughLines2()` fez-se necessária para identificar a posição exata no plano da imagem desses elementos e suas dimensões. A primeira função, aproxima os prendedores por círculos de diâmetros delimitados pelas bordas (em branco) na imagem de entrada. A segunda, aproxima os filetes por retas encontradas na mesma imagem.

A Figura 4.4 mostra a imagem original da câmera (um *frame*) com os prendedores e filetes destacados em vermelho logo após as identificações serem feitas. Em branco vemos as coordenadas dos centros dos prendedores abaixo de cada círculo e também as

coordenadas dos dois pontos que definem um possível filete nas extremidades da reta.

É importante citar que a identificação de filetes é um pouco mais complexa que a de prendedores, pois é preciso definir onde ele inicia e onde termina, além da detecção de falsos filetes como cantos e detalhes estruturais por exemplo, que não precisam ser selados e podem ser identificados erroneamente. Na Figura 4.4 vários possíveis filetes são mostrados, porém só o mais à direita, destacado pelo retângulo em amarelo, que seria um candidato, pois é uma junção entre duas partes, todos os outros identificados são reforços estruturais ou cantos que fazem parte de uma só estrutura sólida. Uma sugestão seria utilizar a informação de localização dos filetes do avião virtual (CAD) associado à câmera ou utilizar duas câmeras para fornecer a informação de profundidade.

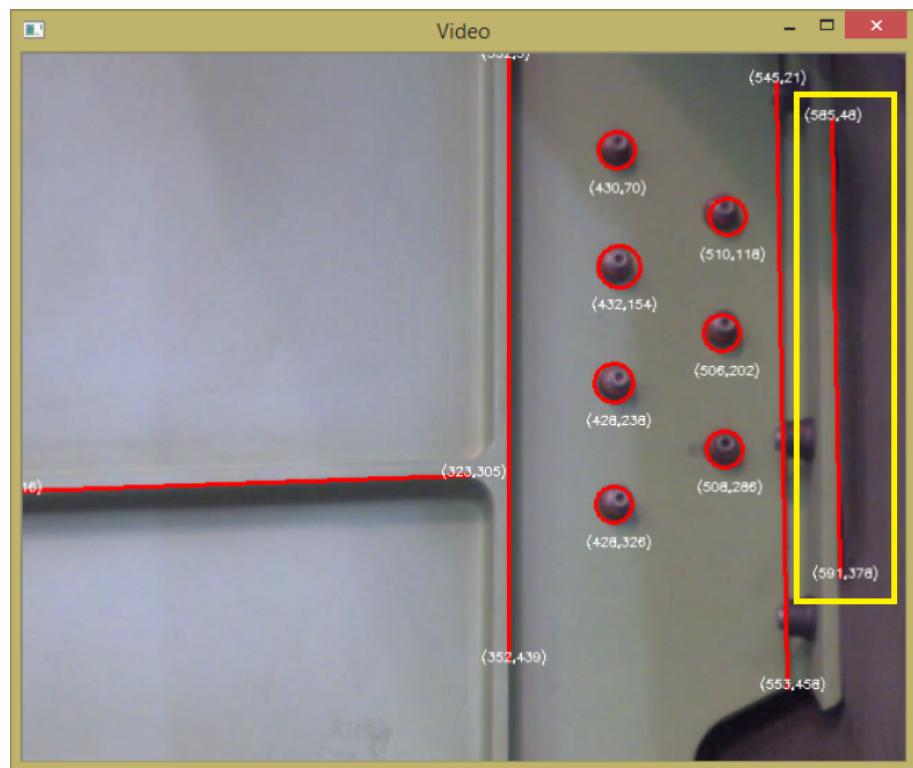


FIGURA 4.4 – Prendedores na asa. Imagem da câmera XIMEA com a execução do software identificador de prendedores e filetes.

4.1.3 Testes de Desempenho

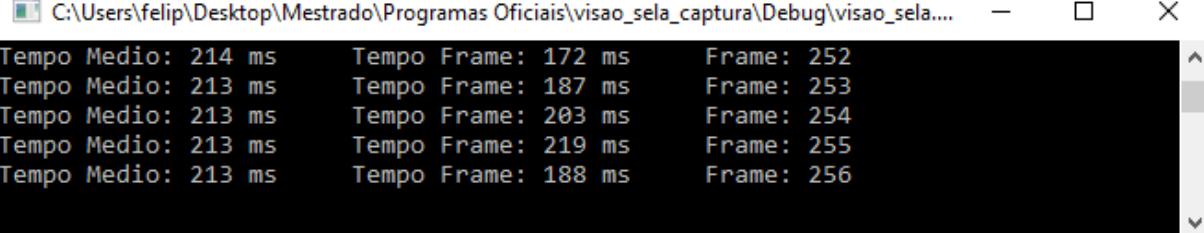
Para avaliar o desempenho do software desenvolvido na tarefa de identificação, mediuse o tempo de execução de cada ciclo de processamento e foi criada uma função que retorna um indicador de taxa de acerto no reconhecimento de prendedores. Realizou-se testes para avaliar a influência da variação de luminosidade nessa taxa. Muitos outros fatores podem influenciar na identificação, como por exemplo: sujeira, óleo, sombra, etc. Os testes realizados focaram na variação da luminosidade, pois se trata do fator mais

relevante do ponto de vista de visão computacional.

4.1.3.1 Tempo de Execução

Para cada *frame* é executado um ciclo de processamento do programa. Com o objetivo de medir o tempo gasto para processar todas as informações durante um ciclo completo, desde a captura do *frame* até a exibição da imagem com os componentes detectados, utilizou-se a função `GetTickCount()` da biblioteca `windows.h` para obter os instantes de tempo inicial e final, e assim determinar o tempo gasto para processar cada *frame*.

A Figura 4.5 mostra o programa executando as medidas de tempo, o **Tempo Frame** é o tempo de execução do código para o *frame* de índice **Frame**, já o **Tempo Médio** é a soma de todos os tempos de execução dividida pela quantidade de *frames* medidos.



```
C:\Users\felip\Desktop\Mestrado\Programas Oficiais\visao_sela_captura\Debug\visao_sela.... - X
Tempo Médio: 214 ms      Tempo Frame: 172 ms      Frame: 252
Tempo Médio: 213 ms      Tempo Frame: 187 ms      Frame: 253
Tempo Médio: 213 ms      Tempo Frame: 203 ms      Frame: 254
Tempo Médio: 213 ms      Tempo Frame: 219 ms      Frame: 255
Tempo Médio: 213 ms      Tempo Frame: 188 ms      Frame: 256
```

FIGURA 4.5 – Janela do programa em execução, ao medir o tempo de processamento para cada *frame*.

O tempo médio de processamento observado foi de 213 milissegundos. Isso permite que a câmera seja configurada com uma taxa de aquisição de quatro FPS e que o robô manipulador se mova com uma velocidade condizente com essa aquisição.

4.1.3.2 Taxa de Acerto

A avaliação desenvolvida aqui não se aplica aos filetes. O foco desse indicador foi avaliar a identificação de prendedores, dado que o algoritmo de reconhecimento de filetes não se mostrou eficiente sem o uso de equipamentos de suporte ao sistema de visão.

Tendo como referência o número de prendedores em posição de serem identificados na imagem e como dado em avaliação o número de prendedores reconhecidos pelo algoritmo, a taxa de acerto se dá pela equação 4.3, em porcentagem de prendedores certos encontrados:

$$Acerto(P_{ident}) = 100 \times \frac{P_{ref} - |P_{ident} - P_{ref}|}{P_{ref}} \quad (4.3)$$

Onde P_{ref} é o número de prendedores na imagem e P_{ident} é o número de prendedores encontrados pelo algoritmo de identificação. Para o caso da taxa de acerto menor que

zero, é atribuído o valor zero.

A função de taxa de acerto foi pensada de forma a pontuar os prendedores corretos encontrados e penalizar a identificação de falsos prendedores. Aplicando a função ao caso da Figura 4.4 por exemplo, tem-se: $P_{ref} = 7$ e $P_{ident} = 7$, assim $Acerto(7) = 100\%$. Caso o número de prendedores encontrados fosse igual a oito ou seis, a taxa de acerto seria de 85%, aproximadamente, indicando que não identificar um prendedor ou identificar um prendedor errado diminui o desempenho do algoritmo em 15% para esse caso.

4.1.3.3 Robustez à Variação de Iluminação

O sistema SELA prevê a iluminação artificial dos painéis, o que nos garante um nível mínimo de luz para o sistema de visão, porém sabe-se que hangares de fabricação normalmente não possuem isolamento para a entrada de luz do ambiente externo. Logo, avaliar o desempenho do sistema para variações de iluminação na imagem faz-se necessário, com o objetivo de garantir uma robustez mínima à variação desse parâmetro.

Dada a dificuldade encontrada em variar continuamente a iluminação do ambiente em que o painel protótipo se encontra no laboratório, optou-se primeiramente por variar a abertura da iris da lente que pode diminuir ou aumentar a luz na imagem, porém foi observado que o efeito artificial (via software) de clarear ou escurecer a imagem é muito bem aplicado nesse caso, pois a influencia da luz ambiente na captura é homogênea. Uma vantagem dessa variação de luz simulada é a possibilidade de variar desde uma imagem mais escura (preta) até uma imagem com luz saturada (branca) e verificar o desempenho do software nesses casos extremos.

Para avaliar de forma quantitativa o nível de iluminação da imagem, foi aplicado o método `cvCalcHist()` da biblioteca OpenCV. Esse método calcula o histograma da imagem de entrada e nos indica em uma escala de 0 a 255 (imagem em escala de cinza de oito bits) qual a frequência em que cada uma das intensidades de luz (0-255) aparecem na imagem. A Figura 4.6 mostra o histograma plotado com a indicação do percentual do valor máximo (255) do pixel de maior intensidade na imagem. Ou seja, no caso da Figura 4.6 o valor do pixel mais claro da imagem é 52% de 255, ou aproximadamente 132.

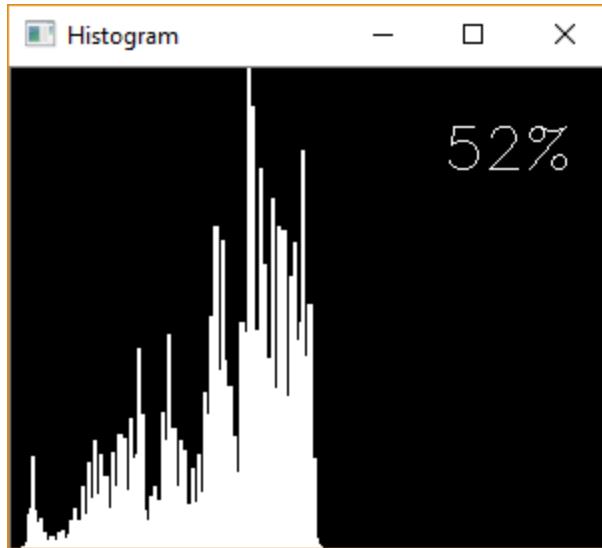


FIGURA 4.6 – Histograma do canal verde no *frame* capturado da câmera XIMEA com indicação do percentual de intensidade do pixel de maior valor na imagem.

Com os indicadores de taxa de acerto e de luminosidade em mãos, foi possível avaliar quantitativamente qual o comportamento do software de visão para variações na luminosidade da imagem. O primeiro teste foi feito variando a luminosidade linearmente, desde um *frame* preto (0%) até um *frame* totalmente branco (100%). Todos os parâmetros foram mantidos iguais aos do teste inicialmente realizado na seção 4.1.2. A Figura 4.7 mostra um gráfico com os dados obtidos da variação do percentual de luminosidade pela taxa de acerto para cada *frame* do vídeo testado. Observa-se que o sistema de visão, com as configurações do método de detecção de borda e de círculos, é mais confiável na faixa de luminosidade entre 50% e 75%.

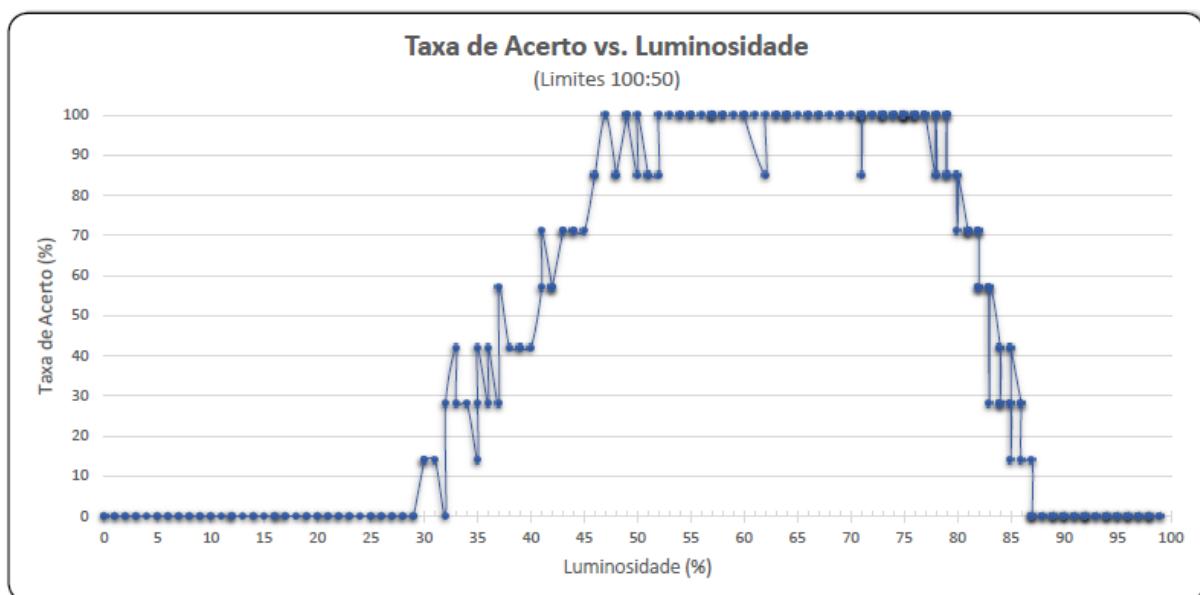


FIGURA 4.7 – Gráfico da taxa de acerto pelo percentual de luminosidade no *frame* para uma configuração de limite inferior de 50 e superior de 100 no algoritmo de Canny.

Decidiu-se então variar os parâmetros de limite superior e inferior do método Canny de detecção de bordas, para tentar aumentar a faixa de confiabilidade do sistema. Os testes foram feitos para limites inferiores de: 5, 15, 20, 25, 50 e 75. Com limites superiores dados pelo dobro de cada um, para manter a relação entre os limites em 2:1.

Verificou-se que a taxa de acerto para os limites 10:5 é melhor nos casos de luz extrema ou falta de luminosidade, isto é, nas bordas do gráfico, e conforme os limites aumentam até 50:25 a taxa de acerto piora nas extremidades e melhora para os valores centrais do gráfico. Porém, quando se aumenta ainda mais os limites para 100:50 e 150:75 a faixa de confiabilidade do sistema de visão se estreita. A maior faixa observada foi entre 24% e 87% de luminosidade, com limites 50:25, como mostra o gráfico da Figura 4.8. O restante dos dados de teste podem ser encontrados no Apêndice A.2.

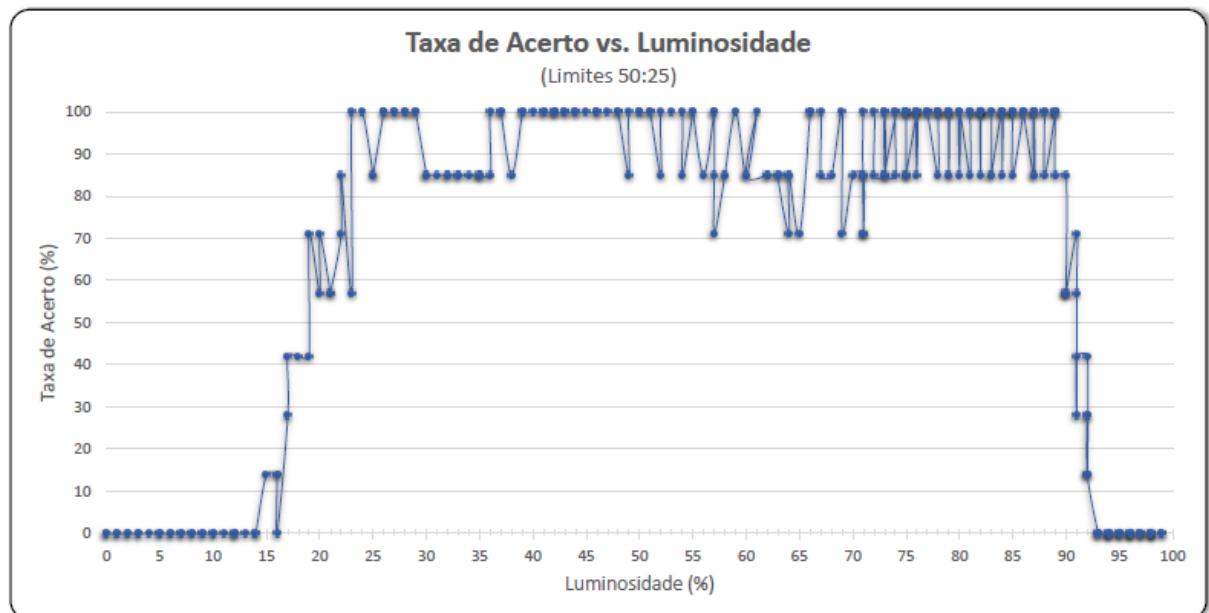


FIGURA 4.8 – Gráfico da taxa de acerto pelo percentual de luminosidade no *frame* para uma configuração de limite inferior de 25 e superior de 50 no algoritmo de Canny.

A Figura 4.9 mostra à esquerda a imagem de um *frame* com luminosidade de 76%. No canto inferior esquerdo é indicada a luminosidade de forma gráfica. Todos os prendedores são encontrados, com uma taxa de acerto de 100%. À direita, está a imagem resultante da aplicação do método de Canny para detecção de bordas, no mesmo *frame*, ajustado com limites superior em 50 e inferior em 25.

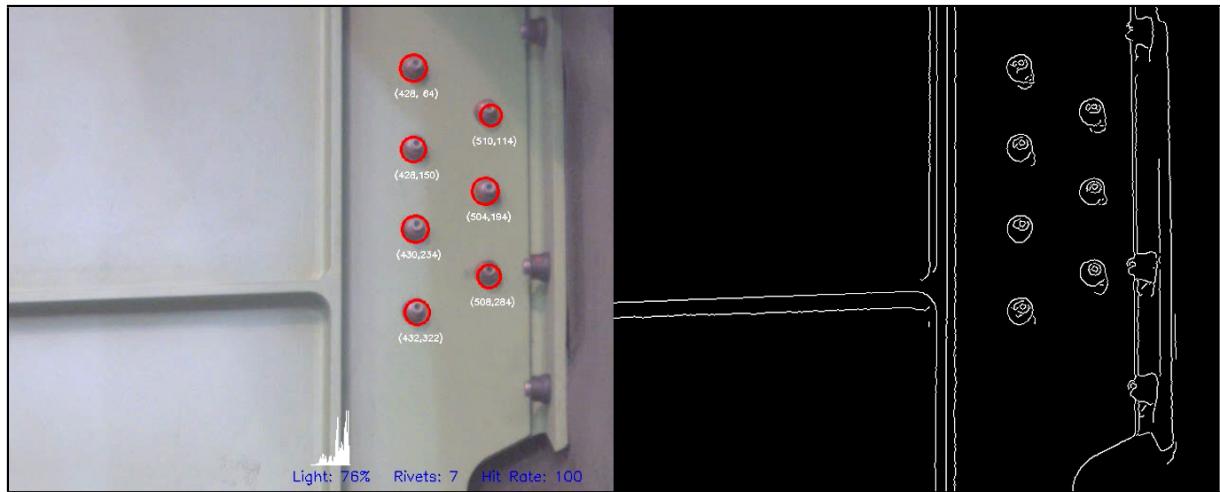


FIGURA 4.9 – *Frame* com 76% de luminosidade e taxa de acerto de 100%. Algoritmo de Canny configurado com limite inferior em 25 e superior em 50.

Para aumentar ainda mais a faixa de operação do sistema, ou seja, a robustez à variação de luminosidade, um teste variando os parâmetros do método de detecção de bordas foi realizado. Associar a boa faixa de operação da configuração com limites em 50:25, com a possibilidade de identificar os prendedores em casos extremos, era o objetivo dessa implementação.

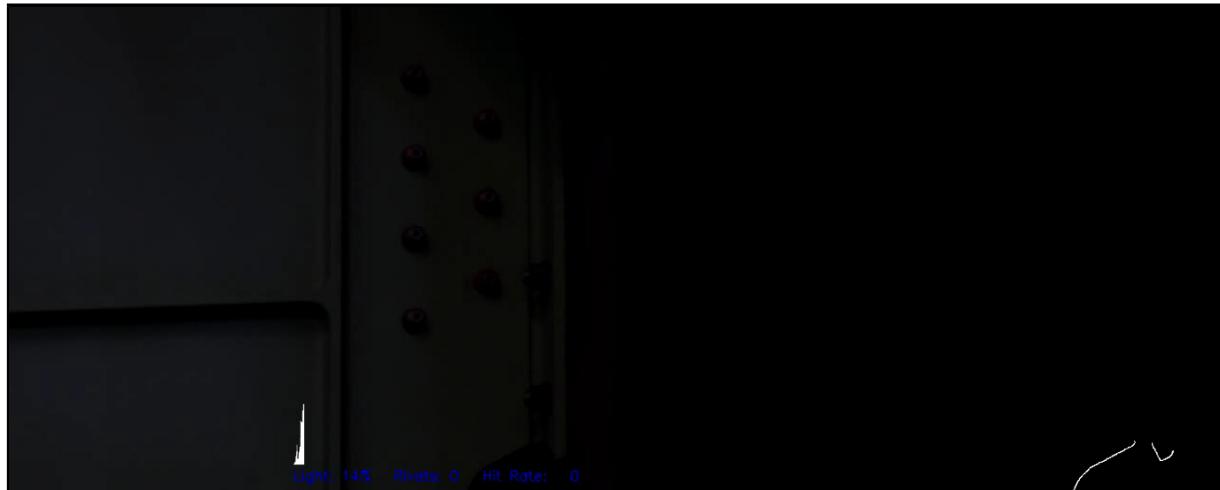


FIGURA 4.10 – *Frame* com 14% de luminosidade e taxa de acerto de 0%. Algoritmo de Canny configurado com limite inferior em 25 e superior em 50.

Observe nas Figuras 4.10 e 4.11 que em casos extremos de luminosidade (14% e 93%, respectivamente), o programa não identifica nenhum prendedor, mesmo na configuração de 50:25 que possui uma grande faixa de operação.

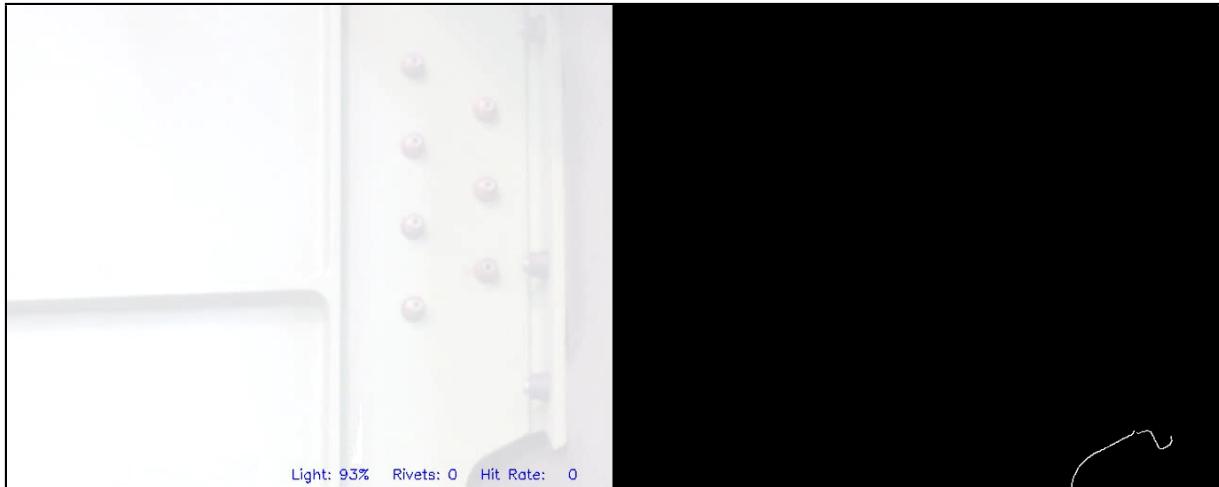


FIGURA 4.11 – *Frame* com 93% de luminosidade e taxa de acerto de 0%. Algoritmo de Canny configurado com limite inferior em 25 e superior em 50.

Uma variação descontínua dos parâmetros de limite de Canny foi implementada. Isto é, para percentuais de luminosidade entre 20% e 90%, os limites teriam valores 50:25 e para o resto da faixa 10:5. Porém, não se obteve o resultado esperado, que era uma ampliação da faixa de luminosidade com alta taxa de acerto. Assim, implementou-se uma variação linear do limite inferior de 5 a 25, entre 0% e 20% de luminosidade, e de 25 a 5, entre 90% e 100%, como mostra o gráfico da Figura 4.12. Mantendo o limite superior sempre o dobro do inferior.

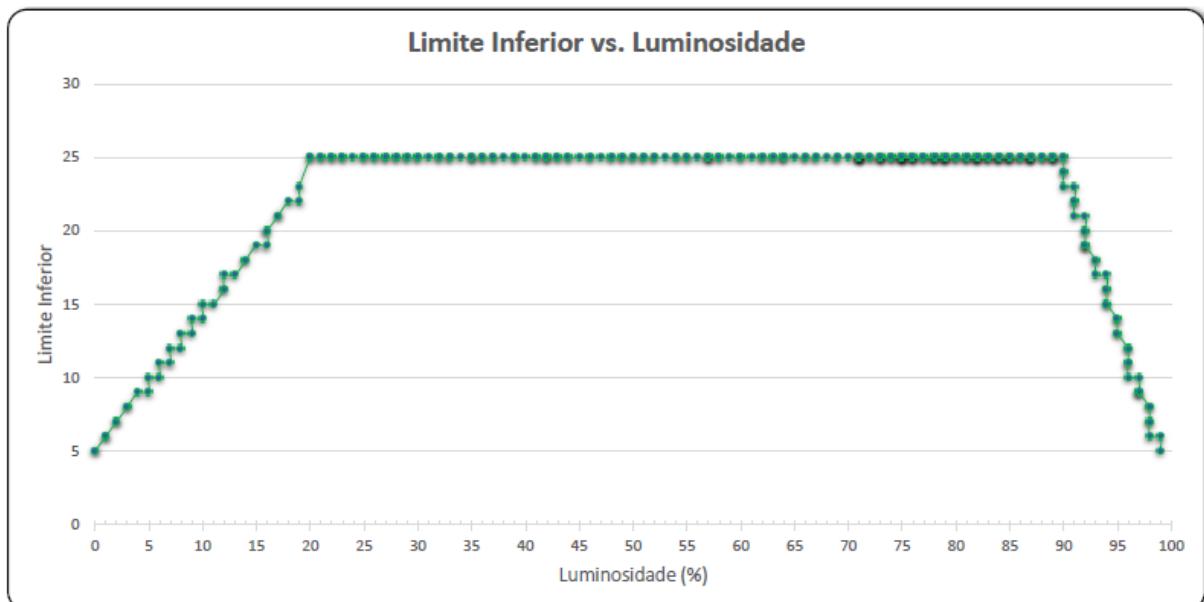


FIGURA 4.12 – Variação do limite inferior do algoritmo de Canny.

O resultado obtido pode ser visto no gráfico da Figura 4.13. Observa-se um leve aumento da faixa de operação do sistema de identificação, porém, a faixa confiável continua entre 24 e 87%.

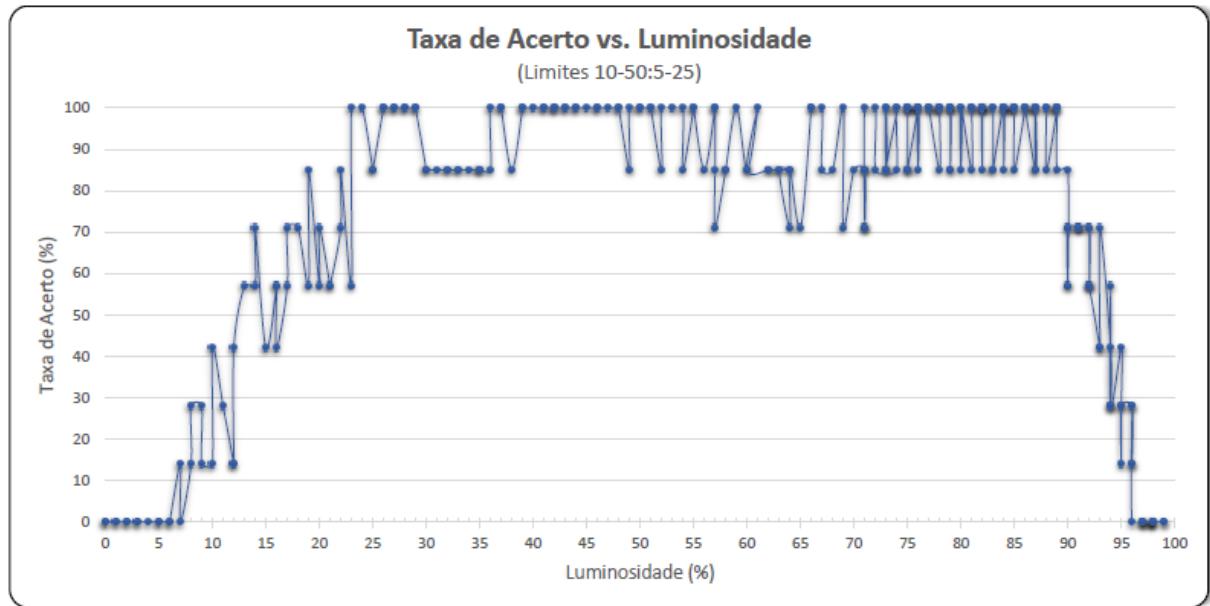


FIGURA 4.13 – Gráfico da taxa de acerto pelo percentual de luminosidade no *frame* para uma configuração de limite inferior do algoritmo de Canny variando conforme a Figura 4.12 e um relação 2:1 entre limites superior:inferior.

A melhoria da faixa de identificação pode ser observada nas Figuras 4.14 e 4.15, que mostram os mesmos *frames* das Figuras 4.10 e 4.11, porém com a configuração dos limites de Canny variando conforme mostrado na Figura 4.12. As taxas de acerto vão de 0% para 71% com essa modificação.

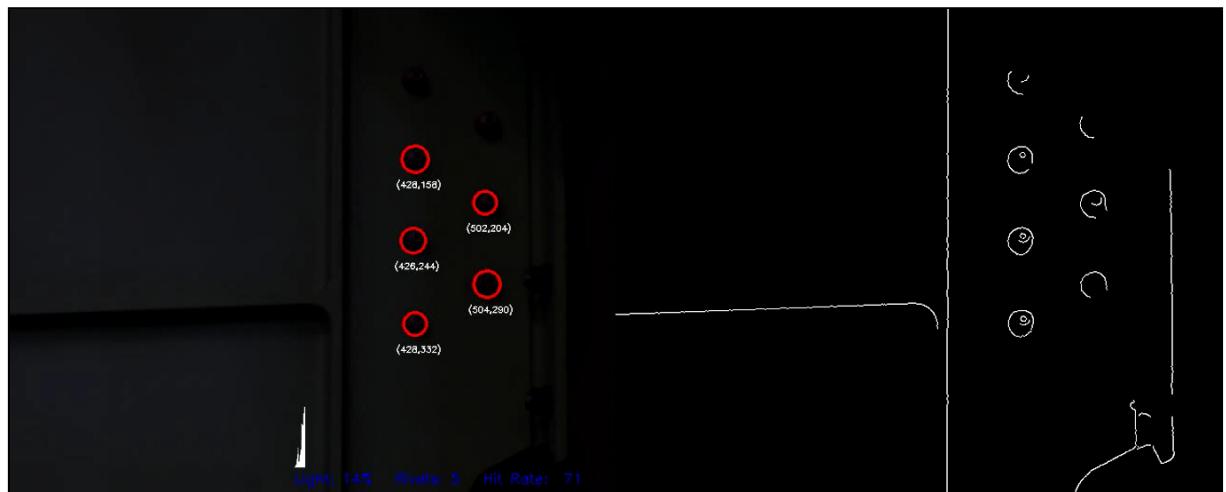


FIGURA 4.14 – *Frame* com 14% de luminosidade e taxa de acerto de 71%. Algoritmo de Canny configurado com limite inferior variando conforme a Figura 4.12 e um relação 2:1 entre limites superior:inferior.



FIGURA 4.15 – *Frame* com 93% de luminosidade e taxa de acerto de 71%. Algoritmo de Canny configurado com limite inferior variando conforme a Figura 4.12 e um relação 2:1 entre limites superior:inferior

4.1.4 Envio de Informações via Serial

Para o teste de envio das coordenadas dos centros dos prendedores via serial, os dispositivos foram configurados conforme mostra o esquemático da Figura 4.16. As portas COM X e COM Y, ilustram portas de comunicação serial, que são criadas de acordo com requisitos específicos de cada computador, onde X e Y podem ser número inteiros positivos.

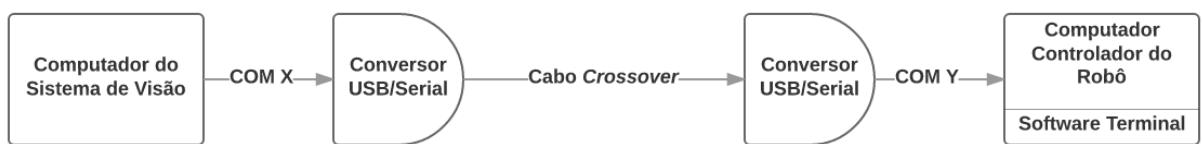


FIGURA 4.16 – Esquemático do sistema de comunicação via serial entre os computadores de visão e de controle do robô.

As coordenadas dos centros dos prendedores identificados são enviadas via serial a cada novo *frame* capturado pelo programa (veja a Figura 4.17). É importante ressaltar que as dimensões dos prendedores não são passadas, pois o objetivo é que o software de visão indique para o programa de controle do robô apenas as posições dos centros dos prendedores, que possuem diâmetro padrão previamente conhecido pelo software. O envio do diâmetro dos prendedores identificados pode ser implementado de forma simples, caso seja necessário. Essa informação pode auxiliar na detecção de prendedores com tamanhos fora das especificações de projeto.

```

Opening serial port...OK
Sending bytes...9 bytes written
Closing serial port...OK
Opening serial port...OK
Sending bytes...9 bytes written
Closing serial port...OK
Opening serial port...OK
Sending bytes...9 bytes written
Closing serial port...OK
Opening serial port...OK
Sending bytes...9 bytes written
Closing serial port...OK
Opening serial port...OK
Sending bytes...1 bytes written
Closing serial port...OK
Opening serial port...OK
Sending bytes...9 bytes written
Closing serial port...OK
Opening serial port...OK
Sending bytes...9 bytes written
Closing serial port...OK
Opening serial port...OK
Sending bytes...9 bytes written
Closing serial port...OK
Opening serial port...OK

```

FIGURA 4.17 – Terminal do software de identificação indicando a abertura da porta serial, o envio dos *bytes* contendo as coordenadas dos prendedores e o fechamento da porta.

A Figura 4.18 mostra a tela do software terminal Tera Term Web 3.1, que pode ser obtido gratuitamente através do link: <https://www.ayera.com/teraterm/>. Esse terminal é utilizado para ler de uma porta serial as informações enviadas pelo programa de identificação (veja a Figura 4.17), executado em outro computador. Ou seja, cada extremidade do cabo de comunicação se refere à uma porta serial (COM), em um lado o software de visão envia as coordenadas dos pontos processadas em um computador, e do outro o programa terminal lê essas informações através de outra máquina. O terminal nesse caso serve para ilustrar os dados sendo recebidos por outro computador, ele faz o papel do computador controlador do robô que recebe as informações das coordenadas.

FIGURA 4.18 – Terminal recebendo as coordenadas dos prendedores em tempo real.

Como citado na seção anterior, a identificação dos pontos de início e fim dos filetes deve

ser melhor desenvolvida, com uma abordagem um pouco mais dependente da preparação da peça ou de localizações predefinidas. Por esse motivo, o software não está configurado para o envio dessas informações via porta serial. Porém, a adição dessas informações para envio no futuro é simples.

Vale ressaltar que as coordenadas enviadas pelo software de visão fazem parte do plano da imagem e são dadas em pixels, ou seja, a coordenada (430,70) por exemplo (primeiro prendedor identificado na Figura 4.4), significa que o ponto central do prendedor na imagem atual está localizado a 430 pixels à direita e 70 pixels para baixo, dado que a origem (0,0) no plano da imagem é o canto superior esquerdo (veja a Figura 2.3). Assim, uma transformação do sistema de coordenadas ainda deve ser feita pelo software que controla o robô para que os dados recebidos sejam devidamente ajustados ao plano real. Para isso, a adição de sensores de distância se faz necessária, com o objetivo de medir a distância em Z da câmera até os prendedores e garantir o paralelismo entre os sistemas de coordenadas.

5 Conclusão e Sugestões para Trabalhos Futuros

Neste capítulo são encontradas duas seções. A primeira é a 5.1, onde são apresentadas as conclusões do trabalho, considerando os objetivos do desenvolvimento e os resultados obtidos. Por último, as sugestões para trabalhos futuros são mostradas na seção 5.2.

5.1 Conclusão

Realizou-se nesse trabalho o projeto de um sistema de visão computacional capaz de identificar prendedores em um painel de asa e enviar as coordenadas que caracterizam esse objeto via comunicação serial em tempo real com o controlador do robô que fará a selagem desses prendedores. Utilizou-se câmera, lente e filtros em imagens para a detecção de objetos de geometrias simples e conhecidas. Foram feitos estudos detalhados sobre o funcionamento dos sensores e lentes em câmeras digitais, representação de imagens, métodos de correção para distorções advindas de lentes, comunicação em tempo real entre dispositivos e por fim, algoritmos e métodos computacionais capazes de filtrar, identificar e transmitir a localização de objetos de interesse em uma imagem.

O uso da linguagem C juntamente com a biblioteca OpenCV se deu pelo prévio conhecimento da biblioteca e pela possibilidade de incorporação dentro de outros ambientes de programação. Outra opção era o uso da linguagem LabVIEW, desenvolvida pela National Instruments, que é a plataforma padrão do projeto AME ASA, porém a biblioteca OpenCV foi escolhida pela sua grande flexibilidade quando se trata de processamento de imagens e sua vasta comunidade de pesquisa na internet. É importante destacar que o software de visão pode ser adicionado como uma função dentro do ambiente LabVIEW, rodar na mesma máquina e se comunicar com o software em LabVIEW, ou ainda utilizar a comunicação serial desenvolvida para enviar os dados capturados pelo sistema de visão a partir de outra máquina que estiver sendo executado.

Funções dedicadas da biblioteca OpenCV para suavização em imagens, detecção de bordas e identificação de círculos e linhas foram estudadas e implementadas. Os métodos

de processamento de imagens para destacar as características dos objetos, como a função `cvSmooth()` e `cvCanny()` se mostraram bem eficientes. Em relação aos algoritmos de identificação das formas, a função `cvHoughCircles()` cumpriu bem o seu objetivo. Já a função `cvHoughLines2()` não conseguiu por si só identificar claramente os filetes no painel, confundindo as arestas da estrutura com filetes.

O desenvolvimento da função de comunicação dentro do programa permite que após reconhecer os objetos, suas coordenadas sejam enviadas via serial para outro computador, o que garante o paralelismo entre o processamento de imagens e a execução da tarefa de selagem. O envio das coordenadas com o cabo de comunicação serial teve bons resultados, mostrando a viabilidade do seu uso.

Os testes de robustez à variação da iluminação no ambiente tiveram ótimos resultados, indicando uma ampla faixa de operação confiável do sistema de visão. Mesmo nos *frames* em que a taxa de acerto não foi de 100%, dentro da faixa confiável a posição do prendedor não encontrado pode ser inferida pelas identificações nos *frames* vizinhos, dado que em um curto intervalo de tempo as posições dos prendedores não variam.

As dificuldades encontradas ao longo do projeto, forçaram algumas tomadas de decisões no decorrer do seu desenvolvimento. A teoria aprendida durante as fases iniciais serviu como base para essas escolhas, que ao final se mostraram coerentes com os resultados obtidos. Uma das primeiras decisões foi sobre o modelo da câmera, a XIMEA foi escolhida em função do seu tamanho, qualidade, robustez e disponibilidade no laboratório. A lente acoplada à ela já estava adequada à aplicação, enquadrando boa parte do painel a uma distância de aproximadamente um metro e meio. O resultado da calibração realizada foi satisfatório e mostrou a qualidade da câmera escolhida no início do desenvolvimento. Em trabalhos futuros seria possível operar sem esse processo, dependendo do grau de incerteza permitido pela aplicação. Vale ressaltar que os estudos aqui desenvolvidos se adaptam facilmente a qualquer câmera que for adotada posteriormente pelo sistema SELA.

Por fim, a identificação dos prendedores na imagem pelo sistema foi satisfatória e robusta às variações de luminosidade. Todos os testes realizados se mostraram aplicáveis à solução. Já a identificação de filetes mostrou-se imatura a priori, soluções futuras com uma melhor caracterização dos elementos lineares que serão selados devem aprimorar a aplicação para uso na selagem de painéis de asas.

5.2 Sugestões para Trabalhos Futuros

O reconhecimento dos filetes por sistema de visão mostrou-se uma tarefa complexa, pela dificuldade em diferenciar filetes de estruturas retas na asa. Aprimorar o sistema de identificação de filetes desenvolvido nesta pesquisa, considerando também formas mais

complexas, além de linhas retas, é uma sugestão de trabalho.

Relacionar o sistema de coordenadas da câmera ao sistema real do robô LBR iiwa, utilizando sensores infra-vermelho para medir a distância e manter a superfície de selagem paralela ao plano da imagem. Integrar os sistemas de visão, controle do robô e controle do processo de selagem. E por fim, realizar testes de aproximação e aplicação do selante no produto.

O sistema SELA prevê a inspeção de qualidade do selante aplicado usando um projetor laser 2D e a câmera. O desenvolvimento da função de inspeção no sistema de visão talvez seja a melhor alternativa de implementação.

Referências

- BEVILAQUA, D. **Desenvolvimento de Sistema de Visão para Movimentação de Robô Móvel Pioneer P3-AT**. Monografia (Graduação) — Universidade de Brasília, Brasília, Julho 2011.
- BRADSKI, G.; KAEHLER, A. **Learning OpenCV**. [S.l.]: OReilly Media, Inc., 2008.
- CANNY, J. A computational approach to edge detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence** 8, p. 679–714, 1986.
- DEVLIEG, R.; FEIKERT, E. One-up assembly with robots. -, 2008.
- DOCS.OPENCV. **Camera calibration With OpenCV**. [S.l.], 2011–2014. Disponível em: <http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration-camera_calibration.html>.
- DUDA, R. O.; HART, P. E. Use of the hough transformation to detect lines and curves in pictures. **Communications of the Association for Computing Machinery** 15, p. 11–15, 1972.
- FUJIFILM CORPORATION. **FUJINON CCTV Lens - For FA/Machine Vision Brochure**. [S.l.], 2009. Disponível em: <www.fujifilm.com>.
- GAMAL, A. E.; ELTOUKHY, H. **Cmos Image Sensors**. [S.l.]: IEEE Circuits & Devices Magazine, 2005.
- GONZALES. **Processamento de Imagens Digitais**. [S.l.]: Prentice Hall, 2002.
- HOUGH, P. V. C. Machine analysis of bubble chamber pictures. **Proceedings of the International Conference on High Energy Accelerators and Instrumentation**, p. 554–556, 1959.
- KUKA. **KUKA LBR iiwa Brochure**. [S.l.], 2014. Disponível em: <lbr-iiwa.com>.
- LIMA, F. **Implementação de Controle de rolagem e Arfagem com Realimentação Visual Aplicado a um Quadrirrotor Comercial**. Monografia (Graduação) — Universidade de Brasília - UnB, 2013.
- LITWILLER, D. **CMOS vs. CCD: Maturing Technologies, Maturing Markets**. [S.l.], 2005.

MUSSABAYEV, R. R. Colour-based object detection, inverse kinematics algorithms and pinhole camera model for controlling robotic arm movement system. **Twelve International Conference on Electronics Computer and Computation (ICECCO)**, 2015.

ROSSI, D. L. **Projeto de um controlador PID para controle de ganho de uma câmera com sensor CMOS utilizando computação reconfigurável**. Dissertação (Mestrado) — USP - São Carlos, 2012.

SCAFE, A. P. Developing robotic sealing processes in aerospace manufacturing. **SAE International**, 2012.

SCHELEYER, G. A new colour image segmentation. **6th International Conference on Computers Communications and Control (ICCCC)**, 2016.

WIKIPEDIA. Hsl and hsv. **Wikipedia**, 2017. Disponível em:
<https://en.wikipedia.org/wiki/HSL_and_HSV>.

XIMEA. **Brochure - USB3 Vision camera series**. [S.l.], November 2014. Disponível em: <www.ximea.com>.

Apêndice A - Informações Adicionais

A.1 Calibração

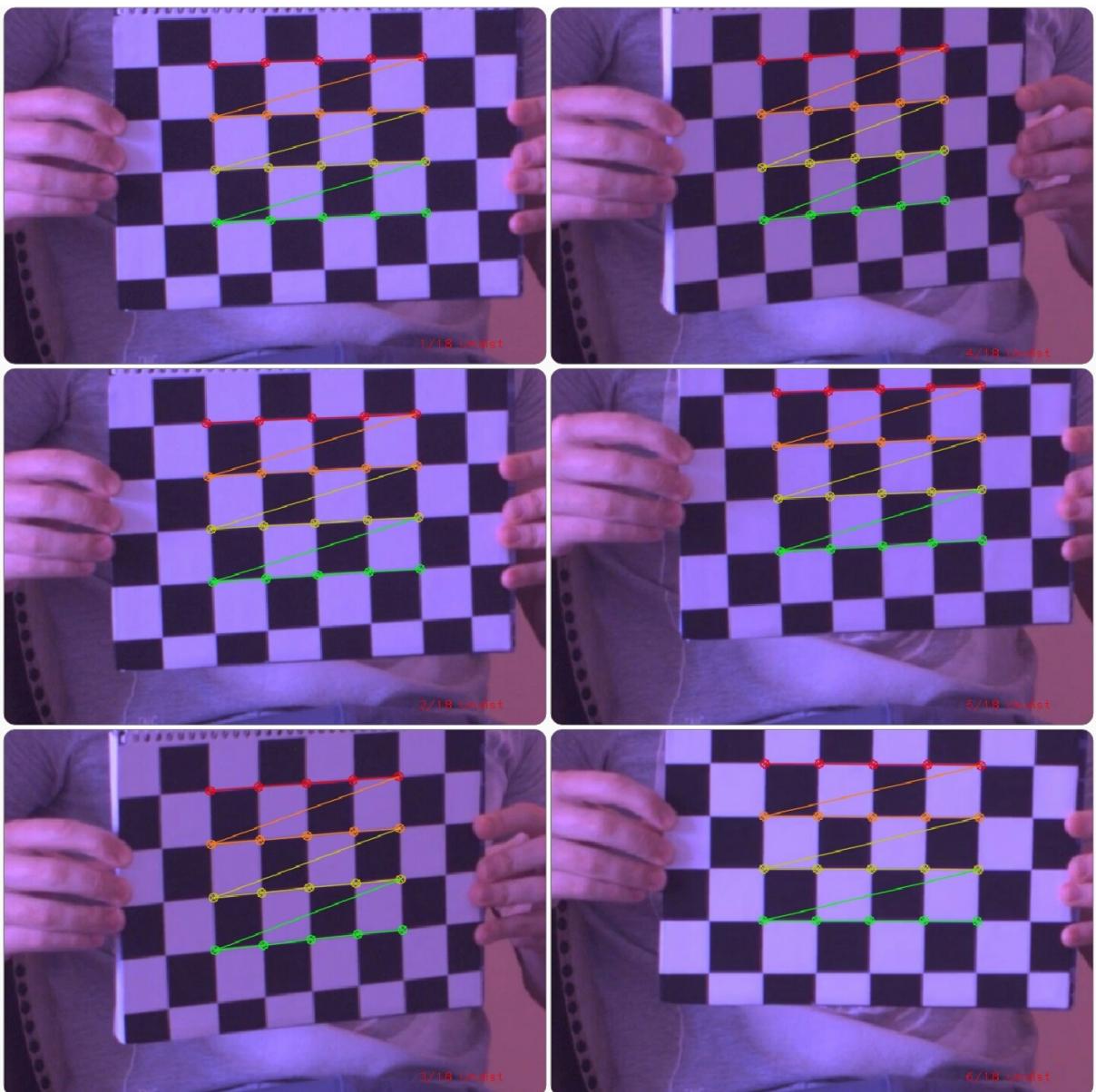


FIGURA A.1 – Imagens de 1 a 6 usadas para calibração, com identificação das arestas.

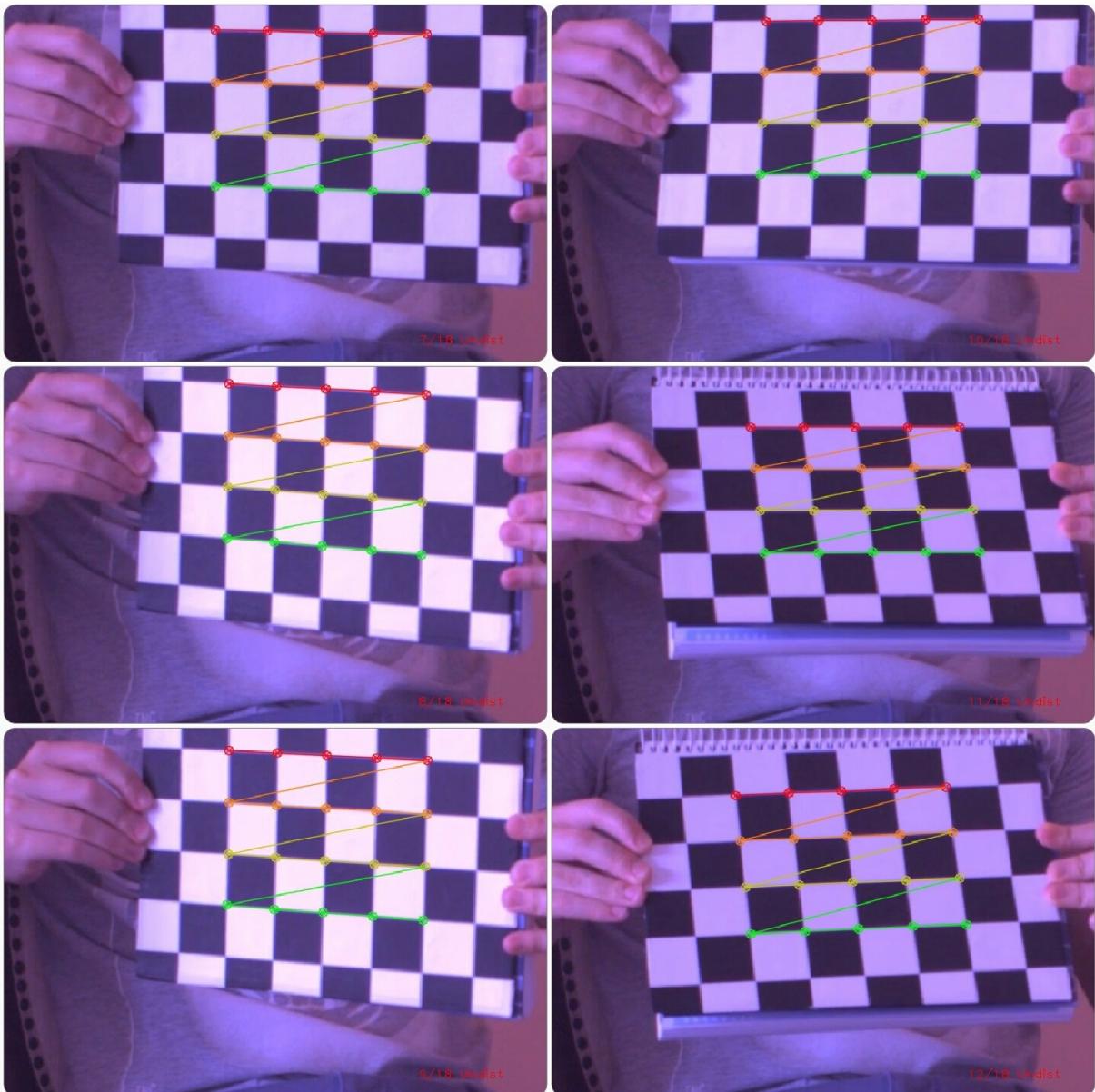


FIGURA A.2 – Imagens de 7 a 12 usadas para calibração, com identificação das arestas.

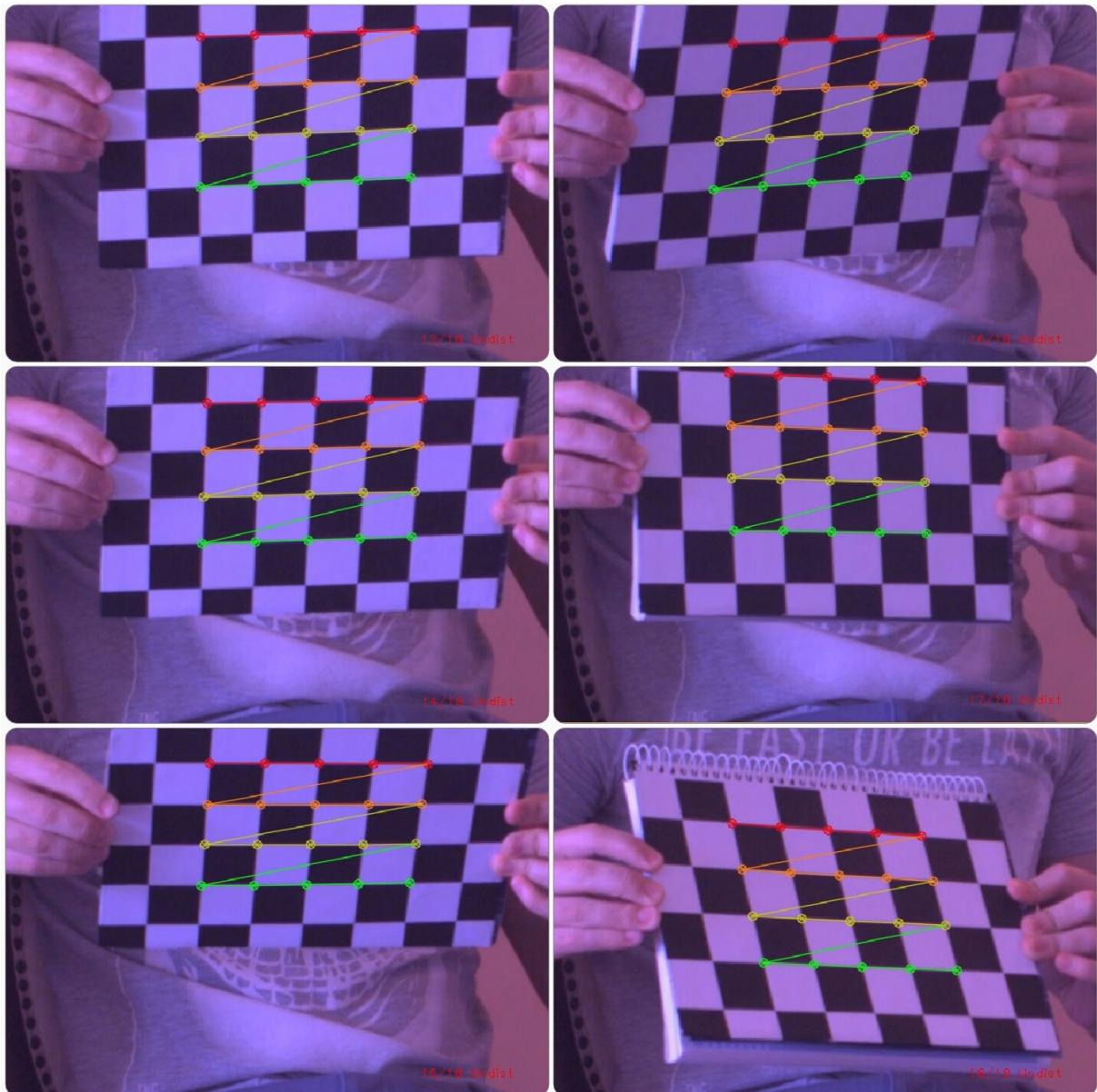


FIGURA A.3 – Imagens de 13 a 18 usadas para calibração, com identificação das arestas.

```

<?xml version="1.0"?>
<opencv_storage>
<calibration_Time>01/24/17 20:24:09</calibration_Time>
<nrOfFrames>18</nrOfFrames>
<image_Width>640</image_Width>
<image_Height>512</image_Height>
<board_Width>5</board_Width>
<board_Height>4</board_Height>
<square_Size>40.</square_Size>
<FixAspectRatio>1.</FixAspectRatio>
<!-- flags: +fix_aspectRatio +fix_principal_point +zero_tangent_dist -->
<flagValue>14</flagValue>
<Camera_Matrix type_id="opencv-matrix">
<rows>3</rows>
<cols>3</cols>
<dt>d</dt>
<data>
    3.1051754555335883e+003 0. 3.195000000000000e+002 0.
    3.1051754555335883e+003 2.555000000000000e+002 0. 0. 1.</data></Camera_Matrix>
<Distortion_Coefficients type_id="opencv-matrix">
<rows>5</rows>
<cols>1</cols>
<dt>d</dt>
<data>
    -6.4462027868245353e-001 1.4889709299256413e+001 0. 0.
    1.2222013452930904e+002</data></Distortion_Coefficients>
<Avg_Reprojection_Error>3.6130583304891806e-001</Avg_Reprojection_Error>

```

FIGURA A.4 – Dados de saída do programa de calibração.

A.2 Testes com Variação de Iluminação

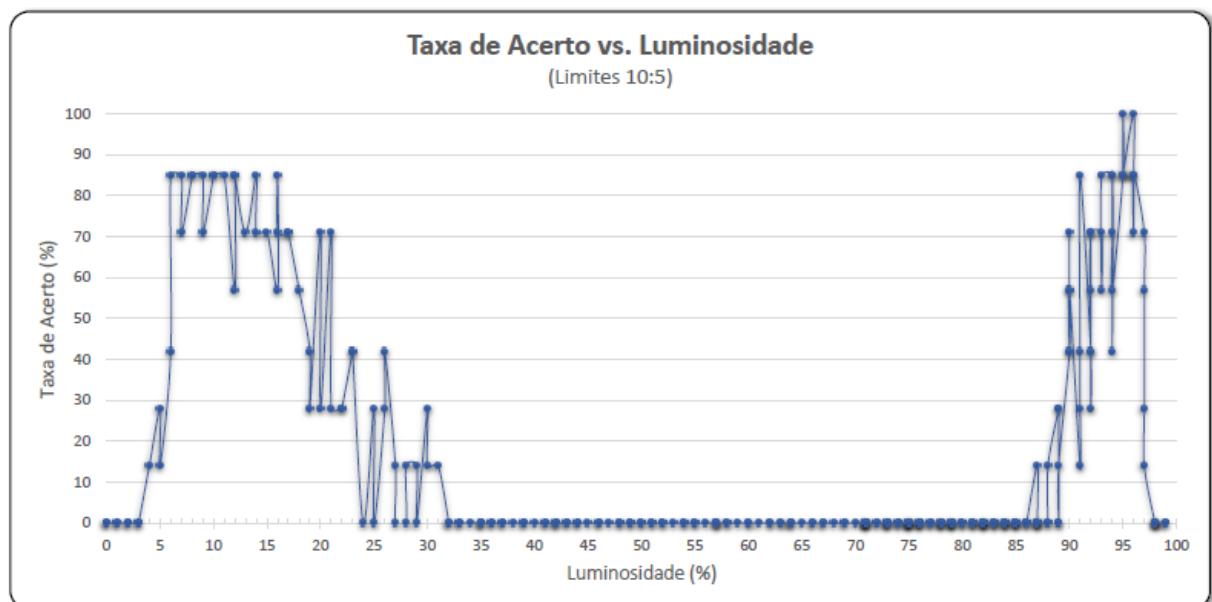


FIGURA A.5 – Gráfico da taxa de acerto pelo percentual de luminosidade no *frame* para uma configuração de limite inferior de 5 e superior de 10 no algoritmo de Canny.

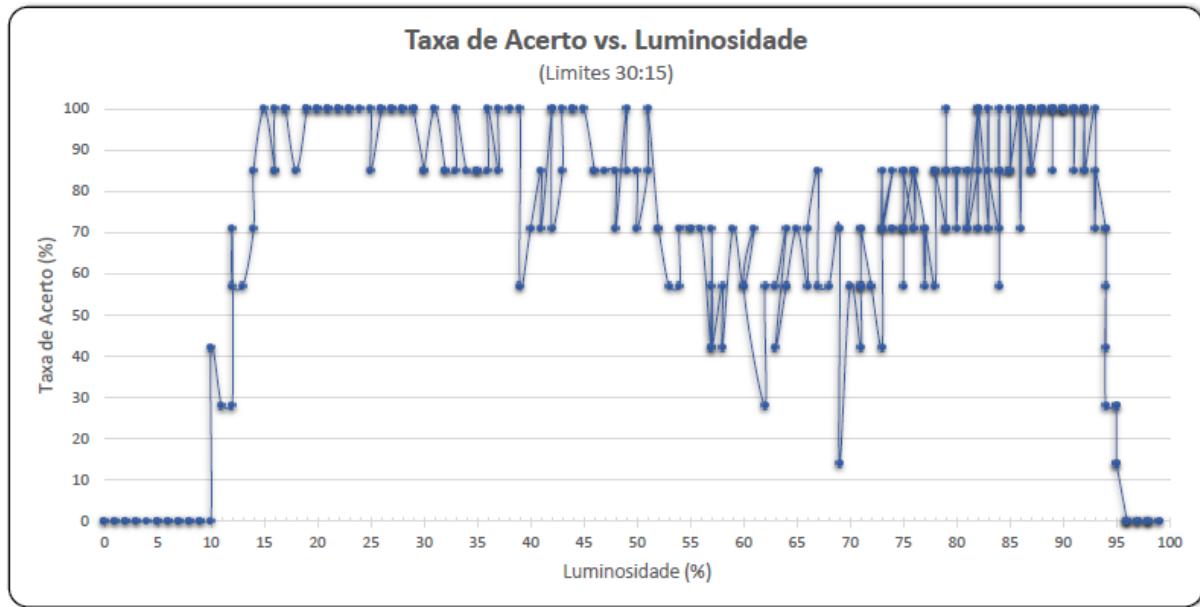


FIGURA A.6 – Gráfico da taxa de acerto pelo percentual de luminosidade no *frame* para uma configuração de limite inferior de 15 e superior de 30 no algoritmo de Canny.

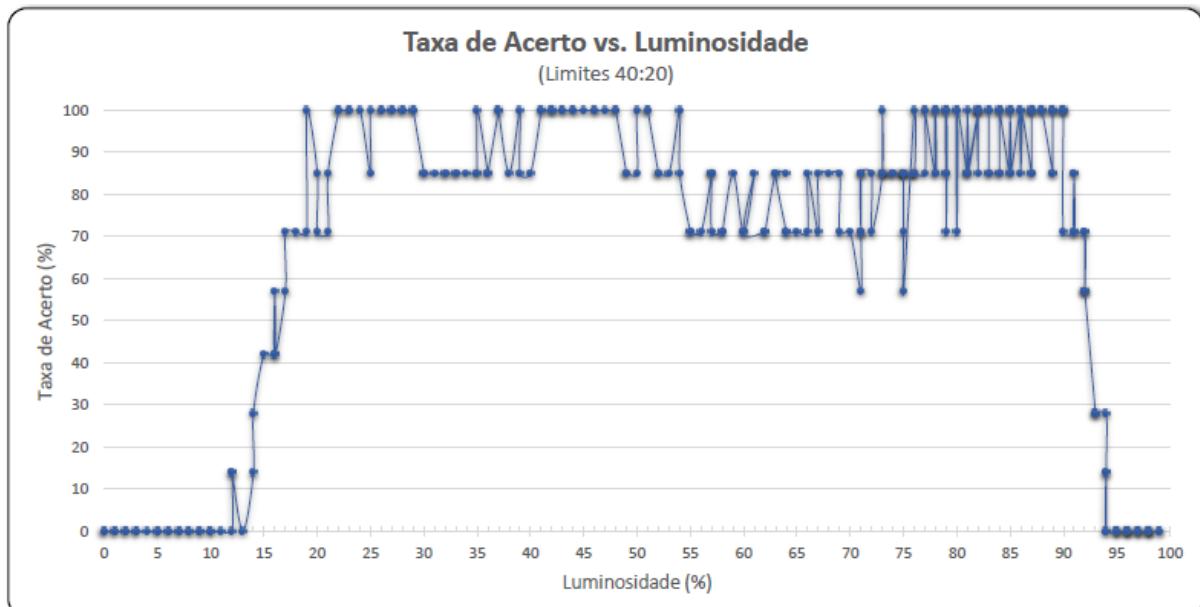


FIGURA A.7 – Gráfico da taxa de acerto pelo percentual de luminosidade no *frame* para uma configuração de limite inferior de 20 e superior de 40 no algoritmo de Canny.

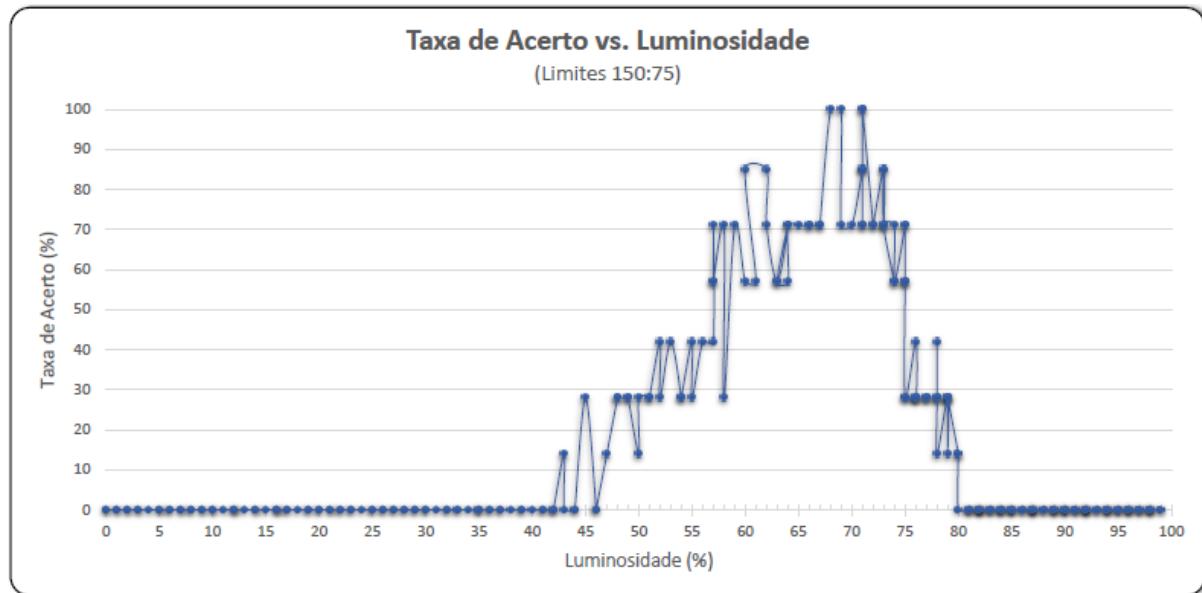


FIGURA A.8 – Gráfico da taxa de acerto pelo percentual de luminosidade no *frame* para uma configuração de limite inferior de 75 e superior de 150 no algoritmo de Canny.

A.3 Robô SELA

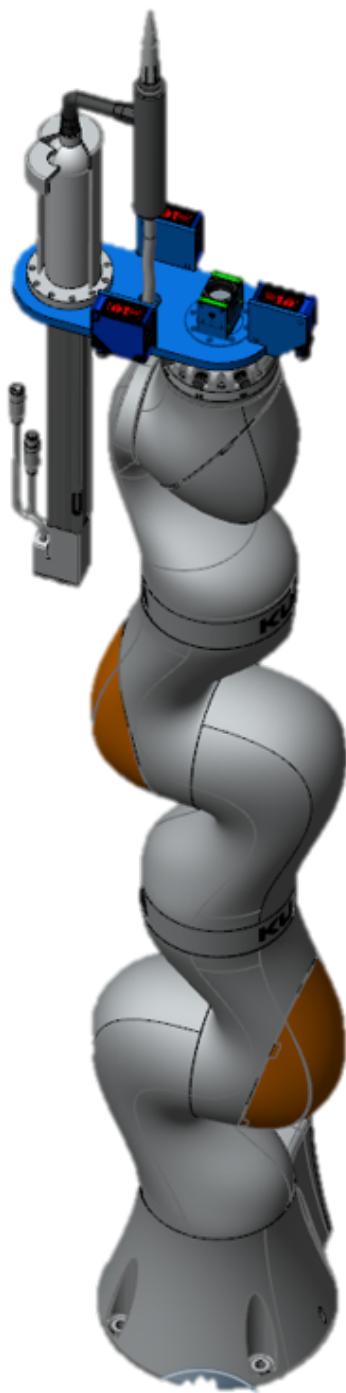


FIGURA A.9 – Desenho 3D do robô KUKA LBR iiwa 14 R820 com o conjunto atuador responsável pela selagem.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO DP	2. DATA 21 de março de 2017	3. DOCUMENTO Nº DCTA/ITA/DP-031/2017	4. Nº DE PÁGINAS 79
5. TÍTULO E SUBTÍTULO: Sistema de Visão Computacional Aplicado ao Processo de Selagem de Estruturas e Componentes Aeronáuticos			
6. AUTOR(ES): Felipe de Paula Lima			
7. INSTITUIÇÃO(ÓES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÓES): Instituto Tecnológico de Aeronáutica – ITA			
8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR: Asa; Automação da manufatura; Filete; OpenCV; Prendedor; Selagem; Visão computacional.			
9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: Processos de manufatura; Automação; Visão por computadores; Asas; Indústria aeronáutica; Engenharia mecânica.			
10. APRESENTAÇÃO:		<input checked="" type="checkbox"/> Nacional <input type="checkbox"/> Internacional	
ITA, São José dos Campos. Curso de Mestrado Profissional em Engenharia Aeronáutica. Programa de Pós-Graduação em Engenharia Aeronáutica e Mecânica. Orientador: Prof. Dr. Luís Gonzaga Trabasso. Coorientador: Eng. Msc. Elton Candia Cordeiro. Defesa em 09/03/2017. Publicada em 2017.			
11. RESUMO: <p>Com a crescente busca pela automação dos processos de manufatura, procurando sempre o aumento de eficiência, com soluções de baixo impacto ambiental, preservação do bem-estar dos seres humanos envolvidos na tarefa e mínimas mudanças nas instalações físicas já instauradas, está sendo executado pelo ITA em parceria com a Embraer, o projeto Automação da Montagem Estrutural de Asas (AME ASA). Dentro dele está inserido o sistema de selagem automática de asas, denominado sistema SELA, responsável pela automação do processo de selagem de prendedores e filetes em asas. Uma das funções necessárias para a consecução da selagem é a identificação de prendedores e filetes. É este o objetivo deste trabalho. Foi desenvolvido um sistema de visão computacional que identifica o tipo de componente aeronáutico (prendedor ou filete), calcula a posição de seu centro de área, para o caso dos prendedores, ou a posição de suas extremidades, para o caso dos filetes, e envia esta informação para o controlador do robô que, por sua vez, executa as tarefas programadas de selagem, com baixa intervenção do operador. Os equipamentos, métodos e software utilizados no sistema computacional estão detalhados no desenvolvimento dessa dissertação. Um método indicador de desempenho do sistema de visão, capaz de avaliar a taxa de acerto dada uma referência conhecida, foi criado. Ou seja, sabendo-se a quantidade de prendedores no painel enquadrados pela câmera, o indicador retorna a taxa de acerto, considerando prendedores não identificados e falsas identificações. Testes com simulação da variação de luminosidade e seu efeito na taxa de acerto tiveram resultados satisfatórios, mostrando robustez adequada a possíveis variações na iluminação. O sistema SELA prevê um sistema dedicado de luz artificial para aumentar ainda mais a robustez na identificação dos componentes e auxiliar a inspeção de qualidade do selante.</p>			
12. GRAU DE SIGILO: (X) OSTENSIVO <input type="checkbox"/> RESERVADO <input type="checkbox"/> CONFIDENCIAL <input type="checkbox"/> SECRETO			