UNIVERSIDADE DE TAUBATÉ
Rodrigo Barbosa Tudeschini

SISTEMA DE INSPEÇÃO AUTOMÁTICA DE ADESIVO EM PARA-BRISAS VEICULAR COM USO DE VISÃO COMPUTACIONAL

Taubaté - SP
2022

Rodrigo Barbosa Tudeschini

SISTEMA DE INSPEÇÃO AUTOMÁTICA DE ADESIVO EM PARA-BRISAS VEICULAR COM USO DE VISÃO COMPUTACIONAL

Monografia apresentada para obtenção do Título de Mestre pelo Curso Mestrado Profissional em Engenharia Mecânica do Departamento de Engenharia Mecânica da Universidade de Taubaté.
Área de Concentração: Automação e Dinâmica dos Sistemas
Orientador: Prof. Dr. Evandro Luís Nohara

Taubaté - SP
2022

Grupo Especial de Tratamento da Informação - GETI
Sistema Integrado de Bibliotecas – SIBi
Universidade de Taubaté - Unitau

T899S Tudeschini, Rodrigo Barbosa
Sistema de inspeção automática de adesivo em para-brisas veicular com uso de visão computacional / Rodrigo Barbosa Tudeschini. -- 2022.
66 f. : il.

Dissertação (mestrado) – Universidade de Taubaté, Pró-reitoria de Pesquisa e Pós-graduação, Taubaté, 2022.
Orientação: Prof. Dr. Evandro Luís Nohara, Departamento de Engenharia Mecânica.

1. Visão computacional. 2. Inspeção automática. 3. Aplicação de Adesivo. I. Universidade de Taubaté. Departamento de Engenharia Mecânica. Mestrado em Engenharia Mecânica. II. Titulo.

CDD – 621.3

Ficha catalográfica elaborada **por Shirlei Righeti – CRB-8/6995**

Rodrigo Barbosa Tudeschini

SISTEMA DE INSPEÇÃO AUTOMÁTICA DE ADESIVO EM PARA-BRISAS VEICULAR COM USO DE VISÃO COMPUTACIONAL

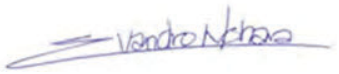
Dissertação apresentada para obtenção do Título de Mestre pelo Curso Mestrado Profissional em Engenharia Mecânica do Departamento de Engenharia Mecânica da Universidade de Taubaté.
Área de Concentração: Automação e Dinâmica dos Sistemas

Data: 29/04/2022

Resultado: Aprovado

BANCA EXAMINADORA

Prof. Dr. Evandro Luís Nohara - Orientador. Universidade de Taubaté

Assinatura: 

Prof. Dr. Luis Filipe de Faria Pereira Wiltgen Barbosa - Membro Interno. Universidade de Taubaté

Assinatura: 

Prof. Dr. Daniel Julien Barros da Silva Sampaio - Membro Externo. Universidade Estadual Paulista

Assinatura: 

Dedico esta pesquisa a todos os professores, que dedicam suas vidas à formação de pessoas e transmissão de conhecimento.

AGRADECIMENTOS

Aos meus pais, que através do exemplo me mostraram a importância e o valor do trabalho, do estudo e do conhecimento.

À Mariana, pelo amor, paciência e parceria ao longo de tantos anos.

À Jaguar Land Rover, em especial ao Sr. João Mattosinho, pelo apoio à pesquisa e incentivo à inovação.

E a todos os que contribuíram de alguma forma para a minha formação, em especial ao meu orientador, pelos ensinamentos transmitidos.

“Los ordenadores son inútiles. Sólo pueden darnos respuestas”

Pablo Picasso

RESUMO

Os adesivos à base de poliuretano são aplicados por sistemas com controle automático, visando garantir a repetibilidade dos processos de aplicação de adesivos nos para-brisas dos veículos produzidos no Brasil. Bolhas de ar no interior do sistema de bombeamento ou entupimentos podem levar a uma interrupção momentânea na aplicação do cordão de adesivo, que nem sempre pode ser detectada pelos sensores do sistema de aplicação, causando um custo alto para a qualidade. Uma falha no cordão de adesivo pode permitir a entrada de água entre o para-brisa e a carroceria do veículo, que só pode ser detectada no teste de estanqueidade, quando o veículo estará completamente montado. E em alguns casos, mesmo o teste de estanqueidade não é capaz de detectar o problema, causando alto custo para reparo em garantia, transtornos aos clientes e prejuízos à empresa. Para detectar uma interrupção no cordão de adesivo logo após sua aplicação no para-brisas, antes que seja montado no veículo, um sistema de inspeção automática baseado em visão computacional é proposto, baseado em hardware de baixo custo, programação em linguagem Python e fazendo uso de bibliotecas de código aberto. Um lote de para-brisas sem defeito foi inspecionado com o uso do sistema de inspeção proposto. Na impossibilidade de obter peças defeituosas para validação, imagens de para-brisas foram modificadas para simular defeitos e as imagens foram avaliadas pelo algoritmo desenvolvido. Ao final, os resultados foram analisados, estabelecendo a eficácia do sistema em 100% de capacidade de detecção de defeitos e 21% de detecções falsas.

Palavras-chave: Visão Computacional, Inspeção Automática, Aplicação de Adesivo.

ABSTRACT

Polyurethane-based adhesives are applied by systems with automatic control, in order to ensure the repeatability of the adhesive application processes on the windshields of vehicles produced in Brazil. Air bubbles inside the pumping system or clogging can lead to a momentary interruption in the application of the adhesive bead, which cannot always be detected by the application system's sensors, causing a high cost for quality. A failure in the adhesive bead could allow water to ingress between the windshield and the vehicle body, which can only be detected in the leak test, when the vehicle is already completely assembled. In some cases, even the leak test is not able to detect the problem, causing high cost due to warranty repairs, inconvenience to customers and damage to the brand. To detect an interruption in the adhesive bead right after its application on the windshield, before it is fitted to the vehicle, an automatic inspection system based on computer vision is proposed, based on low-cost hardware, programming in Python language and making use of open-source libraries. A batch of defect-free windshields was inspected using the proposed inspection system. In the impossibility of obtaining defective parts for validation, windshield images were modified to simulate defects and the images were evaluated by the developed algorithm. In the end, the results were analysed, establishing the system's effectiveness at 100% for defect detection capability and 21% of false detections.

Keywords: Computer Vision, Automatic inspection, Glue application

LISTA DE ILUSTRAÇÕES

Figura 1 – Distribuição das publicações ao longo do tempo 5
Figura 2 – Distribuição das publicações por tipo 5
Figura 3 – Distribuição das publicações por assunto 6
Figura 4 – Forma do cordão recomendada para aplicação de adesivo PU 8
Figura 5 – Aplicação de adesivo em para-brisas automotivo 9
Figura 6 – Componentes do sistema de controle e monitoramento da aplicação de adesivo 10
Figura 7 – Modelo da "camera obscura" 14
Figura 8 – Método de mosaico utilizado na captação independente de distintos comprimentos de onda 15
Figura 9 – Representação em 8 bits de imagem em tons de cinza 16
Figura 10 – Imagem representada em diferentes resoluções de intensidade: a da esquerda, em 8 bits, e a da direita, em 2 bits 17
Figura 11 – Imagem em tons de cinza e seu histograma 18
Figura 12 – Aplicação do filtro de mediana 3x3 sobre o pixel marcado em vermelho: por possuir um valor extremo, seu valor foi substituído pela mediana dos valores dos pixels das proximidades. 19
Figura 13 – Exemplo de aplicação do filtro de mediana: a imagem da esquerda é a imagem original, e a da direita é o resultado da aplicação do filtro. 20
Figura 14 – Imagem de um avião contra o céu azul e aplicação de diferentes limiares para sua binarização. 21
Figura 15 – Exemplos de execução de algoritmo para rotulagem de áreas 22
Figura 16 – Exemplo de aplicação de máscara com a operação binária E 23
Figura 17 – Exemplo de aplicação da operação binária OU 23
Figura 18 – Vista geral da célula de aplicação de adesivo 25
Figura 19 – Histogramas a) da região onde se encontra o cordão de adesivo e b) apenas do cordão de adesivo 27
Figura 20 – Histogramas de diferentes pontos do cordão de adesivo 28
Figura 21 – Fluxograma do algoritmo inicial utilizado nos primeiros testes 28
Figura 22 – Primeiros resultados do algoritmo de segmentação 29
Figura 23 – a) Granularidade apresentada na imagem original. b) Mesma região após a aplicação de filtro de mediana 29
Figura 24 – Segmentação do cordão a) antes de aplicação do filtro de mediana e b) após a aplicação 30
Figura 25 – Fluxograma do algoritmo final implementado para avaliação das imagens 31

Figura 26 – Áreas detectadas realçadas em azul e detalhe da área identificada como descontinuidade 31

Figura 27 – Processo de manipulação da imagem para simulação de uma descontinuidade do cordão 32

Figura 28 – Imagem manipulada para simular descontinuidades na aplicação de adesivo e respectiva detecção de áreas pelo algoritmo 32

Figura 29 – Representação 3D da placa montada 34

Figura 30 – Representação 3D do conjunto desenhado no software FreeCAD, e ajuste de ângulo da câmera possibilitado pelo suporte criado.....34

Figura 31 – Componentes do protótipo do sistema de aquisição de imagens e conjunto montado 35

Figura 32 – Protótipo no modelo 3D da célula de aplicação de adesivo, e instalado na célula real 37

Figura 33 – Diagrama de tempo das trocas de sinais entre robô, CLP e SBC para captura de imagem.....38

Figura 34 – Comparação das telas de desenho das máscaras para a) imagem de referência e b) imagem capturada pelo protótipo 39

Figura 35 – Distribuição de resultados de áreas encontradas pelo algoritmo 40

Figura 36 – Imagens capturadas em momentos diferentes, com considerável variação na iluminação 41

Figura 37 – Relação entre intensidade nas imagens captadas e quantidades de áreas identificadas 41

Figura 38 – a. Reflexo das luminárias identificado como descontinuidade do cordão e b. luminárias de processo posicionadas atrás da câmera 42

Figura 39 – Luminárias e anteparo instalados para correção da iluminação 43

Figura 40 – Comparação das imagens anotadas na condição anterior e após as modificações 44

Figura 41 – Distribuição da quantidade de áreas encontradas pelo algoritmo após as modificações 45

Figura 42 – Distribuição das áreas dos cordões das imagens 45

Figura 43 – Detalhes do projeto 3D do suporte da câmera 65

LISTA DE ABREVIATURAS

CCD: charge-coupled device, 14
CLP: controlador lógico programável, 24
CMOS: complementary metal-oxide semiconductor, 14
CSI: Camera Serial Interface, 33
DIN: Deutsche Industrie Norm, 35
GPIO: General-purpose input and output, 33
PU: Poliuretano, 7, 8, 10
ROI: region of interest, 27
RTC: real-time clock, 36
SBC: Single Board Computer, 33

SUMÁRIO

1 Introdução 1
1.1 Objetivos 3
1.1.1 Objetivo geral 3
1.1.2 Objetivos específicos 4
1.2 Levantamento e Estudo Bibliográfico 4
1.3 Organização do texto 6
2 Referencial teórico 7
2.1 Aplicação automática de adesivos em sistemas robóticos 7
2.1.1 Propriedades do adesivo e condições de aplicação 7
2.1.2 Sistema automático de aplicação de adesivo 8
2.1.3 Modos de falha em aplicação de adesivos e detecção pelo sistema de aplicação 10
2.2 Sistemas de Visão por Computador 12
2.2.1 Câmeras 13
2.2.2 Representação de imagens em sistema digitais 16
2.2.3 Tratamento de imagens e extração de características 17
3 Materiais e métodos 24
3.1 Desenvolvimento do algoritmo para identificação da continuidade do cordão 26
3.2 Desenvolvimento de um protótipo para captação das imagens com interface com o CLP da linha 33
4 Resultados e discussões 39
5 Conclusões e Pesquisas Futuras 47
Referências 49
Apêndice A - Índice de Novos Negócios por montadora na plataforma Reclame Aqui 53
Apêndice B - Reclamações relacionadas a infiltração de água em automóveis 54
Apêndice C - Código fonte do módulo de inspeção de adesivo 56
Apêndice D - Placa para conexão entre sinais 3,3V do Raspberry Pi e 24V do CLP 63
Apêndice E - Desenvolvimento de um suporte com ângulo ajustável para módulo de câmera do raspberry Pi 65

1 INTRODUÇÃO

A indústria automotiva sempre foi extremamente relevante para a economia brasileira, representando aproximadamente 5% do produto interno bruto (PIB) brasileiro até 2018, possuindo capacidade instalada no Brasil de cinco milhões de veículos (DAUDT e WILLCOX, 2018, p. 185). Dados da Associação Nacional dos Fabricantes de Veículos Automotores (2021, p. 4, 50, 65) indicam que entre 2019 e 2020 foram produzidos no Brasil 4.708.825 veículos leves, sendo que 85% para o mercado nacional, posicionando o país como oitavo maior produtor de veículos do mundo.

Em dezembro de 2020, o preço médio entre os 13 veículos mais vendidos era de R$ 80.398,38 (RODRIGUES, 2021). Se considerarmos o salário médio dos trabalhadores de R$ 2.308,00 (IBGE, 2020, p. 1), um trabalhador precisaria empenhar cerca de 35 meses de seus rendimentos, ou seja, quase 3 anos, para a aquisição de um automóvel.

O tamanho desta indústria e quantidade de produtos vendidos, aliado a este elevado custo por unidade, se traduz em um alto nível de exigência de qualidade pelos consumidores, que tendem a buscar com maior frequência reparos de problemas identificados quando comparado a outros bens duráveis. Esta busca por reparos custa caro para as montadoras: em 2019 os custos com garantia em veículos foram de mais de 49 bilhões de dólares americanos, representando 2,5% da receita das montadoras no mundo (WARRANTY WEEK, 2020).

Além do custo financeiro devido aos reparos para sanar os problemas detectados pelos clientes, há ainda o custo que a baixa qualidade causa à empresa. Devido a insatisfação com o produto criada por defeitos percebidos pelos clientes e não solucionados a contento pelas montadoras, muitos proprietários recorrem à Reclame Aqui, a “maior plataforma de solução de conflitos entre consumidores e empresas da América Latina” (RECLAME AQUI, 2020), para expor publicamente sua insatisfação, e assim buscar ajuda para a solução do problema em seus veículos. Dados presentes na plataforma, compilados no Apêndice A, indicam que 36% dos clientes reclamantes não voltariam a fazer negócio com a montadora reclamada, o que representa um enorme potencial de redução nas vendas, agravada pela exposição negativa pública que tende a afastar potenciais novos compradores.

Dentre os problemas relatados pelos usuários da plataforma, um problema que causa grandes transtornos e consequente grande insatisfação é o de infiltração de água no interior do veículo, pois em geral, não são de resolução rápida devido à dificuldade no diagnóstico, sendo muitas vezes necessário desmontar o painel, carpetes e acabamentos. A infiltração de água é comumente percebida pelo cliente através de mau cheiro no interior do veículo e embaçamento dos vidros, decorrente da umidade, mas também podem ser percebida pela ocorrência de falhas elétricas, em sensores ou nos sistemas e módulos, devido à curtos-circuitos nos componentes elétricos e eletrônicos causados pela presença de água (CAYMAN AUTO SERVICES LTD, 2015).

Um dos principais pontos de infiltração de água é o para-brisas, que pode ocorrer por falha na aplicação do adesivo responsável pela fixação e vedação da peça. Isso faz que a água proveniente de chuvas e até mesmo da lavagem do veículo passem por estas falhas na vedação deixadas pela aplicação deficiente de adesivo (SARAFYAN e CATALDI, 2017).

Diante do grande impacto que a infiltração de água pelo para-brisa tem sobre a qualidade e consequentemente sobre os lucros das montadoras, grandes investimentos são feitos para garantir a qualidade da aplicação dos adesivos, tanto na escolha do processo de aplicação, em geral feito por robôs, quanto na escolha do adesivo utilizado, desenvolvido através de parcerias entre os fabricantes de adesivos e montadoras, e nos processos de garantia da qualidade através de testes de estanqueidade realizados em cabines que simulam chuvas intensas.

Mas mesmo estas precauções não são suficientes para garantir a qualidade em 100% dos veículos produzidos. Dados da plataforma Reclame Aqui compilados no Apêndice B indicam que entre 01/01/2020 e 04/04/2021, 85 reclamações relacionadas direta ou indiretamente à entrada de água pelo para-brisas foram feitas por consumidores, gerando um índice de insatisfação ainda maior do que a média geral: nestes casos, 46% dos reclamantes declararam que não voltariam a fazer negócios com a montadora.

A montadora Jaguar Land Rover, empresa de origem inglesa localizada no Cluster Automotivo Sul Fluminense, em Itatiaia, RJ, detectou em 2019 através do teste de estanqueidade uma série de problemas de infiltração de água através do para-brisas. Mesmo sendo detectadas internamente, não chegando às mãos do cliente final, estas falhas geraram grandes transtornos ao processo produtivo, pois o

processo de reparo é semelhante ao executado pelas concessionárias, sendo necessário desmontar os acabamentos para diagnóstico do problema, e posterior substituição do para-brisas.

Após as investigações internas confirmarem que as falhas eram devidas à problemas na aplicação do adesivo, os limites de controle do sistema de aplicação de adesivo foram estreitados para conter o problema. Esta ação permitiu detectar as falhas de aplicação de adesivo antes que o vidro fosse montado ao veículo, mas acabou gerando inúmeras paradas de produção pela detecção de falhas inexistentes pelo sistema de aplicação, tornando evidente a necessidade de uma outra forma de possibilitar a detecção de falhas na aplicação que não cause tanto impacto na produtividade.

1.1 OBJETIVOS

Para melhorar a detecção de falhas na aplicação sem impactar negativamente a produtividade, uma solução que vem sendo introduzida pela indústria é o controle qualidade da aplicação do adesivo realizado por sistemas de inspeção com uso de visão computacional. Nesta solução, imagens do cordão de adesivo são capturadas logo após sua aplicação no para-brisas e antes que seja colado ao veículo, e comparadas a um padrão de aceitação programado. Sistemas como este estão disponíveis comercialmente há alguns anos, mas com custo elevado. Embora este custo não seja expressivo quando comparado aos benefícios obtidos, ainda pode representar uma barreira à sua implementação devido à crise financeira.

1.1.1 OBJETIVO GERAL

O objetivo desta pesquisa é o de desenvolver um sistema de inspeção por visão computacional de baixo custo baseado no computador de placa única *Raspberry Pi* e uma câmera *Raspberry Pi Camera*, amplamente utilizados em aplicações de visão computacional por seu poder de processamento e qualidade de captura de imagens a um baixo custo, integrado ao sistema de controle da célula de aplicação de adesivo e com aplicação desenvolvida a partir de software livre, fazendo com que o custo total da solução proposta seja inferior a R$ 3.000,00, possibilitando sua replicação futura até mesmo em empresas de menor porte e com menor capacidade de investimento.

1.1.2 OBJETIVOS ESPECÍFICOS

Para atingimento do objetivo geral, os seguintes objetivos específicos foram definidos:

- Definição dos requisitos para aprovação do sistema de inspeção junto à engenharia de manufatura;
- Montagem de um hardware para aquisição e processamento de imagens do cordão de adesivo aplicado ao para-brisas;
- Desenvolvimento e montagem de uma placa de interface de sinais entre o *Raspberry Pi* e o controlador da célula de aplicação de adesivos, fabricada com componentes facilmente encontrados no mercado de componentes eletrônicos;
- Desenvolvimento de um algoritmo para captura e processamento das imagens, em linguagem *Python*, fazendo uso de bibliotecas de software de código aberto, como *OpenCV*;
- Avaliação do sistema desenvolvido para garantir o cumprimento dos requisitos.

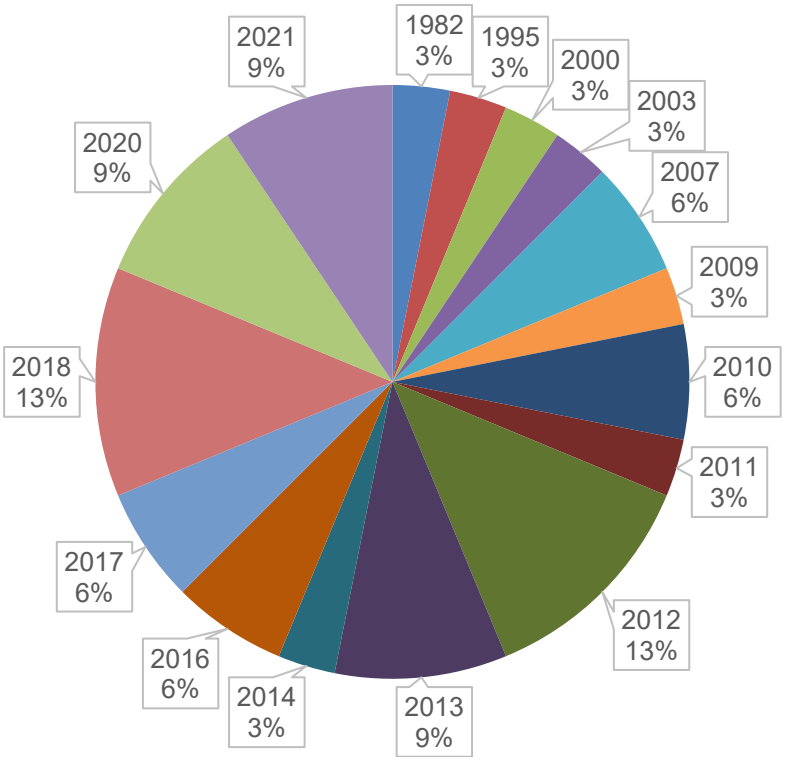
1.2 LEVANTAMENTO E ESTUDO BIBLIOGRÁFICO

Com o objetivo de dar embasamento teórico e científico à pesquisa, foi realizado um extenso levantamento e estudo bibliográfico. Algumas análises das publicações analisadas são apresentadas a seguir, de forma a demonstrar numericamente a sua distribuição ao longo do tempo, por fonte e por assunto, com o uso de gráficos de setores.

Na Figura 1 pode ser observada a distribuição ao longo do tempo das publicações analisadas.

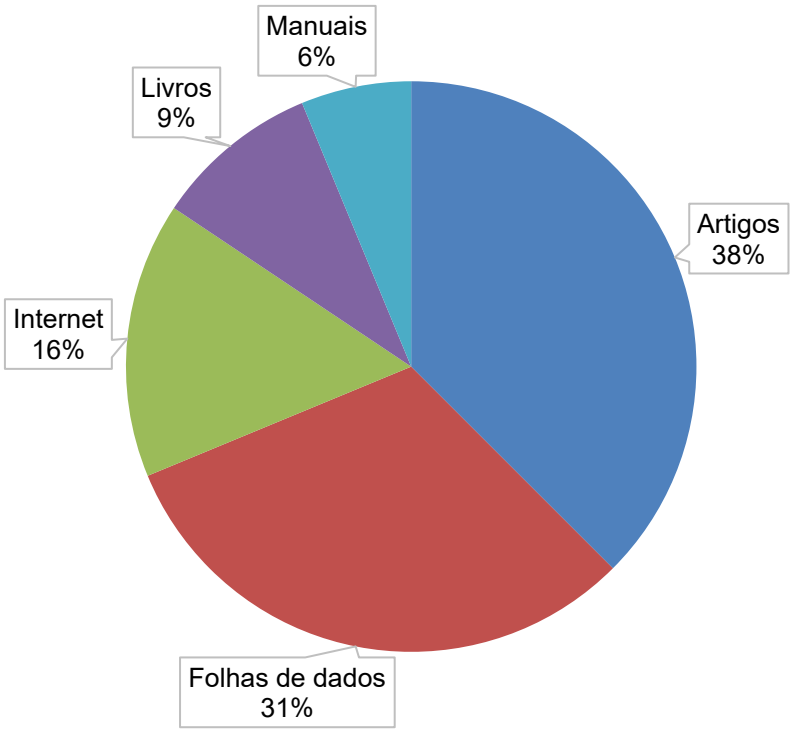
Na Figura 2 pode ser vista a distribuição das publicações analisadas por fontes de pesquisa.

Figura 1 – Distribuição das publicações ao longo do tempo



Fonte: Elaborado pelo autor

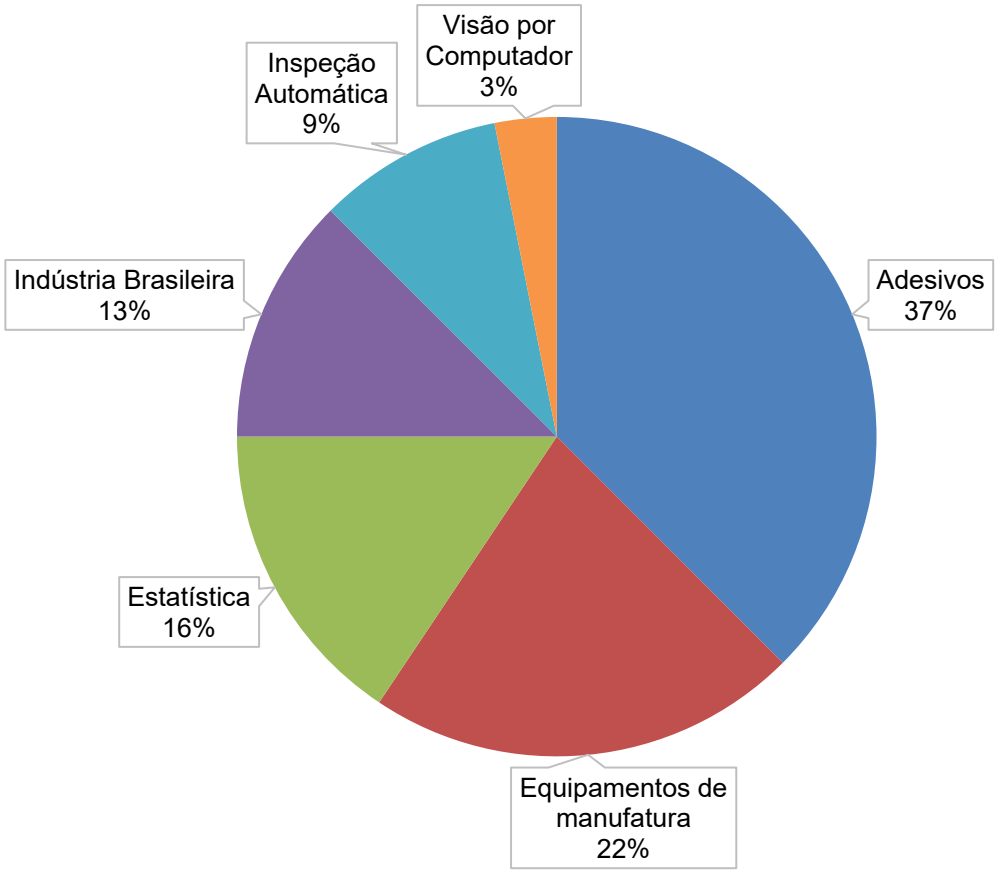
Figura 2 – Distribuição das publicações por tipo



Fonte: Elaborado pelo autor

Também foi avaliada a distribuição das publicações em relação aos assuntos abordados, como pode ser visto na Figura 3.

Figura 3 – Distribuição das publicações por assunto



Fonte: Elaborado pelo autor

1.3 ORGANIZAÇÃO DO TEXTO

A presente dissertação está organizada em 5 capítulos: o Capítulo 1 - Introdução, apresenta o problema de forma abrangente, seu contexto na indústria e define os objetivos da pesquisa; o Capítulo 2 - Referencial teórico, apresenta uma breve introdução teórica dos temas essenciais ao desenvolvimento da pesquisa; o Capítulo 3 - Materiais e métodos, apresenta as várias etapas do desenvolvimento do projeto; o Capítulo 4 - Resultados e discussões, detalha os diversos testes realizados para refinamento e validação do sistema proposto e apresenta os resultados obtidos; e o Capítulo 5 - Conclusões e Pesquisas Futuras, apresenta a conclusão do projeto desenvolvido e sugere pesquisas futuras.

2 REFERENCIAL TEÓRICO

Nesta seção são apresentados os principais conceitos utilizados para a elaboração desta pesquisa, a partir da revisão da literatura disponível sobre os assuntos envolvidos. Na seção 2.1 a seguir, são apresentadas as propriedades do adesivo utilizado na colagem de vidros automotivos, o processo de aplicação do adesivo e os modos de falhas possíveis para este processo. Na seção 2.2 são apresentados os conceitos relacionados à visão por computador, como o hardware necessário à aquisição de imagens, a representação destas imagens em sistemas computacionais e algoritmos comumente utilizados para o tratamento destas imagens e extração de características úteis para o processo em questão.

2.1 APLICAÇÃO AUTOMÁTICA DE ADESIVOS EM SISTEMAS ROBÓTICOS

O uso de adesivos vem aumentando ao longo dos anos no setor automotivo, por serem uma alternativa eficiente para junção de peças.

Para garantir a estabilidade na aplicação do adesivo, robôs são utilizados para aplicar de forma automática adesivos com base de poliuretano (EMERALD GROUP PUBLISHING LIMITED, 2007).

Esta seção aborda as propriedades dos adesivos utilizados para esta aplicação, os sistemas robóticos utilizados para esta atividade, e potenciais modos de falha e suas consequências para a qualidade do automóvel.

2.1.1 PROPRIEDADES DO ADESIVO E CONDIÇÕES DE APLICAÇÃO

O poliuretano (PU) é um polímero com excelentes propriedades mecânicas, entre elas sua alta capacidade de alongamento e absorção de energia, alta resistência a ambientes agressivos, estabilidade térmica, facilidade de aplicação e relação custo-benefício (SOMARATHNA, RAMAN, *et al.*, 2018, p. 996).

Os adesivos com base em PU começaram a ser desenvolvidos na década de 1950 e devido à sua versatilidade, em pouco tempo começaram a ser utilizados para a colagem de diversos substratos como vidros, madeira, plásticos e cerâmica.

Atualmente adesivos de PU podem ser encontrados em uma grande gama de aplicações devido à suas características como adesão, flexibilidade, comportamento em baixas temperaturas e possibilidade de ajuste da velocidade de cura, de acordo

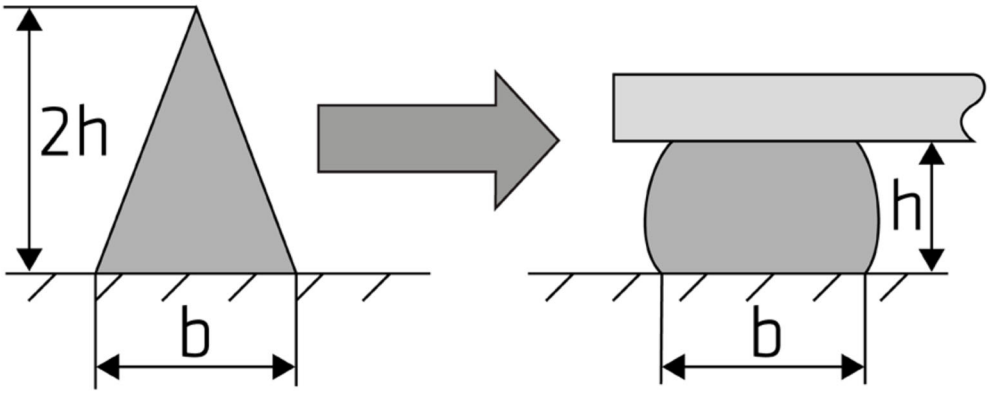
com as necessidades da aplicação, e do agente de cura, que pode ser aquecimento, adição de um solvente ou água.

A cura pela reação com água pode ocorrer também a partir do contato com a umidade do ar (SZYCHER, 2013, p. 393-395), sendo este um método comumente utilizado em adesivos automotivos.

Para permitir uma maior velocidade de aplicação, necessária ao alto volume de produção da indústria automotiva, adesivos de PU são ajustados através de aditivos para garantir uma viscosidade adequada para aplicação pelos sistemas automáticos, em temperaturas que podem variar de 20°C a 85°C.

O formato do cordão de adesivo também influencia no resultado da colagem, e para garantir uma espessura uniforme da linha de colagem, é recomendado o uso de um cordão triangular, como ilustrado na Figura 4 (SIKA, 2020, p. 1-2).

Figura 4 – Forma do cordão recomendada para aplicação de adesivo PU



Fonte: SIKA (2020)

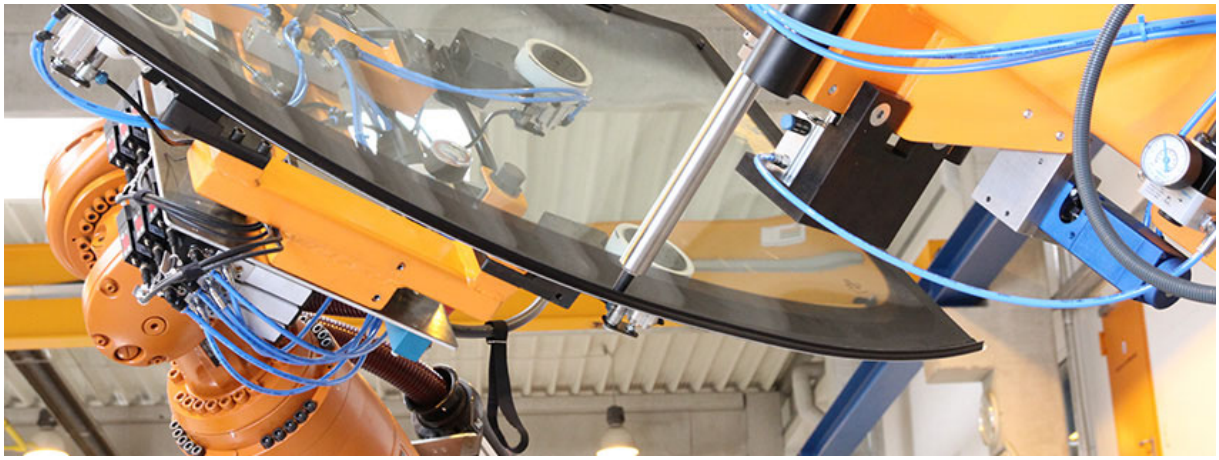
A seguir, são apresentados os sistemas automáticos utilizados na aplicação do adesivo aos vidros automotivos.

2.1.2 SISTEMA AUTOMÁTICO DE APLICAÇÃO DE ADESIVO

Braços robóticos têm sido amplamente utilizados na indústria automotiva por anos, por serem ferramentas versáteis e que podem realizar trabalhos repetitivos com grande precisão, velocidade e repetibilidade, características essenciais em processos produtivos exigentes como os relacionados à manufatura de automóveis. Estas características os tornam essenciais na aplicação de adesivos, tarefa que exige posicionamento preciso do cordão de adesivo, bem como alta velocidade de aplicação para atender aos elevados volumes de produção (MORTIMER, 2004, p. 264,269-270).

Neste tipo de aplicação, o braço robótico dispõe de ventosas que são utilizadas para pegar o vidro a partir de mesas centralizadoras, que garantem a repetibilidade em seu posicionamento. O vidro é então movimentado pelo braço, que posiciona a área onde será aplicado o cordão de adesivo sob o bico de aplicação (Figura 5). A movimentação da peça sob o bico deve ocorrer de forma sincronizada com o posicionamento do bico e fluxo do adesivo, para garantir o formato, altura e largura constantes do cordão de adesivo (ATLAS COPCO, 2019).

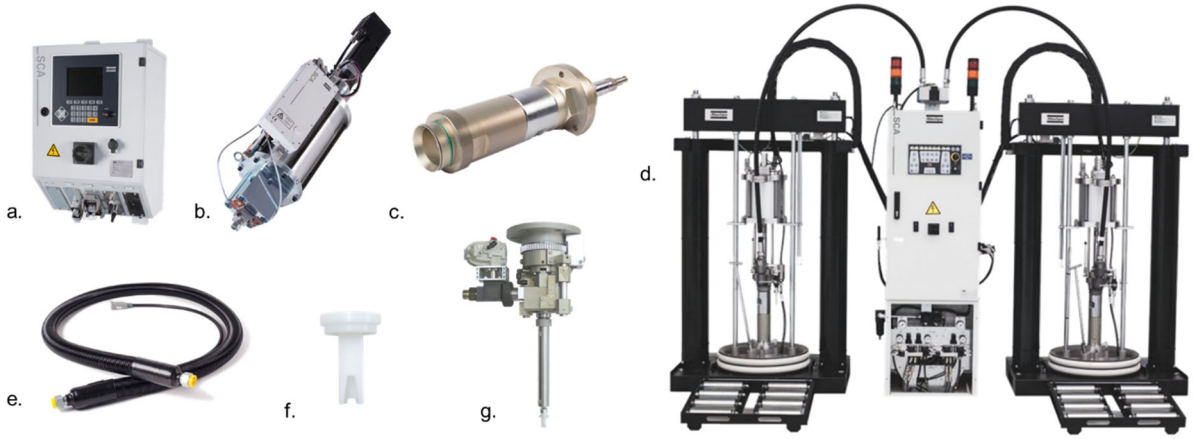
Figura 5 – Aplicação de adesivo em para-brisas automotivo



Fonte: ASM DIMATEC GMBH. (2019)

O controle e monitoramento do fluxo de adesivo aplicado à peça se dá através de um sistema dedicado, composto pelos seguintes componentes, ilustrados na Figura 6: a) controlador, que monitora constantemente o fluxo e pressão do adesivo, e através de um algoritmo atua no dosador para garantir o fluxo constante de acordo com a quantidade programada, realizando também a função de controlar a temperatura do adesivo através do controle de aquecedores posicionados nos diversos elementos do sistema de aplicação; b) dosador, que incorpora um sistema de medição de fluxo e pressão do adesivo, sinais estes que são usados para retroalimentar o controlador; d) unidade de bombeamento, composta por uma c) bomba de alta pressão, que opera entre 15 a 25MPa, válvulas de comutação e controlador das bombas; e) prato da bomba, mangueiras e válvulas com aquecimento, para garantir a viscosidade adequada do adesivo para bombeamento e aplicação; f) bico com formato de acordo com o adesivo a ser aplicado; e g) posicionador robótico do bico, que conectado ao controlador do braço robótico, posiciona o bico de aplicação na direção correta para permitir o formato adequado do cordão de adesivo (ATLAS COPCO, 2019).

Figura 6 – Componentes do sistema de controle e monitoramento da aplicação de adesivo



Fonte: ATLAS COPCO (2019)

O controlador do braço robótico e o controlador do sistema de aplicação de adesivo se comunicam através de rede do tipo *field bus*, para seleção do volume alvo de adesivo conforme a peça sendo manipulada e sincronismo do ponto de abertura e fechamento de válvulas e do fluxo de adesivo, de acordo com a posição e velocidade do vidro em relação ao bico de aplicação (KUKA ROBOTER GMBH, 2012, p. 7-8).

Na seção seguinte, são abordados os potenciais modos de falha na aplicação de adesivos pelos sistemas automáticos, os mecanismos utilizados para detectar tais falhas e a forma como é realizada a correção do cordão de adesivo.

2.1.3 MODOS DE FALHA EM APLICAÇÃO DE ADESIVOS E DETECÇÃO PELO SISTEMA DE APLICAÇÃO

A correta colagem dos vidros automotivos através de adesivos de PU garante a fixação da placa de vidro à carroceria e sua alta resistência a impacto e intempéries como chuva, vento e diferentes temperaturas. Uma colagem inadequada pode interferir até mesmo no funcionamento de itens de segurança, como o *air bag*, além de permitir a infiltração de água no interior da cabine (SAINT-GOBAIN, 2018).

O primeiro ponto a ser observado para garantir uma colagem adequada é a adesão do adesivo aos materiais a serem colados. A falha de adesão, como é chamado este modo de falha, pode ser causada por contaminantes presentes nos materiais, como poeira, que pode se acumular no transporte e armazenamento, e graxas e óleos utilizados para evitar a corrosão de carrocerias, ou por incompatibilidade entre o adesivo e o material a ser colado (SHIN-ETSU CHEMICAL, 2018, p. 3).

O sistema de aplicação não possui meios para detectar falhas de adesão, por isso elas devem ser evitadas através de processos bem documentados e periodicamente verificados, que incluam a limpeza das superfícies a serem coladas de quaisquer contaminantes, e em alguns casos, deve ser especificada a aplicação de um primer para promover uma melhor adesão entre o adesivo e os materiais a serem colados. Testes de compatibilidade devem ser realizados para determinar a eficiência do adesivo para os materiais a serem colados, e são realizados à exaustão durante o desenvolvimento de um adesivo para uso na colagem de vidros automotivos (SIKA, 2020, p. 2).

Em relação ao cordão de adesivo, os principais defeitos que podem ocorrer são a descontinuidade do cordão, quando há uma interrupção do cordão de adesivo durante a aplicação, deixando um espaço sem adesivo por onde a água poderia entrar; e a variação excessiva da largura ou altura do cordão. Ambos podem ser causados por entupimentos no sistema de aplicação, levando a uma redução ou interrupção temporária do fluxo de adesivo (YAN, XU, *et al.*, 2016, p. 2816).

O sistema de aplicação pode detectar variações na pressão do adesivo durante a aplicação causadas pelo entupimento do sistema de aplicação. Entretanto, outras variações de pressão, como as causadas pela variação da viscosidade do adesivo entre lotes, podem ser confundidas com as variações causadas por entupimentos, levando a constantes paradas de produção. Uma seleção cuidadosa dos limites de pressão deve ser realizada de forma a detectar a maior quantidade possível de interrupções no cordão sem comprometer a produtividade (SCA SCHUCKER GMBH & CO. KG, 2013, p. 276-278).

Caso uma falha de continuidade do cordão ocorra e não seja detectada pelo sistema de aplicação ou pelo operador, ela pode ainda ser detectada no teste de estanqueidade ao qual os veículos são submetidos. Nele, o veículo passa por uma cabine onde jatos de água são direcionados ao veículo vindos de diversas direções, com o intuito de simular uma chuva muito intensa.

Em geral, falhas de continuidade do cordão de adesivo criam uma falha na vedação do para-brisas em relação ao ambiente externo, que possibilita a entrada de água durante este teste. Se detectados veículos nestas condições, eles precisam ser retirados do fluxo de produção para que o para-brisas seja removido através do corte do adesivo. O adesivo é então reaplicado, o cordão é inspecionado e o vidro é novamente colado ao veículo.

Este processo de retrabalho causa grandes transtornos aos fabricantes de veículos, pois antes de executar o retrabalho é necessário garantir que a causa do vazamento é o adesivo de fixação do para-brisas, e para isso diversas peças precisam desmontadas, entre elas carpetes, acabamentos e até mesmo o painel de instrumentos. Após o reparo, o veículo ainda precisa passar por um processo de secagem, para evitar mal cheiro devido à presença de umidade, e só então é remontado. Isto pode fazer que o veículo fique por dias fora do processo produtivo, atrasando sua venda, aumentando o estoque de veículos no processo e causando insatisfação aos clientes.

Na seção seguinte são apresentados os conceitos de visão por computador, utilizados para desenvolver a solução proposta nesta pesquisa.

2.2 SISTEMAS DE VISÃO POR COMPUTADOR

A visão por computador tenta prover aos sistemas automáticos meios de “ver” o ambiente, e assim reconhecer formas, objetos e condições que sejam relevantes para uma determinada aplicação.

Em visão por computador tentamos descrever o mundo que vemos em uma ou mais imagens e reconstruir suas propriedades, como forma, iluminação e distribuição de cores (SZELISKI, 2010, p. 5). Desta forma a visão por computador “procura produzir descritores inteligentes e úteis de cenas e sequências, e dos objetos que as compõem, realizando operações nos sinais recebidos de câmeras de vídeo.” (DAUGMAN, 2010, p. 3).

Para Shapiro e Stockman (2000, p. 13),

> o objetivo da visão computacional é tomar decisões úteis sobre objetos físicos e cenas reais baseado em imagens adquiridas. Para tomar decisões sobre objetos reais, quase sempre é necessário construir descritores ou modelos destes objetos a partir de imagens. Por isso, muitos especialistas dirão que o objetivo da visão computacional é a construção de descritores de cenas a partir de imagens.

Krishna (2017, p. 19) decompõe o processo para a geração de tais descritores em duas etapas: na primeira, o dispositivo de detecção captura os detalhes da imagem e transmite ao computador. Câmeras podem detectar sinais além do espectro visível aos olhos humanos, abrindo muitas possibilidades para a visão computacional. Na segunda etapa, o dispositivo de interpretação processa os sinais obtidos do dispositivo de captura, e extrai informações significativas.

Dentre as informações relevantes para os processos industriais que podem ser obtidas por sistemas de visão computacional estão a medição de distâncias, verificação de integridade e controle de qualidade (GOLNABI e ASADPOUR, 2007, p. 632). Estas informações obtidas pelos sistemas de visão por computador são usadas para inspeção, que é o processo de determinar se um produto desvia de dado conjunto de especificações (NEWMAN e JAIN, 1995).

Comparando os sistemas de inspeção baseados em visão por computador aos inspetores humanos realizando inspeção visual, é possível perceber que apesar da flexibilidade proporcionada pela visão humana na detecção de características em diferentes situações e condições, como variações de luminosidade e posicionamento da peça a ser inspecionada, a precisão da inspeção visual humana diminui com trabalhos monótonos e rotineiros, que são características inerentes aos sistemas de produção em massa. Isto torna a inspeção baseada em visão por computador uma alternativa óbvia à inspeção por inspetores humanos (CHIN e HARLOW, 1982).

Sistemas de inspeção baseados em visão computacional também proporcionam resultados consistentes e estatisticamente significativos, levando a uma maior compreensão do estado do processo sendo inspecionado, redução de intervenções humanas nos processos produtivos para análise e segregação de peças defeituosas e garantia de uma qualidade consistente dos produtos (DI LEO, LIGUORI, *et al.*, 2016).

Desta forma, o uso de visão por computador na indústria melhora a produtividade e o gerenciamento da qualidade, proporcionando uma vantagem competitiva às empresas que empregam esta tecnologia (MALAMAS, PETRAKIS, *et al.*, 2003).

Nesta seção são apresentados os principais tópicos relacionados aos sistemas de visão computacional utilizados no desenvolvimento desta pesquisa.

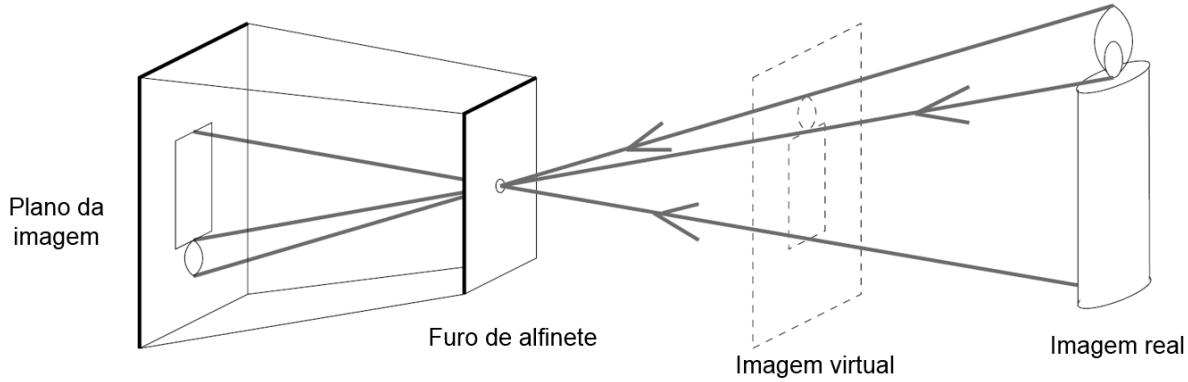
2.2.1 CÂMERAS

O primeiro passo para a visão computacional é a aquisição de imagens. Existem diversos tipos de dispositivos para aquisição de imagens como câmeras de vídeo, infravermelho e radiotelescópios, e eles podem ou não ser equipados com lentes.

As câmeras, como são conhecidos os dispositivos de aquisição de imagens comumente encontrados em sistemas de vigilância, fotografia pessoal e profissional,

telefones celulares e laptops, derivam seu nome de um dispositivo criado no século XVI chamado *camera obscura*, que em latim significa câmera escura. Este dispositivo consistia em uma caixa, ou câmera, totalmente escura em seu interior, com um pequeno furo de alfinete em uma das paredes para captar luz, e na extremidade oposta, uma placa translúcida, que permitia observar a imagem captada pelo dispositivo, como ilustrado na Figura 7. Mais tarde, o furo de alfinete foi substituído por lentes para focalizar as imagens e concentrar a luz, técnica ainda utilizada nas câmeras modernas (FORSYTH e PONCE, 2012, p. 3).

Figura 7 – Modelo da "camera obscura"



Fonte: FORSYTH e PONCE (2012, p. 4)

De forma geral, os princípios de funcionamento e forma construtiva das câmeras atuais permanecem os mesmo desde a *camera obscura*. A principal característica introduzida nas câmeras utilizadas nos sistemas de visão é a substituição da placa translúcida onde a imagem é formada por sensores capazes de transformar sinais ópticos em eletricidade, e posteriormente, em dados. Os sensores utilizados nestas câmeras são tipicamente fabricados a partir de fotodiodos alinhados arranjados de forma matricial, com células CCD ou células padrão CMOS. Apesar dos dois tipos de células apresentarem características similares, o padrão CMOS é o mais utilizado devido a características de manufatura.

O projeto de sensores para captação de imagem leva em conta os objetivos da aplicação específica na qual serão usados, proporcionando diferentes níveis de sensibilidade e qualidade de imagem. O tamanho de cada fotodiodo que compõe o sensor influencia diretamente na quantidade de luz que pode ser captada por ele. Fotodiodos muito pequenos proporcionam pouca imunidade a ruídos, e limitada capacidade de captar luz nas frequências mais altas, seguindo a teoria das amostragens. Entretanto os custos de produção dos sensores aumentam

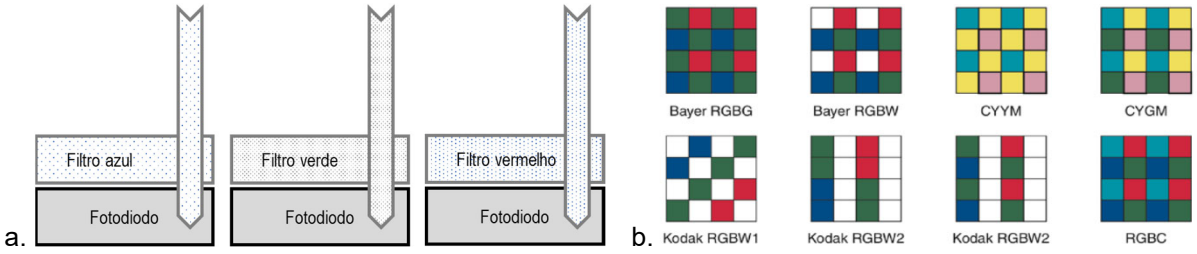
proporcionalmente ao tamanho dos seus fotodiodos. Desta forma, fabricantes buscam uma otimização entre custo e qualidade ao desenvolver seus sensores sendo que em sensores comercialmente disponíveis o tamanho de cada célula é de um mícron ou maior.

Outro fator que impacta diretamente no tamanho dos sensores é a sua resolução, que é a medida da quantidade de informações posicionais que podem ser extraídas de uma imagem. Cada posição ou ponto individual de informação extraído de uma imagem é chamado de pixel. Como os sensores são geralmente produzidos em forma retangular, sua resolução é dada em duas dimensões, que são a quantidade de pixels na horizontal e a quantidade de pixels na vertical. O produto destas duas dimensões também é utilizado para especificar a resolução espacial das câmeras, sendo que nas câmeras modernas é comum encontrar resoluções entre 8 e 30 megapixels (milhões de pixels).

Os fotodiodos mais comumente encontrados em câmeras de aquisição de imagens são fabricados a partir de silício, que tem uma resposta não-linear para as diferentes frequências do espectro, sendo que o pico de resposta se encontra na região próxima ao infravermelho. Sendo assim, o projeto dos sensores deve levar em consideração esta não-linearidade para compensar as diferenças de resposta para cada faixa de frequência.

Em sensores multiespectrais, nos quais diferentes faixas de cores são lidas de forma independente, como no caso dos sensores coloridos, o método mais empregado é o do mosaico, no qual filtros para cada cor são arranjados sobre as células de forma a permitir a passagem de um comprimento de onda específico (Figura 8a). O arranjo dos filtros no mosaico possibilita, através do uso de mais elementos de uma determinada faixa, dar ênfase na captação de um determinado comprimento de onda (Figura 8b), característica que pode ser utilizada para compensar as diferenças de resposta citadas anteriormente (KRIG, 2014, p. 1-5).

Figura 8 – Método de mosaico utilizado na captação independente de distintos comprimentos de onda



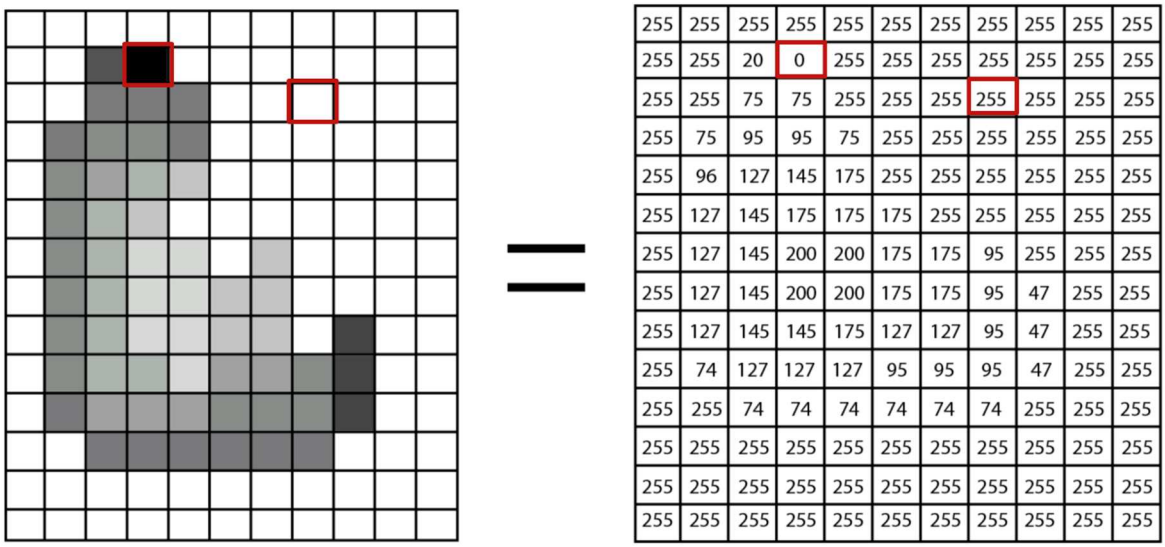
Fonte: KRIG (2014, p. 5)

As formas de representação digital das imagens obtidas são apresentadas a seguir.

2.2.2 REPRESENTAÇÃO DE IMAGENS EM SISTEMA DIGITAIS

Para que o sistema computacional possa tratar as imagens, é necessário que sejam convertidas em informações inteligíveis por estes sistemas, em geral, digitais. Desta forma, os sinais captados de forma analógica pelos sensores são convertidos em valores discretos, com uma resolução de intensidades igual a 2^n, onde n é a quantidade de bits utilizada para representar cada pixel. Pelo fato de os sensores serem produzidos em matrizes retangulares, a representação mais direta é a de uma matriz bidimensional, na qual cada pixel é representado por um elemento da matriz. Na Figura 9 pode ser vista uma imagem em tons de cinza, representada em uma matriz de 11x14 elementos de 8 bits, com valores que variam de 0, representando o tom mais escuro, a 255, representando o tom mais claro.

Figura 9 – Representação em 8 bits de imagem em tons de cinza



Fonte: UNIVERSITY OF TORONTO (2018)

A representação em uma maior quantidade de bits possibilita uma melhor distinção de detalhes da imagem, como pode ser visto na Figura 10, na qual a mesma imagem é representada em duas diferentes resoluções de intensidades: 8 e 2 bits. A melhora na representação da imagem tem um custo no aumento do tempo de processamento e espaço necessário para armazenamento da imagem (BARELLI, 2018, p. 36).

Figura 10 – Imagem representada em diferentes resoluções de intensidade: a da esquerda, em 8 bits, e a da direita, em 2 bits

Fonte: BARELLI (2018, p. 36)

A seguir, são apresentados os processos necessários para a obtenção de informações a partir da representação digital da imagem capturada.

2.2.3 TRATAMENTO DE IMAGENS E EXTRAÇÃO DE CARACTERÍSTICAS

A partir das imagens captadas, o sistema de visão precisa processar os dados recebidos de forma a compensar distorções conhecidas dos sensores e lentes, e maximizar as características da imagem que sejam significativas para a obtenção de informação. Diversas técnicas computacionais são empregadas para minimizar os ruídos inerentes às condições em que a imagem foi captada, que podem ser causados pela iluminação inadequada, pelo baixo contraste entre o objeto principal e o fundo, ou até mesmo pelas características do sensor.

Dois esquemas de inspeção visual são comumente utilizados nos sistemas de visão. Um é a correspondência de modelo, no qual um conjunto de imagens padrão são utilizadas para determinar similaridades entre as imagens e a partir delas extrair características que serão comparadas com as imagens a serem inspecionadas. O outro é a verificação baseada em regras, que envolve o uso de um conjunto de regras predefinidas para determinar quais características são relevantes e quais são as condições para classificar as imagens avaliadas como pertencentes a peças boas ou ruins (SUN, SUN e SURGENOR, 2012).

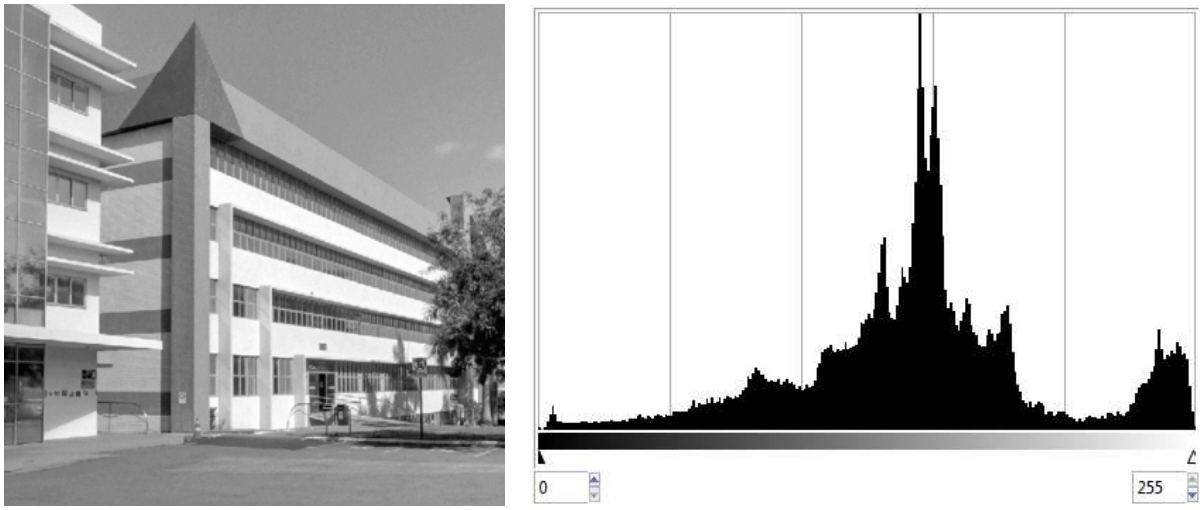
O esquema utilizado nesta pesquisa se baseia em regras que utilizam características como intensidade luminosa e conectividade.

A seguir são apresentadas algumas das técnicas para tratamento das imagens e extração das características relevantes.

2.2.3.1 Histograma

Uma ferramenta da estatística utilizada em sistemas de visão computacional é o histograma, que é uma representação gráfica da ocorrência de um determinado valor ou gama de valores dentro do conjunto de dados. O histograma é construído colocando os valores da variável de interesse no eixo horizontal, e a frequência com que ocorrem no eixo vertical (ANDERSON, SWEENEY e WILLIAMS, 2011, p. 41-42). Sua aplicação em processamento de imagens consiste em contabilizar a frequência da ocorrência de cada valor de intensidade na imagem. A figura resultante demonstra os picos de intensidades mais frequentes e os vales com os valores de intensidade que menos ocorrem na imagem (Figura 11).

Figura 11 – Imagem em tons de cinza e seu histograma



Fonte: Adaptado de BARELLI (2018, p. 36).

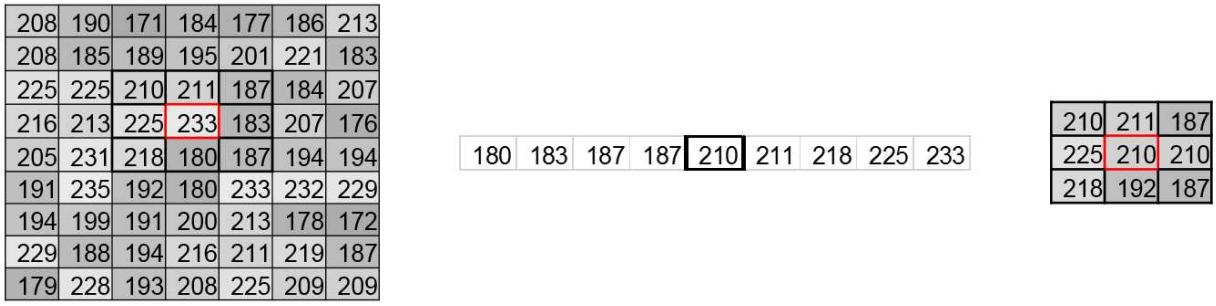
A distribuição dos níveis de intensidade em determinada imagem ou parte da imagem pode ser utilizada como uma espécie de assinatura para identificar de forma rudimentar uma cena previamente conhecida. Por ser uma característica cujo cálculo apresenta baixo custo computacional, é bastante utilizada em conjunto com outras técnicas para identificação de objetos em uma imagem (FORSYTH e PONCE, 2012, p. 76-77).

2.2.3.2 Filtros

Assim como sinais unidimensionais, imagens podem ser filtradas com diversos tipos de filtros, como passa-baixas, utilizado para atenuar ruídos, e passa-altas, utilizados para destacar contornos na imagem.

Um filtro com excelente resposta a ruídos provenientes do processo de captação de imagens é o filtro de mediana. Nele, valores extremos de intensidade dos pixels são substituídos por valores encontrados a partir da mediana dos pixels das proximidades. Na Figura 12, pode ser vista a aplicação de um filtro de mediana com núcleo de 3x3 pixels. É importante ressaltar que, devido ao fato de utilizar a mediana de um intervalo, o núcleo utilizado pelo filtro deve sempre ser composto por uma matriz de tamanho ímpar (OPENCV, 2021).

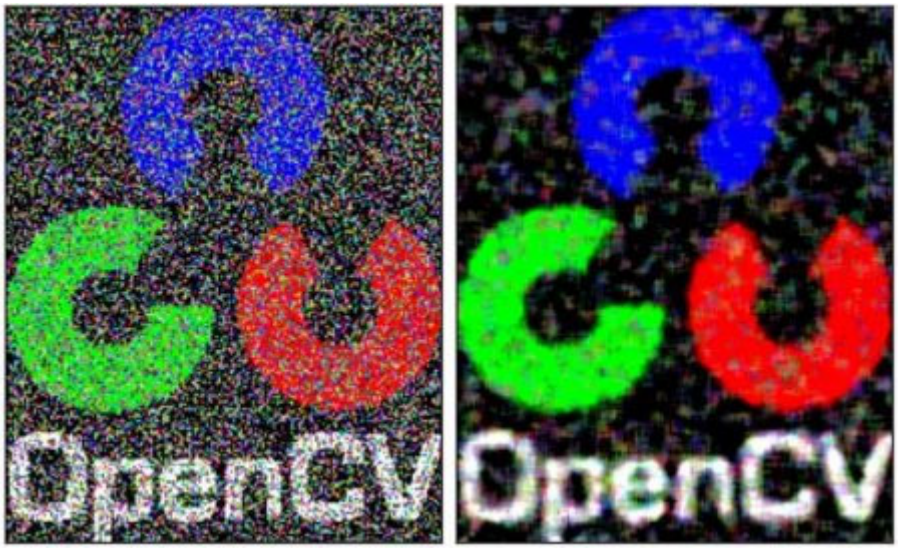
Figura 12 – Aplicação do filtro de mediana 3x3 sobre o pixel marcado em vermelho: por possuir um valor extremo, seu valor foi substituído pela mediana dos valores dos pixels das proximidades.



Fonte: Elaborado pelo autor.

Na Figura 13 pode ser vista a aplicação do filtro de mediana a uma imagem modificada artificialmente para apresentar ruído do tipo “sal e pimenta”, que são ruídos aleatórios introduzidos no processo de captura de imagem. Na imagem do exemplo, 50% dos pixels originais foram substituídos por ruído, resultando na imagem da esquerda. Na imagem à direita pode ser visto o resultado da aplicação do filtro de mediana, na qual é possível distinguir com mais clareza o logotipo original, embora devido ao alto nível de ruído introduzido, a imagem resultante ainda apresente forte distorção.

Figura 13 – Exemplo de aplicação do filtro de mediana: a imagem da esquerda é a imagem original, e a da direita é o resultado da aplicação do filtro.



Fonte: (OPENCV, 2021)

Comparado com outros tipos de filtros espaciais, como o Gaussiano, o filtro de mediana apresenta desfocamento da imagem apenas marginal, apesar de introduzir uma certa perda de detalhes da imagem e uma aparência suavizada. O bom balanço entre redução de ruídos e desfocamento da imagem faz com que o filtro de mediana seja um dos mais utilizados em aplicações gerais de processamento de imagens (DAVIES, 2012, p. 44).

2.2.3.3 Binarização (*threshold*)

Uma técnica muito utilizada na segmentação de imagens, ou seja, na separação do objeto de estudo do fundo da imagem, é a binarização a partir de um limiar de intensidade, conhecida pelo termo em inglês *threshold*. Nela, o objetivo é encontrar um limiar de intensidade a partir do qual todos os pixels cujas intensidades estejam acima deste limiar sejam declarados como pertencentes a uma classe, que pode ser o objeto ou o fundo da imagem, e os pixels iguais ou abaixo deste limiar seriam pertencentes à outra classe.

A simplicidade matemática do método vem acompanhada pela dificuldade em encontrar o limiar ideal para a separação entre o objeto e o fundo, pois limiares elevados podem incluir outros objetos além do fundo, enquanto limiares muito baixos podem fazer com que partes do objeto sejam deixadas de fora na segmentação. Na Figura 14 é ilustrada esta situação, em que para um limiar de 115, parte do leme do

avião não foi incluída na imagem segmentada, enquanto para um limiar de 167, muitas nuvens, que pertencem ao fundo, foram incluídas na imagem segmentada.

Figura 14 – Imagem de um avião contra o céu azul e aplicação de diferentes limiares para sua binarização.

Fonte: Adaptado de PXHERE (2017)

Outro ponto importante é que a variação na iluminação da cena fotografada causa uma variação na luminosidade da imagem capturada, e partes de um mesmo objeto podem apresentar diferenças significativas de luminosidade, dificultando ou impossibilitando a tarefa de se encontrar um limiar global que possa ser aplicado a toda a imagem. Uma forma de se superar esta dificuldade é a separação da imagem em diversas subimagens, encontrar e aplicar o limiar de forma independente a cada uma delas. Esta abordagem é chamada de limiarização local, na qual os histogramas das subimagens são analisados de forma independente para localização do limiar ideal. (DAVIES, 2012, p. 85-86,92).

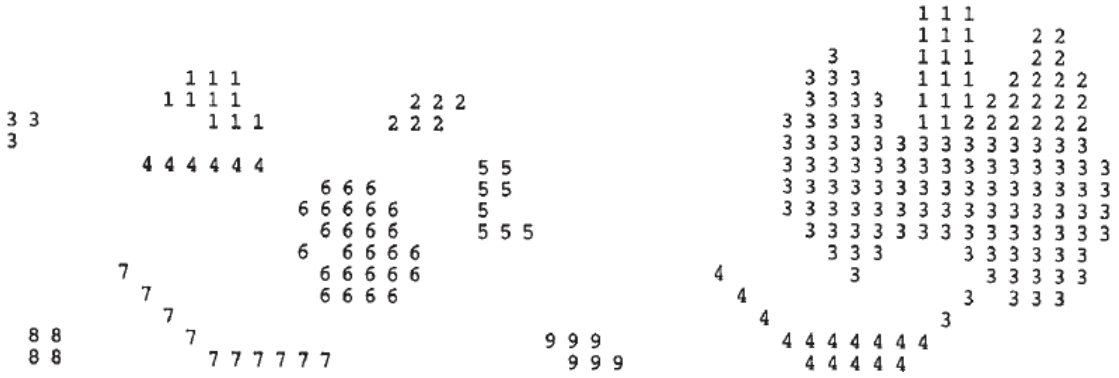
2.2.3.4 Identificação de áreas conectadas

Após a segmentação dos objetos de uma cena em relação ao fundo, a etapa seguinte para a realização de análise dos objetos é a identificação de áreas conectadas, de modo a identificar e distinguir os diferentes objetos que compõem a cena. O primeiro passo para atingir tal objetivo é definir os parâmetros para considerar uma área como conectada. Sendo as imagens binarizadas formadas por conjuntos retangulares de zeros e uns, é necessário assumir que as áreas pertencentes ao objeto segmentado sejam consideradas conectadas se quaisquer dos pixels vizinhos, inclusive os das diagonais, possuírem o mesmo valor, enquanto as áreas pertencentes ao fundo sejam consideradas interligadas apenas se os pixels da vizinhança imediata, desconsiderando os das diagonais, possuírem o valor

considerado do fundo. Desta forma, a conectividade do objeto é de 8 vias, e do fundo, de 4 vias.

Com a definição de conectividade claramente estabelecida, é relativamente simples implementar um algoritmo para percorrer todos os pixels e identificar quais deles são conectados entre si, atribuindo rótulos a cada região conectada, como ilustrado na Figura 15. Para o caso de formas mais complexas, pode ocorrer de a um mesmo objeto conectado serem atribuídos diferentes rótulos. Nestes casos, um pós processamento é necessário para identificar quais dos rótulos são conectados entre si e assim criar uma tabela de tradução entre rótulos (DAVIES, 2012, p. 231-238).

Figura 15 – Exemplos de execução de algoritmo para rotulagem de áreas

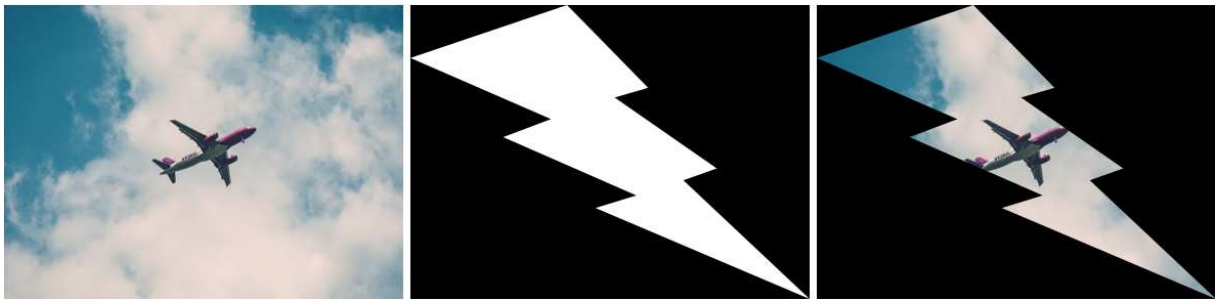


Fonte: DAVIES (2012, pp. 232,233).

2.2.3.5 Operações em lógica booleana

Operações em lógica booleana como OU e E também podem ser aplicadas a imagens. A operação E pode ser usada para aplicação de uma máscara à imagem. Neste caso, duas imagens são tomadas como argumentos: uma imagem a ser mascarada, ou seja, cujas regiões devem ser selecionadas com a aplicação da máscara, e outra imagem que será usada como máscara, com pixels com valor 0 nas áreas a serem removidas e no valor máximo nas áreas a serem mantidas. Cada pixel da primeira imagem é avaliado de forma independente contra seu par na máscara e caso ambos tenham valor diferente de zero, o valor do pixel na imagem final será igual ao valor do pixel na imagem de entrada. Caso algum dos pixels tenha valor igual a zero, o valor do pixel na imagem final será igual a zero, com ilustrado na Figura 16.

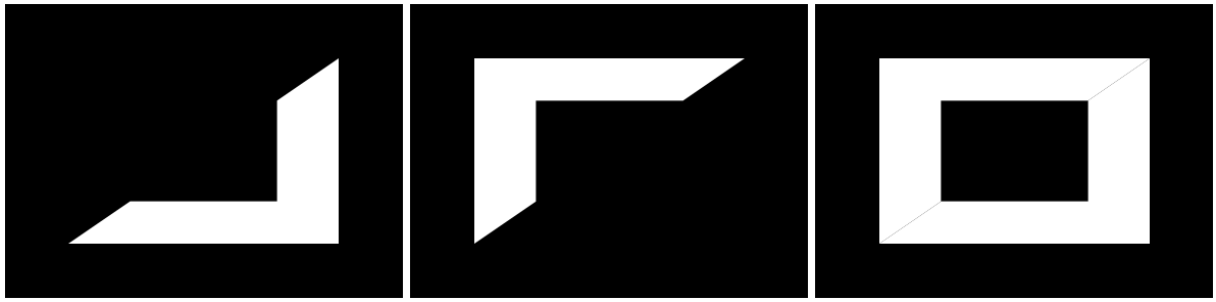
Figura 16 – Exemplo de aplicação de máscara com a operação binária E



Fonte: Adaptado de PXHERE (2017)

A operação OU, por outro lado, utilizada com duas imagens binarizadas, tem a função de somá-las. Duas imagens são utilizadas como argumentos de entrada para a função e assim como na operação E, os pixels das duas imagens são avaliados individualmente contra seus pares. Os pixels da imagem de saída serão iguais a um se o pixel correspondente em qualquer uma das imagens for igual a um, e zero apenas se o pixel correspondente em ambas as imagens for igual a zero. Na Figura 17 pode ser vista a aplicação da operação OU utilizando duas imagens binárias como entrada para a função (OPENCV, 2021).

Figura 17 – Exemplo de aplicação da operação binária OU



Fonte: Elaborado pelo autor

A partir dos conceitos apresentados neste capítulo, o capítulo seguinte aborda as várias etapas necessárias ao desenvolvimento do projeto.

3 MATERIAIS E MÉTODOS

Atualmente existem soluções comerciais para inspeção de adesivos de fabricantes tradicionais que desenvolvem há anos sistemas de inspeção por computador. Estas soluções utilizam, entretanto, sistemas dedicados e proprietários, de custo razoavelmente elevado. A proposta desta pesquisa é, portanto, desenvolver uma solução alternativa para a inspeção de adesivo aplicado no para-brisa, baseada em hardware de aquisição e processamento de baixo custo, e a partir do uso de linguagem de programação e bibliotecas de código aberto, desenvolver algoritmos que possam processar as imagens e extrair as características relevantes do cordão de adesivo e avaliar a efetividade desta solução.

Os testes foram realizados dentro das instalações da empresa *Jaguar Land Rover*, localizada no sul do estado do Rio de Janeiro, em condições reais de produção.

O sistema de aplicação de adesivo existente é composto por controlador de aplicação SYS6000, controlador de bomba PCU5000, dosador ADE-6000 e eixo posicionador do bico 1K, fabricados pela SCA Schucker, conectados através de interface *Ethernet/IP* ao controlador IRC5 de um braço robótico IRB6640 fabricados pela ABB.

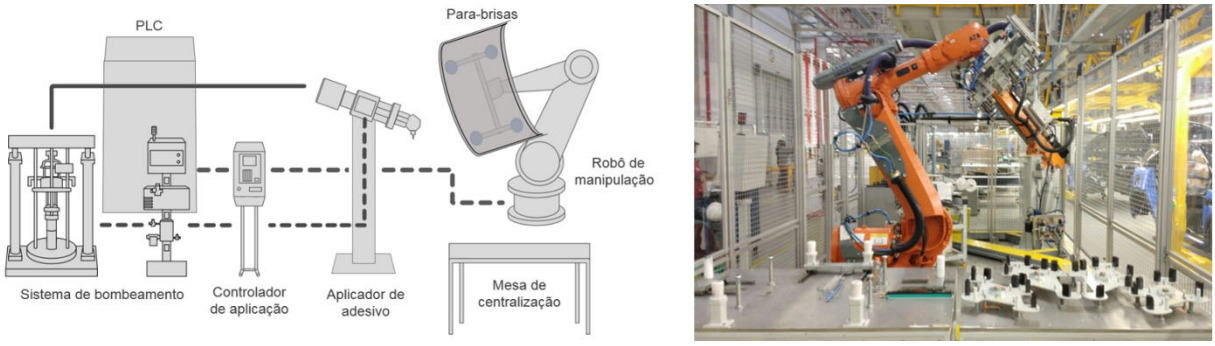
A célula de aplicação de adesivos é controlada por um CLP da família *ControlLogix*, fabricado pela *Allen Bradley*, que também está conectado ao controlador do braço robótico por interface *Ethernet/IP*, de forma que possam trocar sinais como presença de peça na mesa de centralização, estado dos dispositivos de segurança, entre outros.

O CLP possui entradas e saídas digitais discretas sobressalentes, que foram utilizadas no projeto. Na Figura 18 é apresentada uma visão geral dos sistemas que compõem a célula de aplicação de adesivos e uma fotografia da célula a partir da mesa de carregamento de para-brisas.

Juntamente com o time de engenharia de manufatura, foi definido que a característica a ser inspecionada seria a continuidade do cordão de adesivo, pois embora a frequência com que ocorrem defeitos deste tipo não seja alta, os impactos à produção e ao cliente final são muito significativos, como explanado na Introdução e na seção 2.1.3.

O para-brisas analisado foi o do veículo modelo *Discovery Sport*, que possui 1.564 mm de largura e 832mm de altura.

Figura 18 – Vista geral da célula de aplicação de adesivo



Fonte: Elaborado pelo autor

Outra solicitação da engenharia de manufatura foi que as imagens adquiridas deveriam ser gravadas e armazenadas no dispositivo, para posterior consulta em caso de dúvida sobre a capacidade de detecção do algoritmo, ou para avaliar outros defeitos que não os de continuidade. Neste aspecto, foi acordado que embora a memória do dispositivo permita o armazenamento de milhares de imagens, o que corresponderia a mais de um ano de captura de imagens, dada a produção anual da planta em questão, deveria ser previsto um meio fácil para acesso à memória do dispositivo para permitir que as imagens fossem copiadas para posterior armazenamento em um serviço de armazenamento descentralizado.

Para a proteger o cliente final, e evitar que para-brisas com falhas no cordão de adesivo fossem montados nos veículos, os limites de pressão para a célula de aplicação onde o projeto foi desenvolvido estavam bem justos, e com isso níveis elevados de parada de produção estavam sendo observadas por falhas no sistema de aplicação por pressão fora dos limites, rejeitando 24% das peças.

Desta forma, os requisitos acordados com a engenharia de manufatura para determinar a aprovação do sistema foram:

1. Todas as falhas de continuidade do cordão devem ser detectadas (0% de falsos negativos);
2. O nível de rejeição de peças boas (falsos positivos) deve ser menor que o observado no momento da análise (24%);
3. As imagens capturadas devem ser armazenadas em cartão de memória para consulta futura;
4. O sistema deve possibilitar aos técnicos de manutenção acessar a memória do dispositivo para periodicamente liberar espaço para novas imagens, e copiar as imagens para um armazenamento em nuvem.

Com os requisitos estabelecidos, o desenvolvimento do projeto foi dividido em duas etapas: o desenvolvimento do algoritmo para tratamento imagens e identificação da continuidade do cordão, abordado a seguir na seção 3.1; e o desenvolvimento de um protótipo para captação das imagens com interface com o CLP da linha, tratado na seção 3.2.

3.1 DESENVOLVIMENTO DO ALGORITMO PARA IDENTIFICAÇÃO DA CONTINUIDADE DO CORDÃO

Para o desenvolvimento da primeira etapa, a análise do problema do ponto de vista de visão computacional, é necessário avaliar quais características da imagem podem ser utilizadas para a obtenção das informações desejadas. Para isso, fotografias do para-brisas foram obtidas após a aplicação do adesivo, com o uso de uma câmera Sony modelo DSC-W210, na configuração automática, com resolução espacial de 8MP e sem iluminação auxiliar (*flash*). A escolha deste modelo de câmera em detrimento a câmeras de telefones celulares é que por ser um modelo antigo, esta câmera introduz ruídos na imagem que são típicos de sensores de baixo custo, possibilitando ainda em um estágio inicial a avaliação do impacto destes ruídos no sistema proposto.

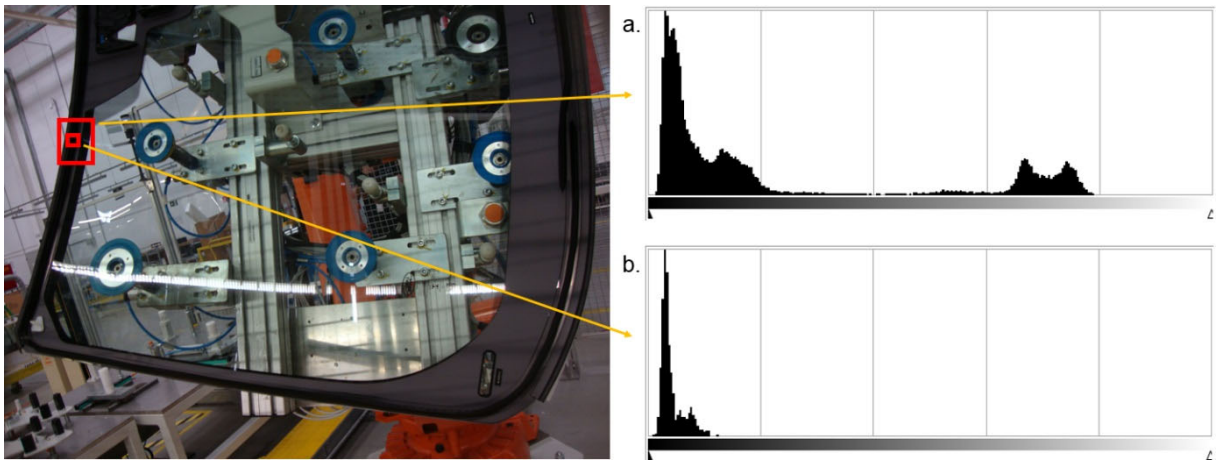
A partir das imagens obtidas, cor, intensidade luminosa e forma do cordão de adesivo foram avaliadas nas imagens obtidas, buscando identificar características que melhor descrevem o cordão de forma a distingui-lo do para-brisas sobre o qual ele é aplicado.

O *software* GIMP versão 2.8 foi utilizado para avaliar estas características, por ser gratuito e disponibilizar ferramentas como histograma, separação por canais de cor e ferramentas para binarização de imagens, de forma fácil e intuitiva.

A partir das primeiras análises foi observado que o cordão de adesivo é o objeto com menor intensidade luminosa da região onde é aplicado no para-brisas, embora existam outros objetos na imagem com intensidades próximas. Na Figura 19 é possível ver duas regiões analisadas: na primeira, sinalizada com um retângulo vermelho, é possível observar através do histograma a) dois picos de intensidade, um na intensidade 7, e outro na intensidade 11; na segunda região, posicionada diretamente sobre o cordão de adesivo, representada por um quadrado vermelho dentro da primeira região, pode ser observado no histograma b) apenas um pico na

intensidade 7. Esta característica foi observada em distintas regiões ao longo da imagem do cordão de adesivo, demonstrando ser útil para a distinção entre o cordão de adesivo e o para-brisas ao fundo.

Figura 19 – Histogramas a) da região onde se encontra o cordão de adesivo e b) apenas do cordão de adesivo



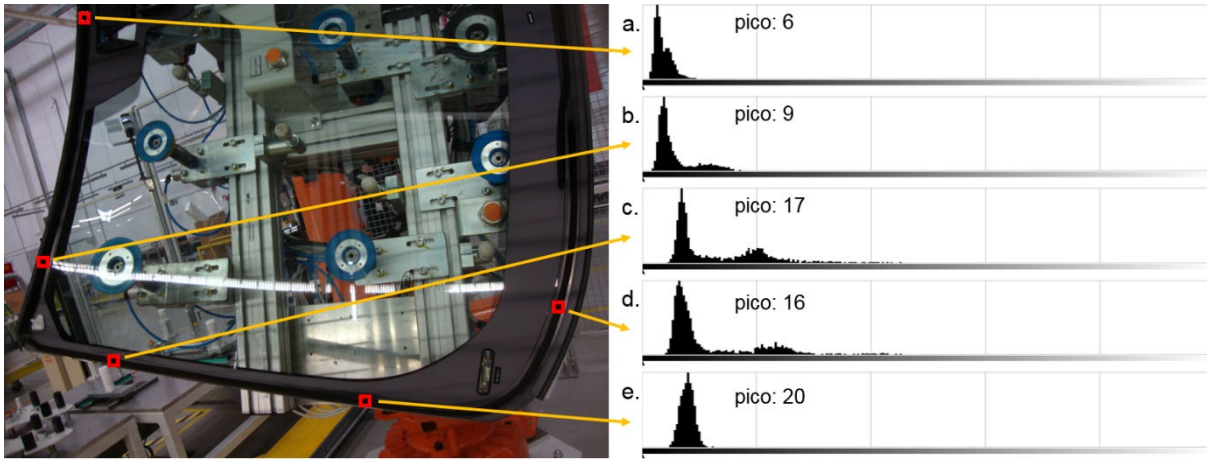
Fonte: Elaborado pelo autor

Um aspecto importante a ser observado é que o uso de picos de intensidades na imagem para detecção do cordão não depende do uso de cores. Desta forma, a imagem poderia ser convertida para tons de cinza, reduzindo assim o uso de memória e o tempo de processamento da imagem, por utilizar apenas um canal de intensidade, ao invés dos três necessários à imagem colorida.

Ao comparar as diferentes regiões de aplicação de adesivo, foi constatado que devido à diferença de iluminação entre estas regiões, os picos de intensidade luminosa que representam o cordão de adesivo variam entre 6 e 20, como pode ser visto na Figura 20, o que tornaria inviável a aplicação de um nível único para a binarização de todas as regiões da peça. Uma abordagem similar à proposta por Chen e Lin (2009) foi utilizada, separando a imagem em pequenas regiões de interesse (ROI), que poderiam ser analisadas independentemente. O resultado da análise de cada uma destas regiões seria então sobreposto para compor uma única imagem, contendo apenas o cordão de adesivo.

Entretanto, as regiões da imagem que não pertencem ao para-brisas e ao cordão de adesivo não são interessantes para a análise, e algumas delas apresentam regiões escuras que poderiam causar a detecção de objetos que não fazem parte do cordão de adesivo. Dado que o posicionamento do robô em frente à câmera teria pouca variação, devido à natureza da operação de aplicação de adesivo, foi utilizada

Figura 20 – Histogramas de diferentes pontos do cordão de adesivo

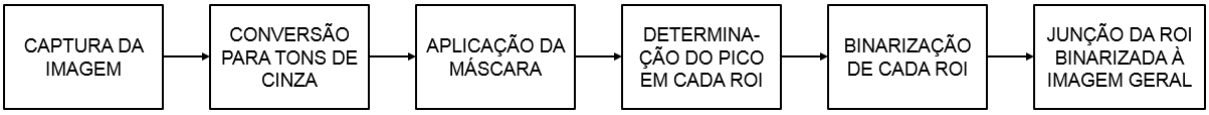


Fonte: Elaborado pelo autor

uma máscara, como exposto na pesquisa de Pinto e Bianchi (2013), para segregar do restante da imagem somente as áreas nas quais é provável a presença do cordão de adesivo. Esta máscara, desenhada pelo usuário no momento do cadastro de um novo padrão de cordão de adesivo, seria então utilizada para determinar quais regiões da imagem seriam analisadas, e quais seriam descartadas.

A partir desta sistemática básica, o algoritmo inicial foi criado e implementado na linguagem *Python* 3.6, utilizando a biblioteca *OpenCV* para as operações de manipulação de imagens, como conversão para tons de cinza, cálculo de histograma e binarização através das funções *cvtColor*, *calcHist* e *threshold* e a biblioteca Numpy para as operações com matrizes, como aplicação da máscara (operação E) e junção de cada ROI binarizada à imagem geral (operação OU), através das funções *bitwise_and* e *bitwise_or*. Na Figura 21 é apresentado o fluxograma do algoritmo inicialmente implementado.

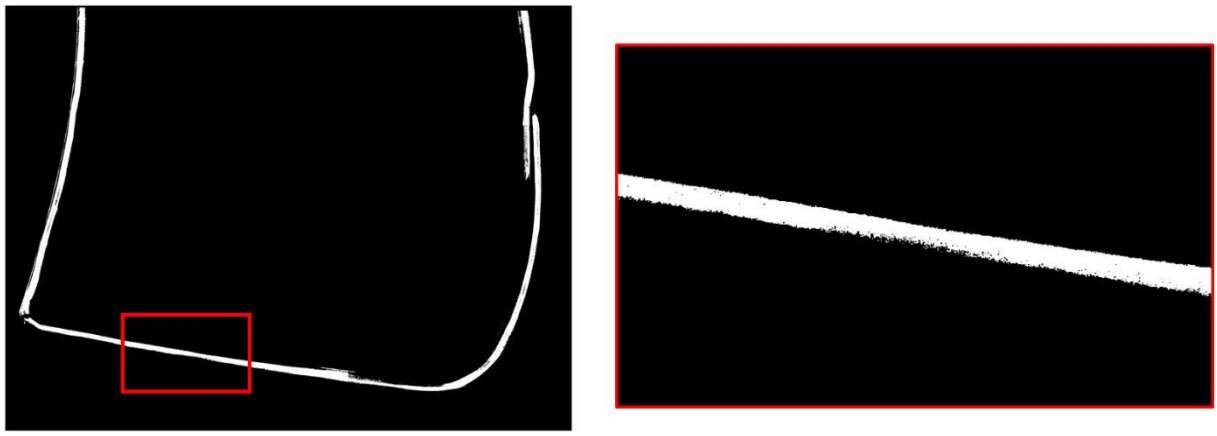
Figura 21 – Fluxograma do algoritmo inicial utilizado nos primeiros testes



Fonte: Elaborado pelo autor

Apesar de os primeiros resultados apresentarem excelente capacidade de evidenciar o cordão de adesivo sobre o para-brisas, foram observadas regiões com irregularidades na detecção, com a presença de pequenos pontos brancos nas extremidades do cordão de adesivo, e pontos pretos em seu interior, como pode ser visto no detalhe na Figura 22.

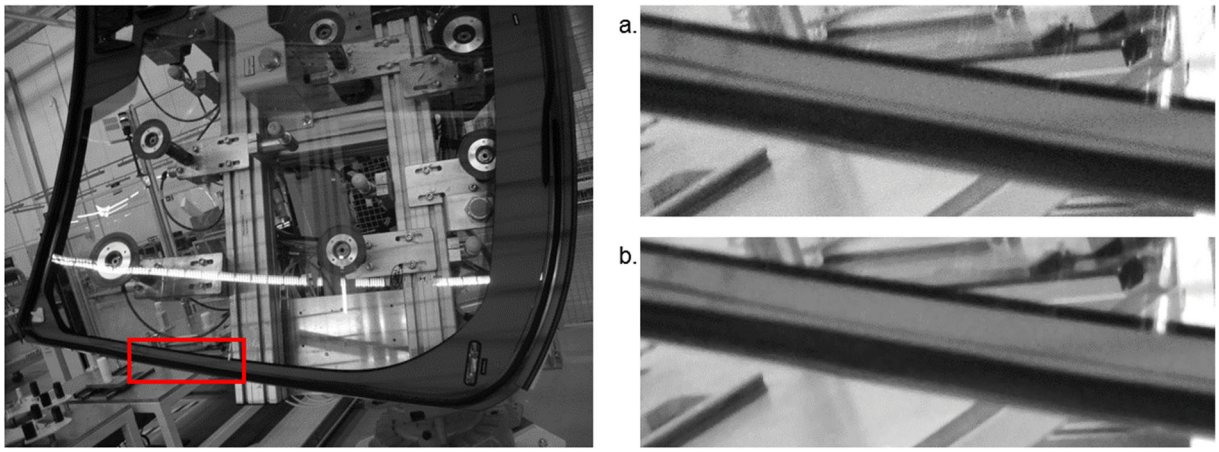
Figura 22 – Primeiros resultados do algoritmo de segmentação



Fonte: Elaborado pelo autor

Ao investigar a imagem de origem, foi observado que apresentava uma considerável granularidade como pode ser visto na Figura 23a, ruído proveniente do sensor ao captar a imagem com condição de pouca iluminação. Esta granularidade pode ser caracterizada como ruído gaussiano, em que a distribuição de frequência das intensidades do ruído segue a distribuição normal. Para redução do efeito da granularidade sobre o resultado da segmentação, foi empregado um filtro de mediana através da função *medianBlur* da biblioteca *OpenCV*, por apresentar bom balanço entre redução de ruídos e preservação de bordas (BARELLI, 2018, p. 107,119-120), como pode ser observado na Figura 23b.

Figura 23 – a) Granularidade apresentada na imagem original. b) Mesma região após a aplicação de filtro de mediana

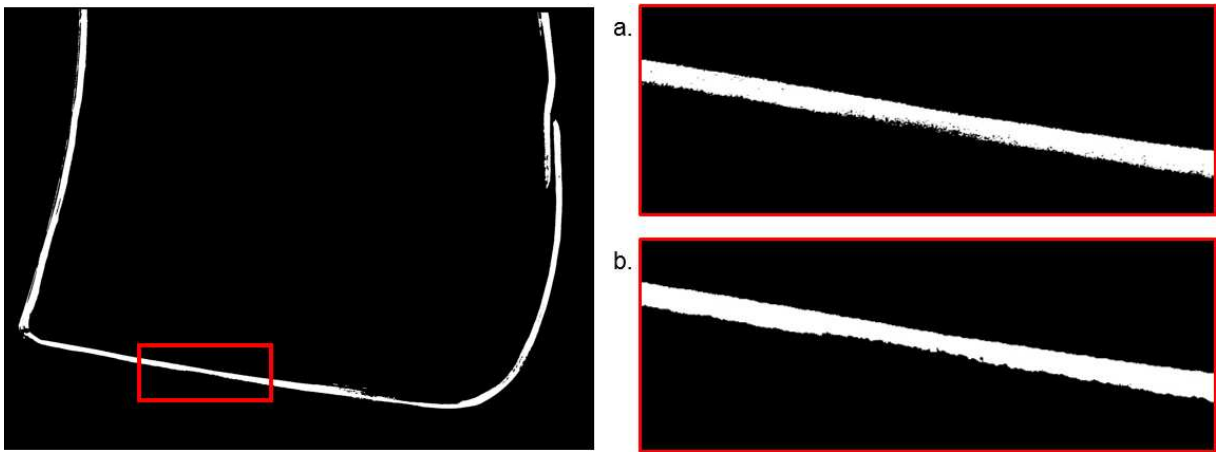


Fonte: Elaborado pelo autor

Após a implementação do filtro de mediana na etapa anterior à aplicação da máscara, foi observada uma considerável melhora na segmentação do cordão de adesivo e redução tanto dos pontos pretos no interior do cordão segmentado, quanto

de pontos brancos fora da área do cordão segmentado, que podem ser observadas na Figura 24.

Figura 24 – Segmentação do cordão a) antes de aplicação do filtro de mediana e b) após a aplicação



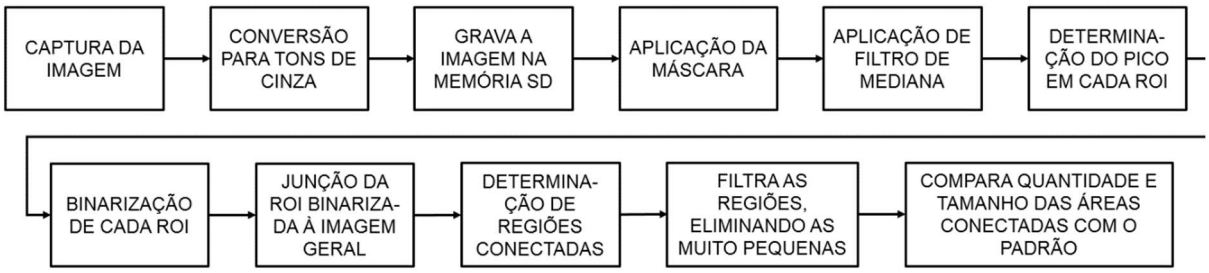
Fonte: Elaborado pelo autor.

A partir da imagem segmentada, a próxima etapa seria a localização de descontinuidades do cordão de adesivo. A abordagem utilizada para o problema é a de encontrar as regiões do cordão de adesivo que sejam conectadas entre si, de forma que a partir da contagem de regiões conectadas, é possível determinar se há descontinuidades no cordão de adesivo, uma vez que caso existam descontinuidades a quantidade de regiões conectadas seria alterada. Foi então utilizada a função *findContours* da biblioteca *OpenCV*, que tem justamente a função de encontrar contornos de regiões conectadas entre si. Entretanto, inúmeras microrregiões foram encontradas pela função, algumas com apenas poucos pixels.

Através da função *contourArea* também da biblioteca *OpenCV*, a área de cada uma destas regiões, em quantidade de pixels, é determinada, possibilitando a escolha de regiões que atendam critérios mínimos de tamanho. Desta forma, foi implementado no algoritmo um filtro para eliminar as regiões com menos de 4 pontos, e regiões cujas áreas fossem menores que 800 pixels, o que para a imagem em questão corresponde a um cordão de adesivo de aproximadamente 10 x 10mm. Na Figura 25 pode ser visto o fluxo do algoritmo para avaliação das imagens, após a implementação das mudanças.

Após estas modificações, o algoritmo de detecção de contornos identificou corretamente os cordões de adesivo, exceto por uma quina do cordão que na imagem obtida refletia a luz de uma luminária próxima, e assim tinha uma intensidade luminosa

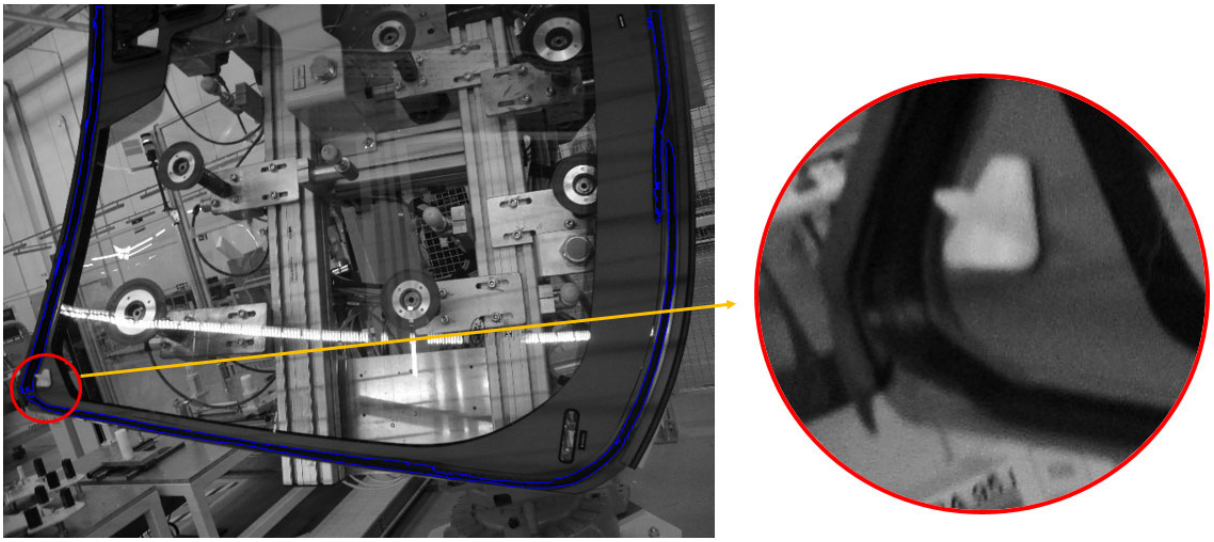
Figura 25 – Fluxograma do algoritmo final implementado para avaliação das imagens



Fonte: Elaborado pelo autor

muito acima das áreas vizinhas, sendo identificada como uma descontinuidade, como pode ser visto na Figura 26. Por ser um problema relacionado ao posicionamento da câmera em relação ao para-brisas fotografado, e dos reflexos ocasionados por este posicionamento, nenhuma modificação no algoritmo foi implementada imediatamente para tratar esta falha de identificação na continuidade do cordão. Ao invés disso, na etapa de posicionamento do robô para a captação de imagem, cuidados teriam que ser tomados para evitar este modo de falha.

Figura 26 – Áreas detectadas realçadas em azul e detalhe da área identificada como descontinuidade

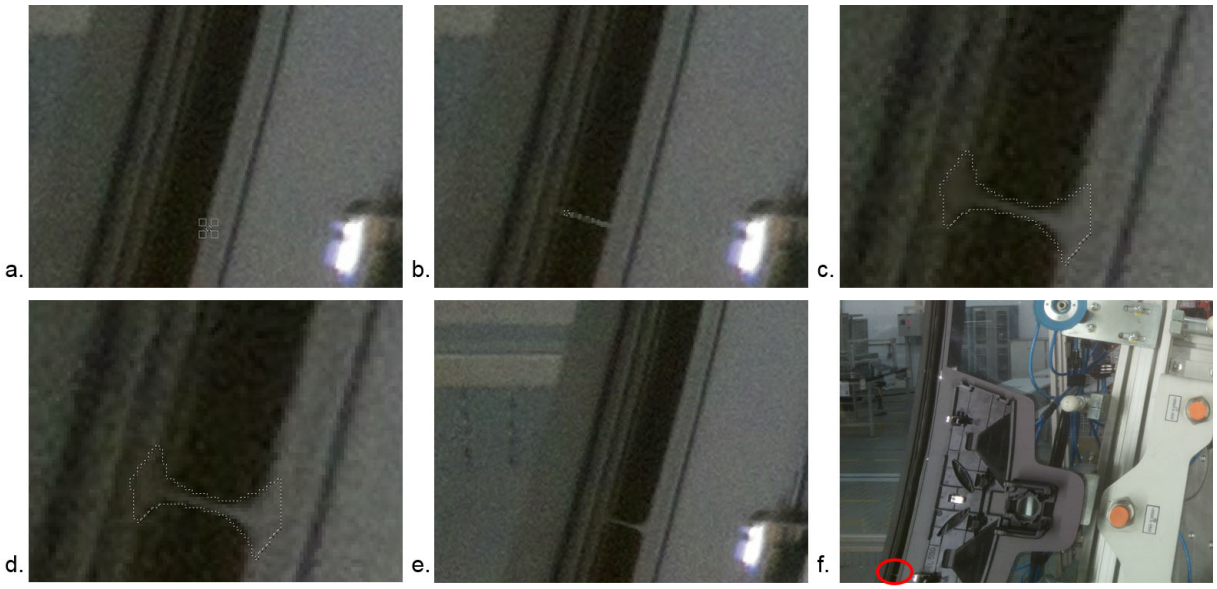


Fonte: Elaborado pelo autor

Para avaliar a efetividade do algoritmo em identificar descontinuidades nas regiões onde foi identificado o cordão de adesivo, a imagem utilizada inicialmente para teste de detecção do cordão de adesivo foi manipulada digitalmente com o software GIMP para apresentar descontinuidades. O procedimento, como pode ser visto na Figura 27, foi realizado a) copiando regiões vizinhas ao cordão de adesivo, em formato circular de 3 pixels de diâmetro, correspondendo a 1mm para o para-brisa considerado, b) para dentro da área do cordão de adesivo. Em seguida, c) uma

ferramenta de suavização foi aplicada para suavizar as bordas, de forma que a variação de cores seja gradual, como ocorre nas imagens reais. O último passo foi d) aplicar ruído para simular os ruídos característicos no processo de aquisição. Na Figura 27e pode ser visto o detalhe da região onde foi aplicada a modificação na imagem, que é dificilmente percebida quando se observa f) a imagem como um todo.

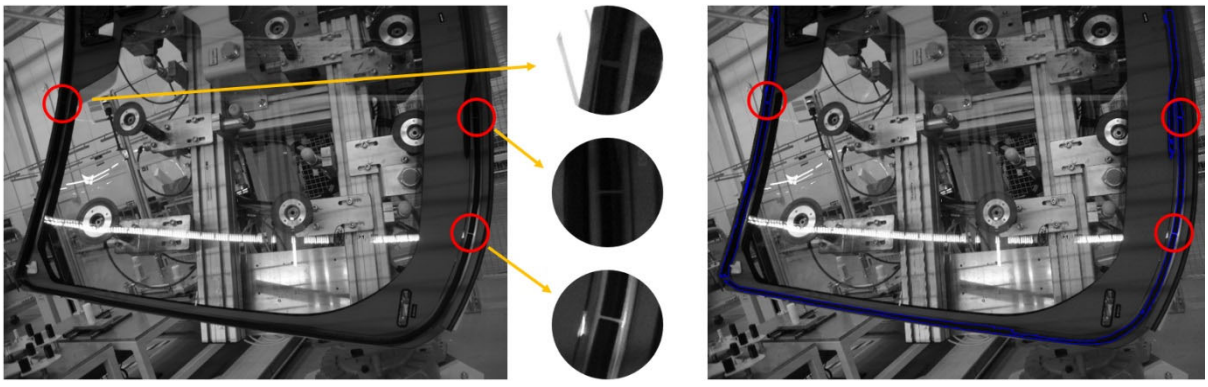
Figura 27 – Processo de manipulação da imagem para simulação de uma descontinuidade do cordão



Fonte: Elaborado pelo autor

A imagem modificada para incluir descontinuidades no cordão de adesivo foi então apresentada ao algoritmo, que conseguiu identificar com sucesso as três descontinuidades introduzidas sinteticamente no cordão de adesivo, como pode ser visto na Figura 28, demonstrando assim sua efetividade para a detecção de descontinuidades nas condições da imagem de referência.

Figura 28 – Imagem manipulada para simular descontinuidades na aplicação de adesivo e respectiva detecção de áreas pelo algoritmo



Fonte: Elaborado pelo autor.

O código fonte do programa desenvolvido pode ser encontrado no Apêndice C, e pode ser baixado do site https://github.com/rtudeschini/puinspection.

Com a conclusão do desenvolvimento da primeira etapa do projeto, foi iniciado o desenvolvimento do sistema para aquisição de imagens com interface com o controlador da linha de produção, descrito na seção a seguir.

3.2 DESENVOLVIMENTO DE UM PROTÓTIPO PARA CAPTAÇÃO DAS IMAGENS COM INTERFACE COM O CLP DA LINHA

O protótipo deste sistema é composto por um módulo de câmera *Raspberry Pi Camera* v2.1, equipado com um sensor Sony IMX219 CMOS de silício em configuração mosaico tipo *Bayer* com padrão BGGR, com resolução espacial de 8MP com imagens de até 3.280x2.464 pixels, operando no espectro visível (PAGNUTTI, RYAN, *et al.*, 2017). O módulo de câmera foi conectado à interface específica para câmera MIPI CSI de um computador de placa única (SBC) *Raspberry Pi*, modelo 3 B+, dotado de um processador 64-bit ARM Cortex-A53 de quatro núcleos de 1.4 GHz, e 1GB de memória RAM utilizando o sistema operacional *Raspberry Pi* OS versão 2020-05-27, baseado em *Linux*.

Para armazenamento do sistema operacional, softwares necessários ao sistema e gravação das imagens, foi utilizado um cartão de memória SD classe 4 de 32GB.

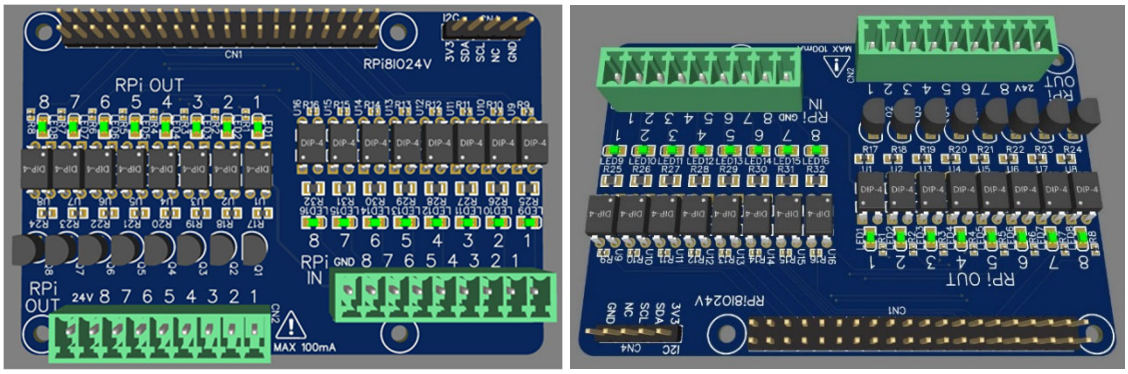
Entre outras interfaces de comunicação, o SBC utilizado possui uma interface de 40 pinos com entradas e saídas de uso geral (GPIO), 4 portas USB 2.0 e interface HDMI para conexão de monitor (RASPBERRY PI FOUNDATION, 2020).

Embora existam no mercado opções de SBC similares ao *Raspberry Pi*, ele foi selecionado por seu baixo custo e por ser facilmente encontrado a pronta entrega no mercado local, o que o torna vantajoso para manutenção do sistema caso venha a ser implementado em ambiente de produção. Para alimentação do SBC foi utilizada uma fonte CA-CC de 5V e 2A, conectada ao SBC por um cabo USB A-microB de 20 cm.

A interface entre o SBC e o controlador da célula de aplicação de adesivo foi feita através de sinais discretos, com o uso das GPIO do SBC. Como os sinais das GPIO operam com tensão de 3,3V, e as placas de entrada e saída digitais do controlador operam em 24V, com o uso da plataforma EasyEDA, foi desenvolvida uma placa de conversão de sinais com opto-acopladores PC817, de forma a isolar

eletricamente os sinais entre os dois sistemas e ainda assim permitir a troca de informações discretas. Na Figura 29 pode ser vista a representação 3D da placa montada com todos os componentes, gerada pela plataforma EasyEDA. A placa foi fabricada na China e montada pelo autor no Brasil. Os diagramas, desenhos da placa e link para o projeto podem ser encontrados no Apêndice D.

Figura 29 – Representação 3D da placa montada

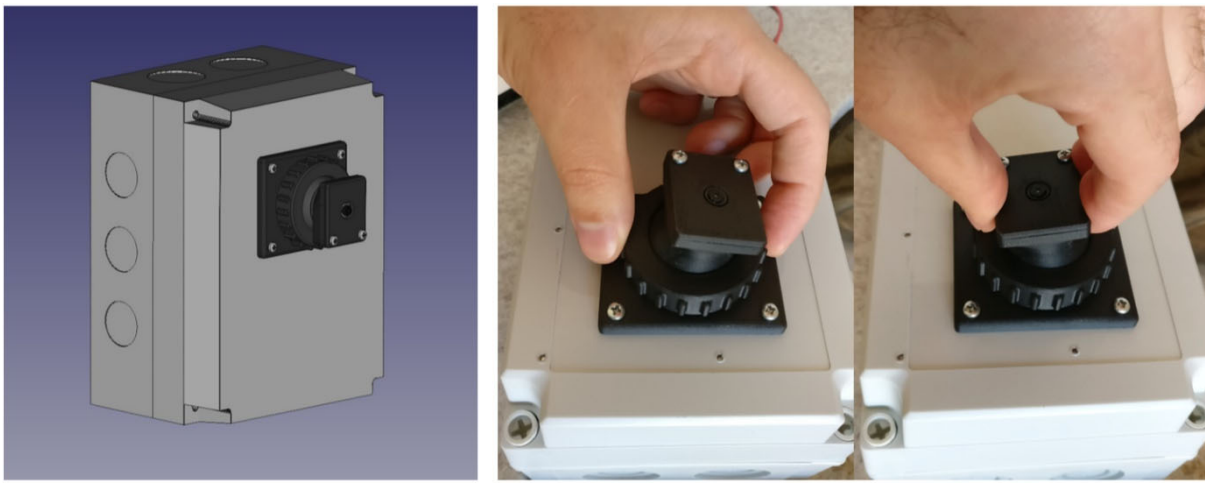


Fonte: Elaborado pelo autor

Todos os componentes do sistema foram montados em uma caixa plástica padrão industrial de 130mm de largura, 180mm de profundidade e 100mm de altura, de forma a protegê-los e às suas conexões de poeira e possíveis impactos.

Para acomodar a câmera e possibilitar um ajuste de seu ângulo após o posicionamento na célula de aplicação, um suporte foi desenhado no software FreeCAD versão 0.18 e impresso em impressora 3D em material Onyx, da empresa Markforged, como pode ser visto na Figura 30, e detalhado no Apêndice E. Para tornar

Figura 30 – Representação 3D do conjunto desenhado no software FreeCAD, e ajuste de ângulo da câmera possibilitado pelo suporte criado

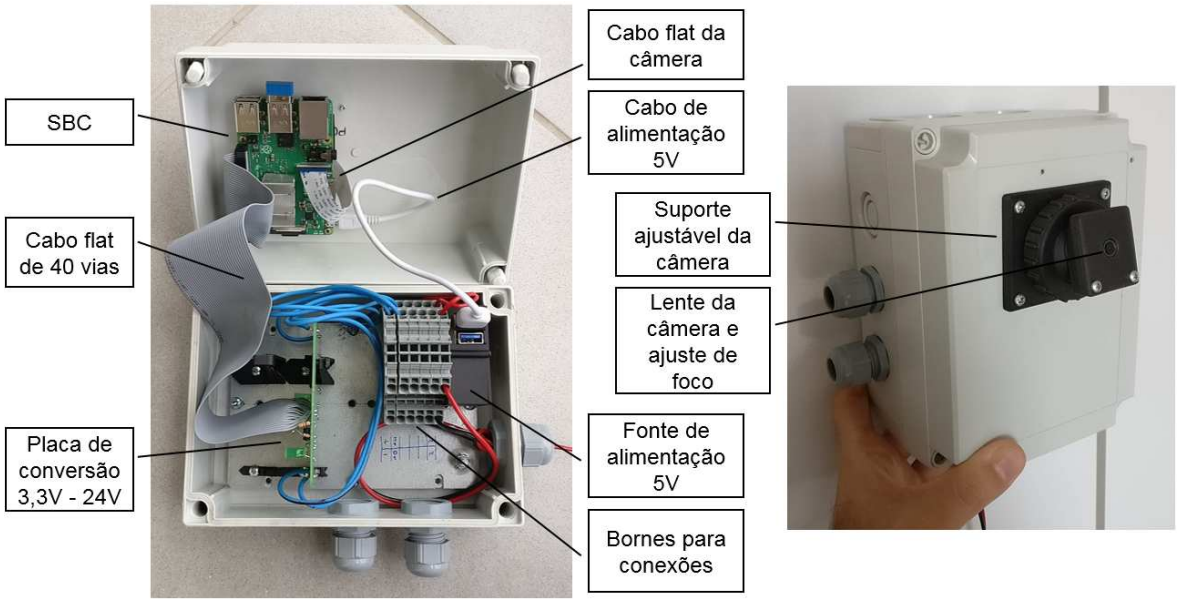


Fonte: Elaborado pelo autor

o conjunto mais compacto, a caixa da fonte de alimentação foi removida, os terminais de alimentação CA foram soldados a fios e uma nova caixa foi desenhada e impressa em impressora 3D, para possibilitar que a fonte fosse presa a um trilho DIN padrão IEC/EN 60715.

Para facilitar as conexões entre os componentes, bornes do tipo mola foram adicionados ao conjunto. Um trilho DIN 35x7,5mm foi fixado à placa metálica da caixa para que fossem montados a fonte e os bornes. Por fim, os componentes foram fixados e suas conexões estabelecidas. Na Figura 31 pode ser vista a distribuição dos componentes no interior da caixa, bem como o conjunto montado e caixa fechada.

Figura 31 – Componentes do protótipo do sistema de aquisição de imagens e conjunto montado



Fonte: Elaborado pelo autor.

Após a conclusão da montagem, um programa em linguagem *Python* foi criado para exibir a imagem captada pela câmera continuamente na tela, de forma a permitir o ajuste de foco e posicionamento da câmera depois que o protótipo fosse fixado na célula de aplicação de adesivo. Também foi criado um programa para obter amostras de imagens no ambiente real de utilização. Neste caso, como há a necessidade de sincronismo com o posicionamento do robô, para garantir que a imagem somente fosse capturada quando a peça estivesse posicionada em frente à câmera, foi utilizada a biblioteca *gpiozero* para aguardar que uma das entradas digitais GPIO fossem ativadas.

Após a captura e gravação da imagem no cartão de memória do SBC, um sinal de saída digital do GPIO é então ativado, para informar ao robô o fim da captura, para

que prossiga com a execução de seu programa e disponibilize a peça com o adesivo para a pega pelo operador. As imagens obtidas eram então salvas no cartão de memória, sendo os arquivos nomeados de acordo com a data e hora do sistema. O programa criado foi adicionado à lista de programas que são executados na inicialização do sistema, para que não fosse necessário aos técnicos o iniciar manualmente em caso de desligamento ou falta de energia.

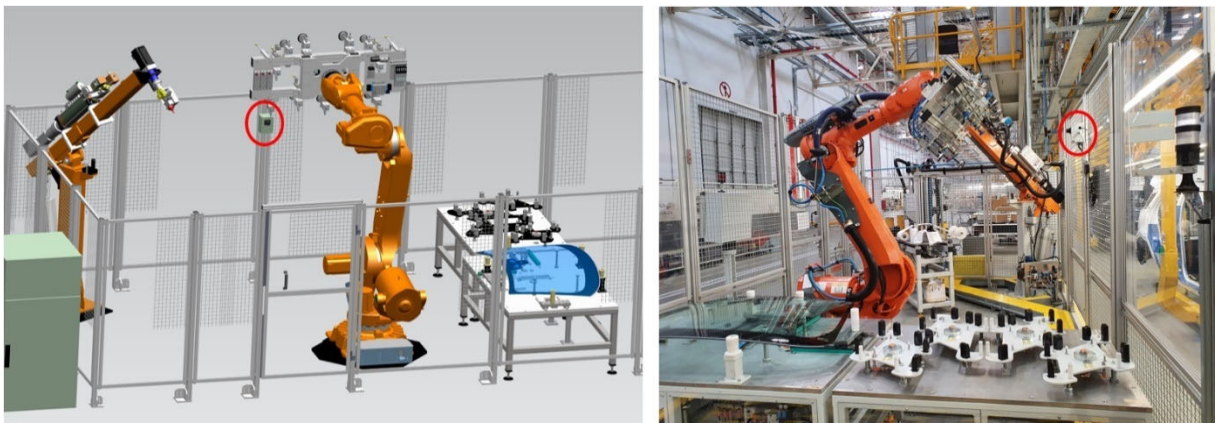
Testes foram realizados em bancada para garantir o funcionamento do programa de captura e a reposta das entradas e saídas. Um guia de instalação foi produzido para auxiliar os técnicos na instalação do protótipo e ajustes do posicionamento da câmera e ajuste de foco. Devido ao fato de o Raspberry Pi não possuir relógio de tempo real (RTC), o relógio do sistema retornava à data de lançamento do sistema operacional a cada vez que o SBC era reiniciado, fazendo com que alguns arquivos fossem sobrescritos, por possuírem nome relacionado à data e hora. Foi então realizada uma modificação no programa de forma a criar pastas com numeração sequencial, incrementada a cada vez que o programa era iniciado. Desta forma, uma nova pasta passou a ser criada sempre que o SBC era reiniciado, evitando assim que as imagens previamente gravadas na memória fossem sobrescritas.

O protótipo do sistema de captação de imagens desenvolvido foi então instalado em um pilar da grade de segurança da célula de aplicação de adesivo, em uma posição intermediária entre o pedestal de aplicação de adesivo e a mesa de centralização do para-brisas, de forma a minimizar o tempo de deslocamento adicionado para o posicionamento do para-brisas em frente à câmera. Na Figura 32 pode ser visto o posicionamento da caixa do protótipo instalada na célula de aplicação de adesivo.

Um monitor LG de 19” foi instalado na parte externa da célula e conectado ao SBC através da interface HDMI, para possibilitar aos técnicos visualizar as imagens obtidas, copiar as imagens gravadas para um dispositivo de armazenamento USB e realizar diagnóstico do sistema. Um adaptador de teclado e mouse sem fio também foi acoplado a uma das portas USB para possibilitar interação com o sistema, e à outra porta USB foi conectada uma extensão de USB de um metro, para possibilitar a conexão do drive USB para cópia dos arquivos de imagens. Para alimentação do protótipo e do monitor, duas tomadas padrão NBR 14136 foram instaladas, alimentadas por um disjuntor de 4A instalado no painel principal da célula de adesivo.

Cabos foram instalados conectando a placa de interface 3,3V-24V às entrada e saída digitais do CLP no painel principal.

Figura 32 – Protótipo no modelo 3D da célula de aplicação de adesivo, e instalado na célula real

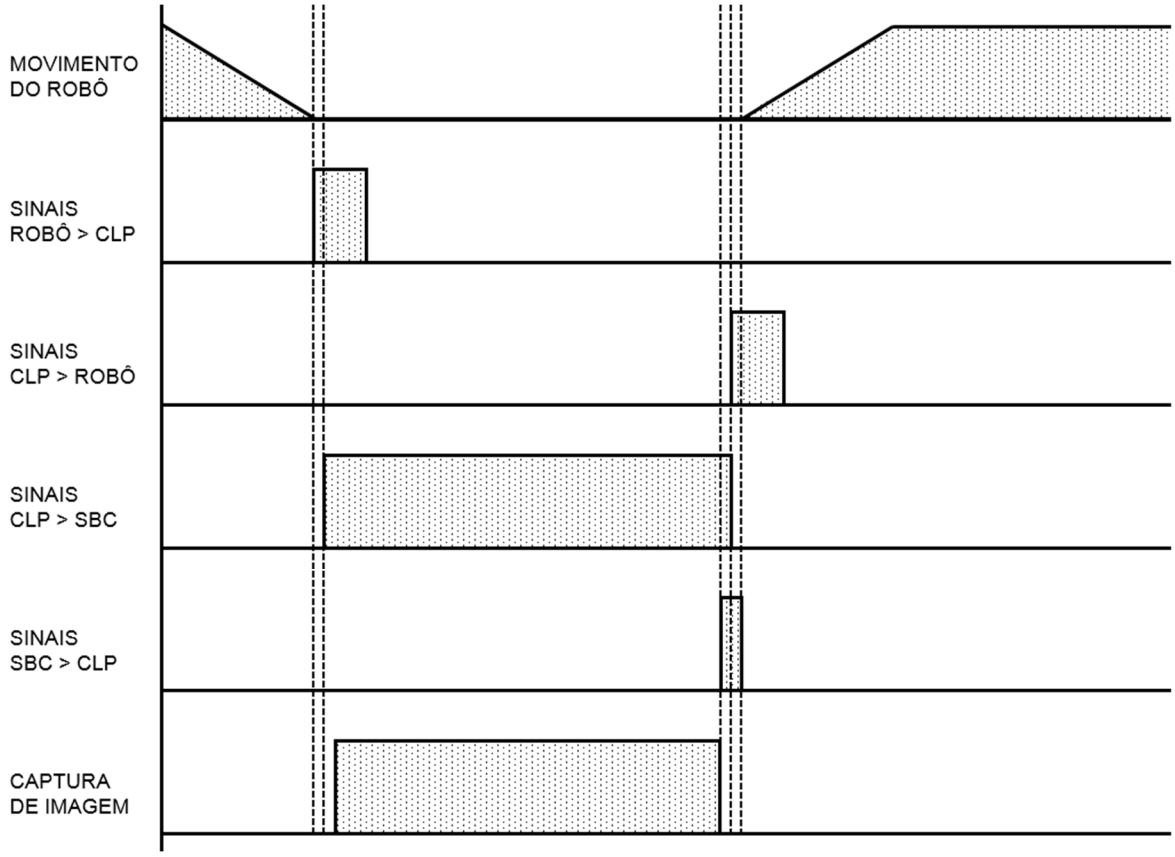


Fonte: Elaborado pelo autor.

Na célula de aplicação de adesivos estudada, o CLP tem o papel de sincronizar os dispositivos da célula. Desta forma, para garantir o sincronismo entre a aquisição de imagem pelo SBC e o robô, os programas do CLP da célula, desenvolvido em linguagem *ladder*, e do robô, desenvolvido em linguagem *Rapid*, de propriedade da empresa ABB, foram alterados para que após o término da aplicação de adesivo pelo robô, o robô se movimente até uma posição pré-definida em frente à câmera para a aquisição da imagem e envie então um sinal ao CLP informando a realização desta tarefa. O CLP então aciona um sinal digital conectado ao SBC, indicando ao SBC que o robô está posicionado. O SBC realiza a captura e salva a imagem capturada em sua memória e aciona então um sinal informando ao CLP que terminou a captura. O CLP desaciona o sinal enviado anteriormente, indicando que recebeu a informação do SBC, e envia informação ao robô para prosseguir seu programa, depositando a peça na mesa para que seja retirada da célula pelo operador.

Na Figura 33 pode ser visto o diagrama de tempo destas trocas de sinais entre robô, CLP e SBC. Caso o CLP não receba resposta do SBC em 10 segundos, assume que houve falha no SBC, desaciona o sinal enviado ao SBC e informa o robô para prosseguir com o programa. Todas as condições do programa, bem como a condição de falha por tempo excedido para resposta do SBC, são informadas ao operador da célula através de uma tela conectada ao CLP da célula, previamente existente.

Figura 33 – Diagrama de tempo das trocas de sinais entre robô, CLP e SBC para captura de imagem



Fonte: Elaborado pelo autor.

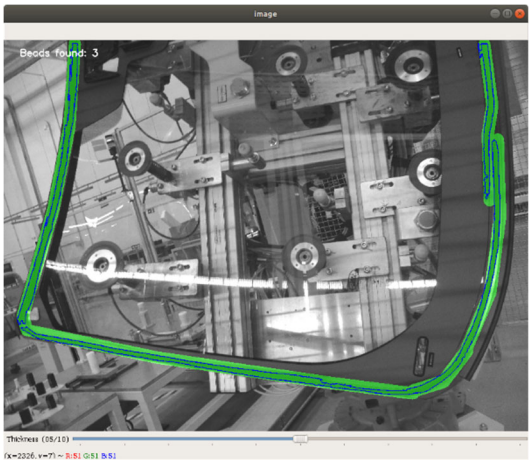
Após a instalação do protótipo e implementação das modificações nos programas do CLP e robô, o programa de aquisição de imagens ficou em execução por cerca de uma semana, coletando imagens do para-brisas após a aplicação do adesivo. Ao final deste período, um técnico da empresa efetuou a cópia das imagens para um dispositivo de armazenamento USB, e enviou ao autor para avaliação.

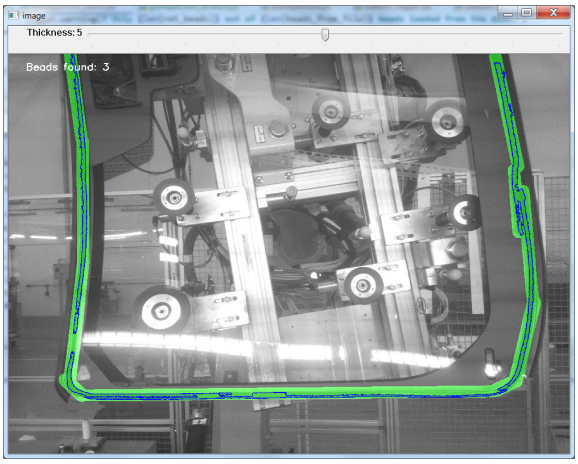
O capítulo a seguir descreve o resultado da avaliação das imagens pelo algoritmo, bem como as adaptações necessárias ao sistema para que os resultados esperados fossem atingidos.

4 RESULTADOS E DISCUSSÕES

O teste do algoritmo foi realizado com o uso das imagens obtidas em ambiente real de produção pelo protótipo instalado. Uma das imagens foi selecionada aleatoriamente para servir como padrão para seleção da posição e quantidade de áreas para o algoritmo. Como pode ser notado na Figura 34, a imagem capturada pelo protótipo apresenta um contraste inferior ao da imagem utilizada como referência para o desenvolvimento do algoritmo. Outro ponto observado foi que há um reflexo na imagem capturada pelo protótipo proveniente de uma luminária localizada atrás do protótipo.

Figura 34 – Comparação das telas de desenho das máscaras para a) imagem de referência e b) imagem capturada pelo protótipo

a. 

b. 

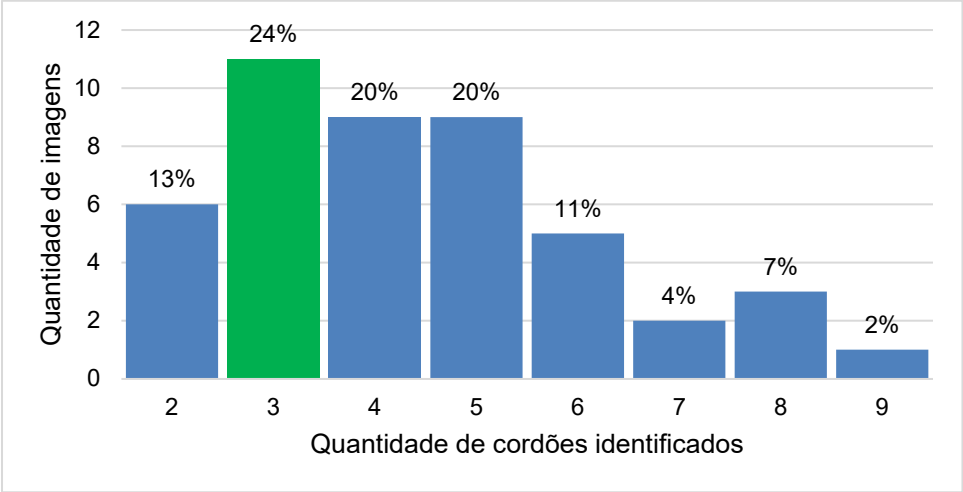
Fonte: Elaborado pelo autor

Este reflexo intenso provocou uma perda da capacidade de distinguir o adesivo nas regiões de interesse próximas ao reflexo, causando uma detecção incorreta de uma falha na continuidade do adesivo. Desta forma, ao desenhar a máscara sobre o cordão de adesivo, o algoritmo encontrou três cordões contínuos de adesivo, e não dois, como esperado. Para os testes seguintes, o valor de três cordões contínuos passou a ser utilizado como referência para aprovação pelo algoritmo.

Executando o algoritmo para verificar as demais imagens, que possuem características semelhantes de contraste e brilho devido ao posicionamento da peça em relação à câmera e reflexo na imagem, foi encontrado um índice baixo de aprovação. Das 46 imagens examinadas, em apenas 11, ou 24%, foram encontradas a mesma quantidade de áreas que na imagem padrão, e em nenhuma delas o tamanho das áreas ficou próximo do tamanho das áreas encontradas na imagem

padrão. Na Figura 35 pode ser vista a distribuição dos resultados obtidos, com a quantidade esperada de cordões (três) destacada em verde.

Figura 35 – Distribuição de resultados de áreas encontradas pelo algoritmo



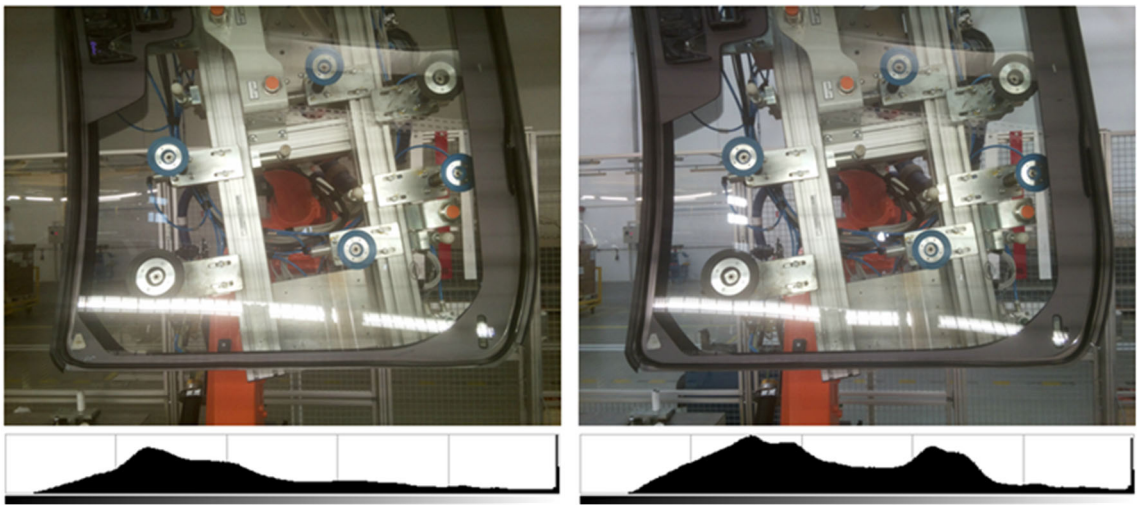
Fonte: Elaborado pelo autor

Por não ter sido inicialmente instalada nenhuma iluminação adicional para este projeto, a iluminação da peça nas imagens se devia exclusivamente à iluminação do galpão, proveniente de claraboias e luminárias no teto, instaladas a cerca de 10 metros de altura, e da iluminação de processo da linha de produção vizinha à célula de aplicação de adesivos. Como a iluminação natural proporcionada pelas claraboias varia ao longo do dia, era esperada alguma variação nos resultados devido à mudança de iluminação.

No entanto, ao analisar o histograma de imagens capturadas em diferentes momentos, foi observado que as mudanças nas condições de iluminação causaram uma expressiva mudança no resultado da captura, que pode ser notada tanto visualmente, pela mudança de contraste, quanto pelo histograma, na qual se pode notar uma melhor distribuição de intensidades na imagem mais iluminada, como pode ser visto na Figura 36. Correlacionando os resultados desta análise da iluminação com os resultados do algoritmo, foi identificado que na imagem com melhor iluminação o resultado da quantidade de áreas identificadas foi mais próximo do esperado, enquanto a imagem com resultado mais distante do esperado apresentava iluminação mais deficiente.

Para pesquisar mais a fundo esta correlação, foi incluído no algoritmo de análise o cálculo da média das intensidades da imagem, de forma a prover uma métrica da iluminação da imagem captada. Em seguida, os resultados foram

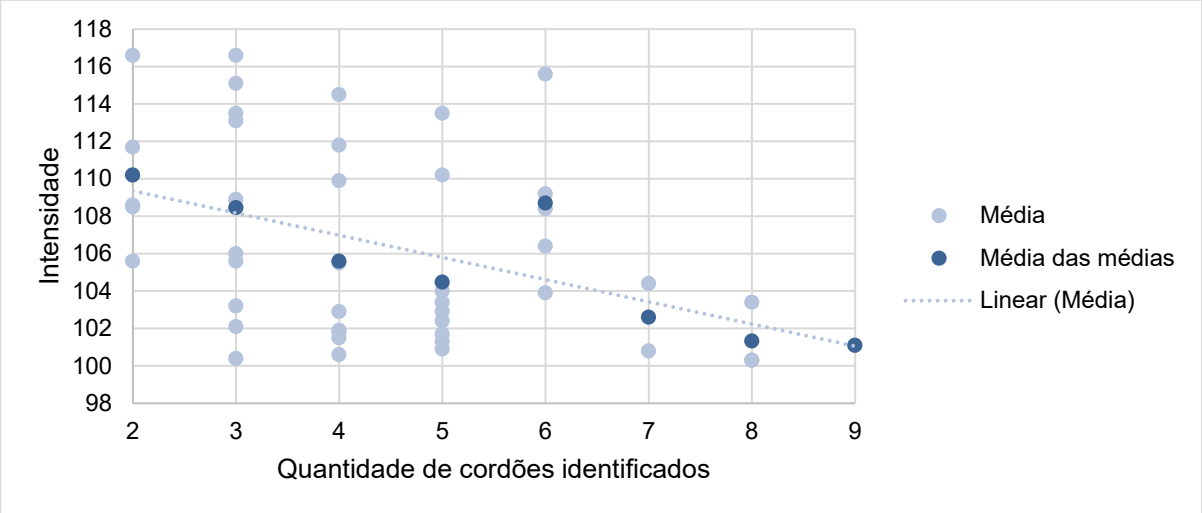
Figura 36 – Imagens capturadas em momentos diferentes, com considerável variação na iluminação



Fonte: Elaborado pelo autor

exportados para o programa *Microsoft Excel 365*, em que foram plotados no gráfico da Figura 37. De forma geral, é possível notar uma certa correlação entre as médias das intensidades e a quantidade de áreas identificadas, apesar de haver uma grande variação nas médias entre as imagens com a mesma quantidade de áreas identificadas.

Figura 37 – Relação entre intensidade nas imagens captadas e quantidades de áreas identificadas



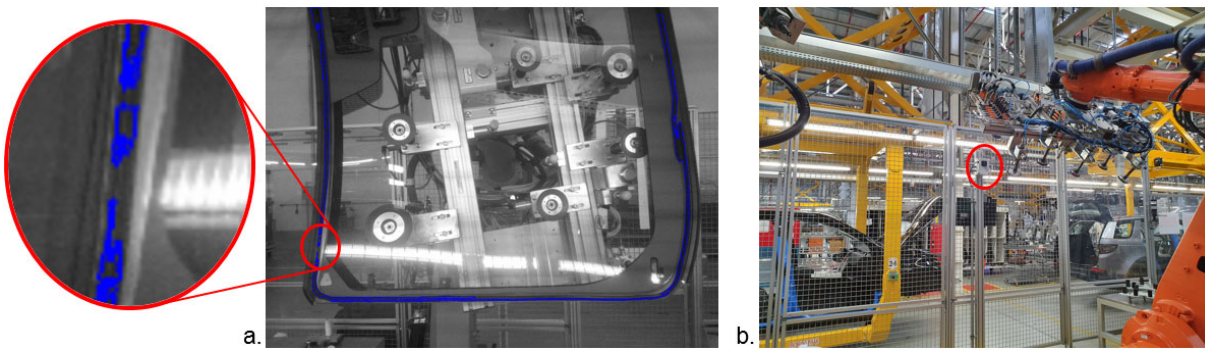
Fonte: Elaborado pelo autor

Foram também calculadas as médias das médias de intensidade das imagens com a mesma quantidade de áreas identificadas, a fim de simplificar a visualização no gráfico. Em seguida, foi calculado o coeficiente de correlação entre as médias das intensidades e a quantidade de áreas, e entre a média das médias de intensidade e a quantidade de áreas identificadas.

O coeficiente de correlação é uma medida estatística calculada pela relação entre a covariância de amostras de duas variáveis e seus desvios padrão, e indica o quanto a variação em uma variável está relacionada à variação em outra variável. Coeficientes próximos a 1 indicam uma forte correlação positiva, ou seja, a variação positiva em uma variável é acompanhada pela variação positiva na outra variável; coeficientes próximos a -1 indicam forte correlação negativa, ou seja, a variação positiva em uma variável é acompanhada por uma variação negativa na outra variável; e coeficientes próximo a zero indicam fraca correlação entre as variáveis (ANDERSON, SWEENEY e WILLIAMS, 2011, p. 119-121). O coeficiente de correlação entre as médias das intensidades e a quantidade de áreas calculado foi de -0,42, confirmando a análise gráfica de que há uma certa correlação entre as duas variáveis. O coeficiente calculado entre as médias das médias das intensidades e a quantidade de áreas identificadas foi de -0,86.

Outro ponto observado foi que havia nas imagens um forte reflexo causado por luminárias posicionadas atrás da câmera. Este reflexo fazia com que o algoritmo identificasse as áreas com reflexo, como descontinuidades, comprometendo os resultados. Esta situação pode ser observada na Figura 38a, na qual as áreas identificadas pelo algoritmo são demarcadas em azul, e é possível ver claramente que há descontinuidades nestas áreas. Este reflexo estava sendo causado pelas luminárias utilizadas na iluminação da linha de produção, posicionadas atrás da câmera e fora da célula de aplicação de adesivo, mencionadas anteriormente. Na Figura 38b pode ser vista a câmera, circulada em vermelho e as luminárias fonte dos reflexos.

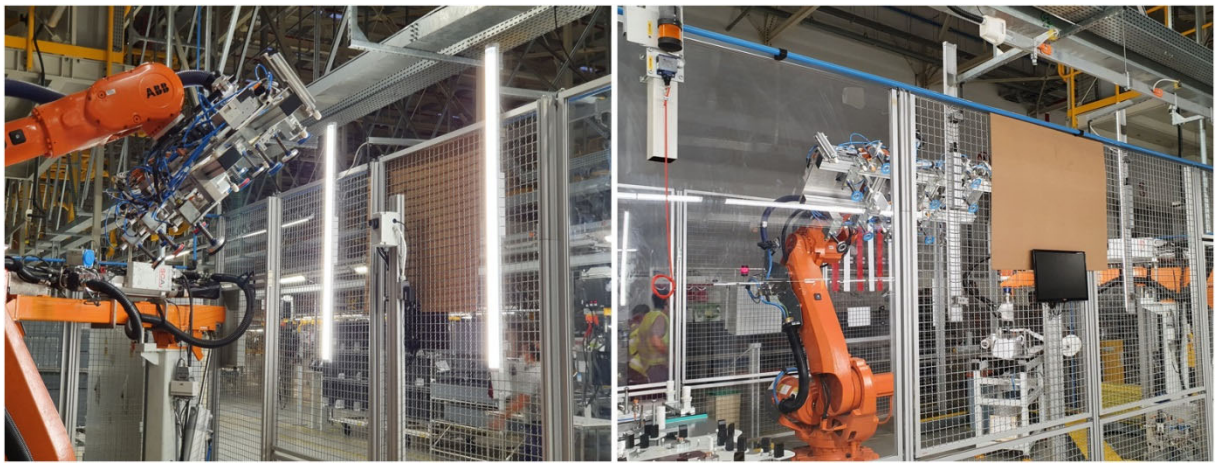
Figura 38 – a) Reflexo das luminárias identificado como descontinuidade do cordão e b) luminárias de processo posicionadas atrás da câmera



Fonte: Elaborado pelo autor

Com a indicação de que a iluminação exerce um papel importante na capacidade de detecção das áreas pelo algoritmo, o passo seguinte para melhoria dos resultados foi modificar a iluminação da cena de forma a reduzir o reflexo das luminárias posicionadas atrás da câmera, e reduzir também a variação da luminosidade em geral. Para isso, as luminárias posicionadas imediatamente atrás da câmera foram removidas, evitando assim o forte reflexo identificado anteriormente, e foi instalado um anteparo na grade atrás da câmera, de forma a bloquear a luz das luminárias localizadas no outro lado da linha que incidia sobre o para-brisas. O programa do robô foi ajustado para que o posicionamento do para-brisas em frente à câmera reduzisse os reflexos indesejados e foram também instaladas luminárias em posições que permitem iluminar o para-brisas de forma mais homogênea, sem que fossem causados reflexos indesejados, como pode ser visto na Figura 39.

Figura 39 – Luminárias e anteparo instalados para correção da iluminação



Fonte: Elaborado pelo autor

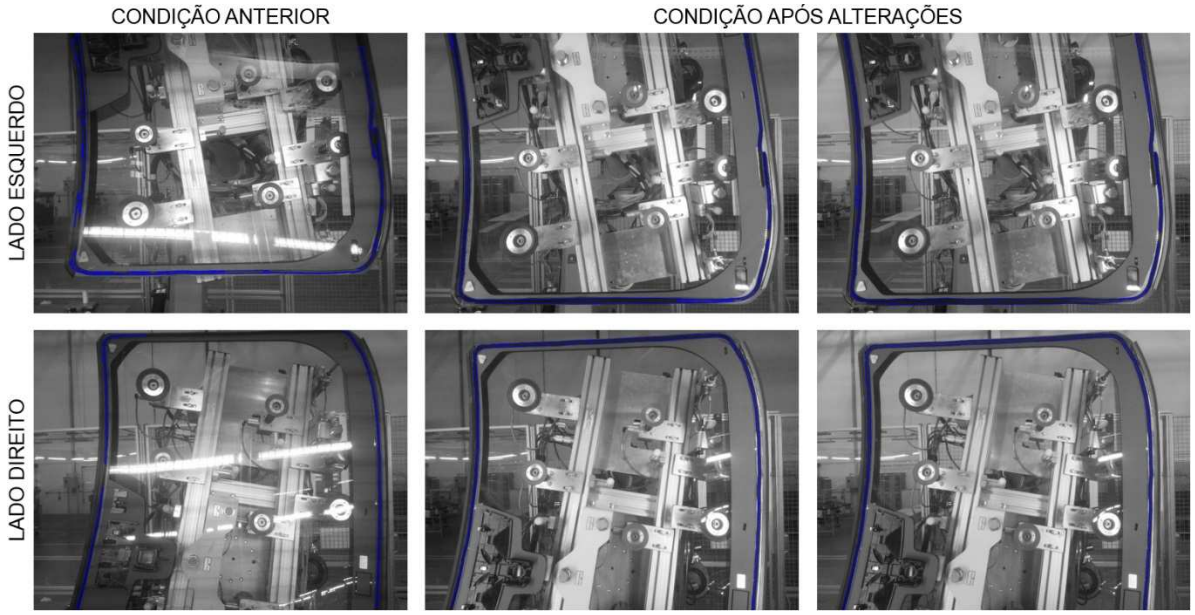
Após as modificações implementadas, a iluminação da peça durante a aquisição das imagens melhorou consideravelmente, sendo possível notar que mesmo em variadas condições de iluminação do fundo, devido à variação da luz ambiente pelas claraboias do teto, a iluminação da peça não foi prejudicada. As imagens do lado esquerdo da peça ainda apresentaram alguns reflexos indesejados no lado direito da imagem, devido ao posicionamento da peça em relação às luminárias instaladas. As imagens do lado direito, entretanto, apresentaram condições de iluminação muito boas e constantes, invariantes com a iluminação do ambiente ao fundo.

Na Figura 40 é apresentada uma comparação entre as áreas identificadas pelo algoritmo com as imagens antes e após as modificações serem implementadas. É

possível notar que os fortes reflexos presentes nas imagens tanto do lado direito quanto esquerdo não estão mais presentes, e que a detecção do cordão pelo algoritmo melhorou consideravelmente.

Como referência para o algoritmo, foi utilizada a imagem do lado direito da peça, que apresenta um cordão contínuo de adesivo, que foi corretamente detectado pelo algoritmo na etapa de desenho da máscara. Foram então captadas 77 imagens da peça em diferentes momentos do dia, abrangendo diversas condições de iluminação do ambiente.

Figura 40 – Comparação das imagens anotadas na condição anterior e após as modificações

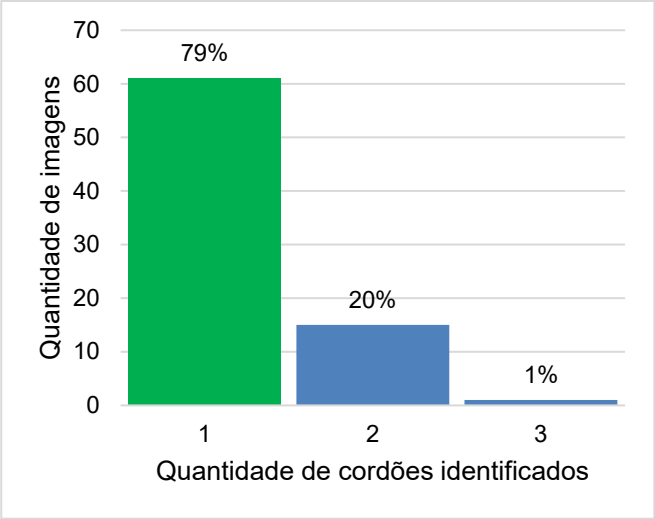


Fonte: Elaborado pelo autor

As imagens foram então processadas pelo algoritmo, para avaliar a capacidade de detecção de cordões de adesivo. Os resultados mostram uma grande melhora na capacidade de detecção da continuidade dos cordões, pois em 61 imagens, ou 79% dos casos analisados, o algoritmo conseguiu detectar corretamente a quantidade de cordões presente na imagem. Desta forma, o nível de falsos positivos, imagens boas que foram reprovadas, ficou em 21%, inferior ao nível de referência do sistema atual. Na Figura 41 pode ser vista a distribuição da quantidade de áreas detectada pelo algoritmo, sendo possível ainda notar que houve significativa redução na dispersão dos valores.

Outro ponto analisado foi a área, em pixels, de cada cordão de adesivo detectado. Para esta análise, foram considerados apenas os casos em que a quantidade de cordões de adesivo foi corretamente detectada pelo algoritmo. Este

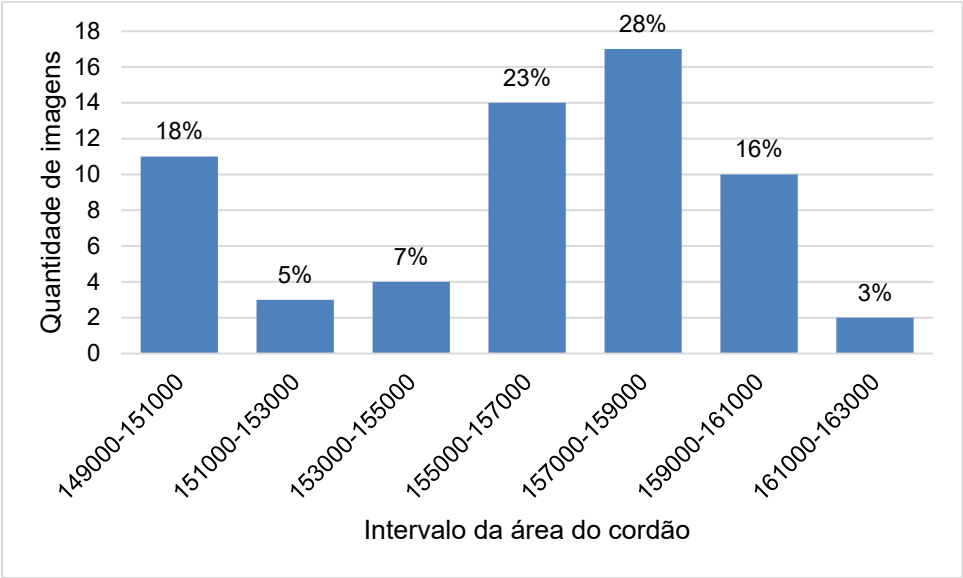
Figura 41 – Distribuição da quantidade de áreas encontradas pelo algoritmo após as modificações



Fonte: Elaborado pelo autor

parâmetro deve ser usado para determinar se o cordão detectado é válido, evitando assim possíveis casos em que houvesse ausência total do adesivo, o que poderia levar o algoritmo a considerar toda a área da máscara como um único cordão. Foi verificado que o menor valor para a área no conjunto verificado foi de 149.179 pixels e a maior área foi de 162.645 pixels. Na Figura 42 pode ser vista a distribuição das áreas dos cordões das 61 imagens consideradas.

Figura 42 – Distribuição das áreas dos cordões das imagens



Fonte: Elaborado pelo autor

Por fim, para avaliar a capacidade do algoritmo em detectar descontinuidades nos cordões de adesivo, uma a cada três das 61 imagens consideradas foi manipulada

digitalmente de forma a introduzir uma descontinuidade no cordão, utilizando o mesmo procedimento empregado na etapa de desenvolvimento do algoritmo, gerando 21 imagens. Apesar de não representar a condição ideal para a validação do algoritmo, esta manipulação das imagens para simulação dos defeitos foi necessária devido à falta de peças com defeito durante os testes, e à dificuldade de simular a descontinuidade do cordão, ou seja, causar propositalmente uma interrupção no cordão de adesivo, no sistema de aplicação real.

As 21 imagens manipuladas foram apresentadas ao algoritmo para avaliação e em 100% das imagens, o algoritmo foi capaz de identificar as descontinuidades introduzidas, representando 0% de falsos negativos.

Desta forma, os quatro requisitos acordados com a engenharia de manufatura foram considerados atingidos, como pode ser visto na Tabela 1:

Tabela 1 – Comparação entre os resultados obtidos e os requisitos acordados

Número	Requisito	Resultado
1	Todas as falhas de continuidade do cordão devem ser detectadas (0% de falsos negativos).	Todas as falhas de continuidade do cordão introduzidas foram detectadas (0% de falsos negativos).
2	O nível de rejeição de peças boas (falsos positivos) deve ser menor que o atual (24%).	O nível de rejeição de peças boas (falsos positivos) foi de 21%.
3	As imagens capturadas devem ser armazenadas em cartão de memória para consulta futura.	Função para armazenamento das imagens foi incluída no algoritmo. O cartão de memória instalado tem capacidade para cerca de 4.100 imagens.
4	O sistema deve possibilitar aos técnicos de manutenção acessar a memória do dispositivo para periodicamente liberar espaço para novas imagens, e copiar as imagens para um armazenamento em nuvem.	O Raspberry Pi possui portas USB 2.0 na qual foi instalado um adaptador para teclado e mouse sem fio. Também foi instalada uma extensão até um ponto de fácil acesso para a conexão de um dispositivo de armazenamento, para a cópia dos arquivos.

Fonte: Elaborado pelo autor

5 CONCLUSÕES E PESQUISAS FUTURAS

A inspeção automática do cordão de adesivo é um importante incremento aos sistemas de controle de aplicação de adesivos em vidros automotivos, pois reduz significativamente a possibilidade de um vidro com cordão de adesivo interrompido ser montado ao veículo, e em última instância, chegar ao cliente, causando grandes transtornos. O sistema desenvolvido nesta pesquisa para inspeção do cordão de adesivo consiste em um hardware de baixo custo e amplamente disponível no mercado, como o SBC *Raspberry Pi* e seu módulo de câmera, e placa de interface desenvolvida a partir de componentes eletrônicos discretos. O algoritmo para análise das imagens desenvolvido utiliza uma abordagem simples, baseada em binarização a partir de limiares locais, para segmentar o cordão de adesivo, e quantidade de áreas contínuas detectadas, para a verificação da continuidade do adesivo, que se mostrou bastante robusta nos testes realizados.

Uma taxa de detecção de 100% foi atingida nas imagens manipuladas digitalmente para incluírem o defeito de descontinuidade do cordão de aplicação, e a taxa de falsos positivos foi de 21%, inferior ao nível de falsos positivos do sistema existente, que utiliza controle de pressão do adesivo no bico de aplicação. Apesar do resultado, testes com um maior número de imagens precisam ser realizados, além de um acompanhamento a longo prazo, para garantir que na eventual ocorrência de uma falha no sistema de aplicação que cause uma descontinuidade no cordão, a solução desenvolvida consiga reconhecer corretamente a falha.

Um cartão de memória tipo SD possibilita ainda o armazenamento de aproximadamente 4.100 imagens de peças, o que é uma importante ferramenta para a análise de problemas, caso ocorram falhas na aplicação de adesivo que não forem identificadas por esta solução ou pelos sistemas de controle existentes.

As principais contribuições desta pesquisa foram a criação de uma plataforma de hardware de baixo custo para aquisição de imagens que pode ser integrada a uma linha de produção, o desenvolvimento de um algoritmo para detecção de descontinuidades em cordões de adesivos escuros, como o cordão de PU aplicado em vidros automotivos, o estabelecimento de métricas para a aprovação de novos sistemas de inspeção de adesivos que venham a ser instalados na indústria automotiva, bem como uma referência de performance para este tipo de sistema e uma revisão bibliográfica de adesivos para colagem de vidros automotivos, seus

sistemas de aplicação, sistemas de aquisição de imagens e técnicas utilizadas para extração de informações a partir delas.

Apesar dos bons resultados obtidos neste projeto, várias melhorias ainda podem ser implementadas, tornando a detecção ainda mais confiável e robusta e melhorando a integração com a linha de produção. Neste sentido, uma lista é apresentada a seguir com algumas sugestões para pesquisas futuras:

Iluminação e parâmetros de captura: adequações na iluminação da peça, como reposicionamento das luminárias, instalação de luminárias adicionais e modificação dos parâmetros de exposição da câmera podem levar a imagens nas quais o cordão de adesivo se torne mais nítido, possibilitando uma detecção pelo algoritmo mais robusta;

Controle da largura do cordão de adesivo: um importante incremento à detecção de deficiências na aplicação do adesivo seria o controle da largura do cordão a partir da imagem capturada, como proposto por YAN, XU, *et al.* (2016). Este controle poderia identificar variações na largura do cordão, que também representam um potencial de infiltração de água, caso ocorram em uma grande área concentrada do cordão;

Integração com o CLP com uso de Ethernet: o uso da interface de rede disponível no *Raspberry Pi* para comunicação com o controlador da linha de aplicação de adesivos possibilitaria uma melhor comunicação de eventuais falhas com o operador da linha a partir da tela previamente existente, sincronismo do relógio do Raspberry Pi com o do CLP, que possui RTC, e cópia dos arquivos da memória do dispositivo através da rede, sem necessidade de conexão local de um dispositivo de armazenamento USB;

Utilização do hardware para outros tipos de inspeção visual: novos algoritmos podem ser desenvolvidos para realizar inspeções visuais como contagem de peças montadas, como parafusos montados nas rodas, presença de emblemas e acessórios, identificação de peças com variação entre modelos, como rodas, paridade de cor entre carroceria e espelhos entre outros.

Ajuste automático da posição da máscara: um algoritmo pode ser desenvolvido para que caso ocorra um evento que modifique a posição da câmera em relação à peça, como uma colisão contra o suporte da câmera, a posição da máscara seja ajustada automaticamente em relação à nova posição da peça na imagem.

REFERÊNCIAS

ANDERSON, D. R.; SWEENEY, D. J.; WILLIAMS, T. A. **Statistics for Business and Economics**. 11ª. ed. Mason, OH, EUA: South-Western Cengage Learning, 2011.

ANFAVEA. **Anuário da Indústria Automobilística Brasileira**. Associação Nacional dos Fabricantes de Veículos Automotores. São Paulo, p. 148. 2021.

ASM DIMATEC GMBH. Robot cell for adhesive application, 2019. Disponivel em: https://www.asmdimatec.de/en/roboterklebeanlage.html. Acesso em: 13 mar. 2021.

ATLAS COPCO. Dispensing system for final assembly, 2019. Disponivel em: https://www.atlascopco.com.cn/en/itba/joining-solutions/products/dispensing/sca-final-assembly-dispensing-system. Acesso em: 13 mar. 2021.

ATLAS COPCO. Glazing applications for final assembly, 2019. Disponivel em: https://www.atlascopco.com.cn/en/itba/joining-solutions/industry-solutions/automotive/final-assembly/dispensing-system/glazing. Acesso em: 18 fev. 2021.

BARELLI, F. **Introdução à Visão Computacional:** Uma abordagem prática com Python e OpenCV. São Paulo, Brasil: Casa do Código, 2018.

CAYMAN AUTO SERVICES LTD. Rain, Rain Go Away: Water Ingress Issues, 2015. Disponivel em: https://www.caymanautos.co.uk/water-ingress-issues/. Acesso em: 06 abr. 2021.

CHEN, S.; LIN, M. **A New Method for Automatic Quality Inspection of Glue-Line**. 2009 International Conference on Information Engineering and Computer Science. Wuhan, China: IEEE. 2009. p. 1-3.

CHIN, R. T.; HARLOW, C. A. Automated Visual Inspection: A Survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. PAMI-4, n. 6, p. 557-573, Nov. 1982. ISSN 1939-3539.

DAUDT, G.; WILLCOX, L. D. INDÚSTRIA AUTOMOTIVA. In: PUGA, F.; DE CASTRO, L. B. **Visão 2035:** Brasil, país desenvolvido: agendas setoriais para alcance da meta. 1ª. ed. Rio de Janeiro: Banco Nacional de Desenvolvimento Econômico e Social, 2018. p. 183-208.

DAUGMAN, J. G. **Computer Vision Computer Science Tripos:** 16 Lectures by J G Daugman. Cambridge: [s.n.]. 2010.

DAVIES, E. R. **Computer and Machine Vision:** Theory, Algorithms, Practicalities. 4ª. ed. Kidlington, Inglaterra: Academic Press, 2012.

DI LEO, G. et al. A vision system for the online quality monitoring of industrial manufacturing. **Optics and Lasers in Engineering**, v. 89, p. 162-168, mai. 2016. ISSN 0143-8166.

EMERALD GROUP PUBLISHING LIMITED. Bonding technology in cars: adhesives instead of rivets, screws and welding. **Assembly Automation**, v. 27, n. 2, abr. 2007. ISSN 0144-5154.

FORSYTH, D. A.; PONCE, J. **Computer Vision:** A Modern Approach. 2ª. ed. Upper Saddle River, NJ, EUA: Pearson Education, Inc., 2012.

GOLNABI, H.; ASADPOUR, A. Design and application of industrial machine vision systems. **Robotics and Computer-Integrated Manufacturing**, v. 23, n. 6, p. 630-637, dez. 2007. ISSN 0736-5845.

IBGE. **Rendimento de todas as fontes : 2019**. IBGE, Coordenação de Trabalho e Rendimento. Rio de Janeiro, p. 12. 2020. (978-85-240-4529-5).

KRIG, S. **Computer Vision Metrics:** Survey, Taxonomy, and Analysis. Nova Iorque, NY, EUA: Apress Media, LLC, 2014.

KRISHNA, R. **Computer Vision:** Foundations and Applications. [S.l.]: Stanford University, 2017.

KUKA ROBOTER GMBH. **KUKA.GlueTech 4.1**. V2 en. ed. Ausburg, Alemanha: [s.n.], 2012.

MALAMAS, E. N. et al. A survey on industrial vision systems, applications and tools. **Image and Vision Computing**, v. 21, p. 171-188, fev. 2003.

MORTIMER, J. Robots unchallenged as the auto industry's workhorse. **Industrial Robot: An International Journal**, v. 31, n. 3, p. 264–272, Jun 2004. ISSN 0143-991X.

NEWMAN, T. S.; JAIN, A. K. A Survey of Automated Visual Inspection. **Computer Vision and Image Understanding**, v. 61, n. 2, p. 231-262, mar. 1995. ISSN 1077-3142.

OPENCV. Open Source Computer Vision Library. **Image Processing**, 2021. Disponivel em: https://docs.opencv.org/master/d2/d96/tutorial_py_table_of_contents_imgproc.html. Acesso em: 23 mar. 2021.

OPENCV. Open Source Computer Vision Library. **Core Operations**, 2021. Disponivel em: https://docs.opencv.org/master/d7/d16/tutorial_py_table_of_contents_core.html. Acesso em: 30 mar. 2021.

PAGNUTTI, M. A. et al. Laying the foundation to use Raspberry Pi 3 V2 camera module imagery for scientific and engineering purposes. **Journal of Electronic Imaging**, v. 26, n. 1, p. 1-13, fev. 2017. ISSN 013014.

PINTO, W. T.; BIANCHI, R. A. C. **Comparação entre Dois Sistemas de Identificação de Falhas em Soldas na Produção de Amortecedores Veiculares**. Simpósio Brasileiro de Automação Inteligente. Fortaleza, Brasil: [s.n.]. 2013.

PXHERE. **aircraft_airplane_aviation_clouds_flight_flying_plane_sky**, 2017. Disponivel em: https://pxhere.com/en/photo/949873. Acesso em: 30 mar. 2021.

RASPBERRY PI FOUNDATION. **Raspberry Pi 3 Model B+**. Raspberry Pi Foundation. Cambridge, Inglaterra, p. 5. 2020.

RECLAME AQUI. Reclame Aqui. **Institucional**, 2020. Disponivel em: https://www.reclameaqui.com.br/institucional/. Acesso em: 22 abr. 2021.

RODRIGUES, R. Em um ano, preços dos carros mais vendidos subiram 9,4% em média. **Kelley Blue Book**, 2021. Disponivel em: https://www.kbb.com.br/detalhes-noticia/precos-carros-mais-vendidos-subiram-94-media/?id=3153. Acesso em: 29 jun. 2021.

SAINT-GOBAIN. Escolher um adesivo para colagem de vidro automotivo de qualidade é fundamental para a segurança do veículo, 2018. Disponivel em: https://www.saint-gobain-autover.com.br/content/escolher-um-adesivo-para-colagem-de-vidro-automotivo-de-qualidade-%C3%A9-fundamental-para. Acesso em: 18 Fev 2021.

SARAFYAN, I.; CATALDI, A. Infiltração de água no carro: descubra as causas e como evitar, 2017. Disponivel em: https://autoesporte.globo.com/servicos/noticia/2017/03/infiltracao-de-agua-no-carro-descubra-causas-e-como-evitar.ghtml. Acesso em: 06 abr. 2021.

SCA SCHUCKER GMBH & CO. KG. **Controle SYS 6000**. Bretten, Alemanha: [s.n.], 2013.

SHAPIRO, L.; STOCKMAN, G. **Computer Vision**. [S.l.]: Prentice Hall, 2000.

SHIN-ETSU CHEMICAL. **Functional Sealants:** Technical Manual. 1. ed. Tokyo, Japão: [s.n.], 2018.

SIKA. Ficha Técnica de Produto. **Sikaflex®-250 PC**, 2020. Disponivel em: https://bra.sika.com/content/dam/dms/br01/0/sikaflex-250-pc.pdf. Acesso em: 18 Fevereiro 2021.

SOMARATHNA, H. M. C. C. et al. The use of polyurethane for structural and infrastructural engineering applications: A state-of-the-art review. **Construction and Building Materials**, v. 190, p. 995-1014, jun. 2018. ISSN 0950-0618.

SUN, J.; SUN, Q.; SURGENOR, B. W. An adaptable automated visual inspection scheme through online learning. **The International Journal of Advanced Manufacturing Technology**, v. 59, n. 5-8, p. 655-667, jul. 2012.

SZELISKI, R. **Computer Vision:** Algorithms and Applications. New York, EUA: Springer Science & Business Media, 2010.

SZYCHER, M. **Szycher's Handbook of Polyurethanes**. 2nd. ed. Boca Raton, FL, EUA: Taylor & Francis Group, 2013.

UNIVERSITY OF TORONTO. 2D Image Digital Representation, 2018. Disponivel em: https://edtech.engineering.utoronto.ca/object/2d-image-digital-representation. Acesso em: 23 mar. 2021.

WARRANTY WEEK. **Worldwide Auto Warranty Expenses**. Warranty Week. Nova Iorque, NY, EUA. 2020. (1550-9214).

YAN, C. et al. **Width estimation of glue line for engine oil sump based on planar vision**. 2016 Chinese Control and Decision Conference (CCDC). Yinchuan, China: IEEE. 2016. p. 2816-2821.

APÊNDICE A - ÍNDICE DE NOVOS NEGÓCIOS POR MONTADORA NA PLATAFORMA RECLAME AQUI

A Tabela 2 foi elaborada a partir de levantamento de dados realizado na plataforma Reclame Aqui no dia 29/06/2021, considerando o total de reclamações para cada um dos 12 fabricantes de veículos com mais reclamações na plataforma, e seus respectivos índices de novos negócios (IN), que é um índice calculado a partir da resposta à pergunta “Voltaria a fazer negócios com a empresa?”, apresentada aos reclamantes após encerramento da reclamação. Os dados são relacionados a um período de 12 meses, entre 01/06/2020 e 31/05/2021. A média total foi calculada como média ponderada, considerando o total de reclamações como peso.

Tabela 2 – Reclamações e índice de novos negócios na plataforma Reclame Aqui

Fabricante	Reclamações em 12 meses	IN
Chevrolet	5.155	64%
Volkswagen	3.760	52%
Fiat	3.309	63%
Ford	2.834	69%
Jeep do Brasil	2.524	66%
Renault	2.235	76%
Hyundai Motor Brasil HB20 - Creta	1.804	57%
Nissan do Brasil	1.615	74%
Honda Automóveis	1.277	81%
Citroën	815	45%
CAOA Chery	803	60%
Peugeot do Brasil	552	46%
Total	26.683	64%

Fonte: Elaborado pelo autor

Desta forma, foi identificado que 36% dos reclamantes não voltariam a fazer negócio com as montadoras reclamadas.

APÊNDICE B - RECLAMAÇÕES RELACIONADAS A INFILTRAÇÃO DE ÁGUA EM AUTOMÓVEIS

A Tabela 3 foi elaborada a partir de pesquisa realizada no dia 08/04/2021 no site Reclame Aqui, considerando reclamações feitas entre 01/01/2020 e 04/04/2021 que mencionassem diretamente em sua descrição infiltração de água pelo para-brisas, ou infiltração de água na parte frontal da cabine, sem causa claramente definida, considerando os 12 fabricantes que mais apresentavam reclamações em geral no momento da pesquisa. Apenas a empresa CAOA Cherry não apresentou reclamações que atendessem aos critérios acima.

Ao todo, foram identificadas 63 reclamações no ano de 2020 e 19 reclamações em 2021, totalizando 82 reclamações diretamente ou potencialmente relacionadas a infiltração de água pelo para-brisas. Vale ressaltar que estes registros no site não representam o total de problemas relacionados a infiltração de água, mas dos problemas cuja solução foi insatisfatória aos clientes, fazendo com que buscassem ajuda além da que poderia ser obtida com as concessionárias e montadoras.

Tabela 3 – Reclamações de infiltração de água em automóveis por fabricante.

Fabricante	Quantidade de reclamações	Número de identificação das reclamações
Chevrolet	13	120613107, 120277849, 115274143, 114398699, 106743653, 104152235, 103377999, 102655733, 102212411, 102127633, 101532135, 101117561 e 100514497
Citroën	1	119675789
Fiat	11	119779831, 116990083, 115674209, 115028293, 110215841, 109712971, 105195289, 101385721, 101138749, 100083805 e 99729889
Ford	20	121973837, 121512231, 121007271, 119605569, 118058831, 118048051, 117499217, 115130869, 115066163, 112431925, 110187693, 108881273, 106678901, 105062073, 102412547, 100862757, 100782717, 99774399, 99263763 e 99150659
Honda Automóveis	3	118226695, 115558299 e 105041909
Hyundai Motors Brasil	7	121320145, 119655485, 117662607, 114409229, 114399239, 111202395 e 101137215
Jeep do Brasil	2	120837661 e 100144053
Nissan do Brasil	2	112496571 e 109113897

Fabricante	Quantidade de reclamações	Número de identificação das reclamações
Renault	6	121027349, 111140845, 101447501, 101118729, 99931311 e 98817493
Toyota	6	112091999, 105443297, 104272035, 103858881, 100314811 e 99837569
Volkswagen	11	121485749, 117951395, 109823797, 107561569, 100555335, 100426385, 100332789, 99957859, 99541025, 99213103 e 99035875

Fonte: Elaborado pelo autor

Até a data de 29/06/2021, apenas 28 das reclamações acima haviam sido finalizadas na plataforma, sendo que em 46% dos casos os reclamantes indicaram que não voltariam a fazer negócios com a empresa.

APÊNDICE C - CÓDIGO FONTE DO MÓDULO DE INSPEÇÃO DE ADESIVO

Classe pubead.py

```
import logging
import threading
import time
from datetime import datetime
from typing import Tuple

import cv2
import numpy as np
import math

# Create a custom logger
logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)


def date_time():
    return datetime.now().strftime("%Y%m%d-%H%M%S")


def find_first_peak_from_histogram(sect_img):
    """
    Returns the first peak level from the histogram.

    :param sect_img: (numpy.uint8) gray image to be analysed
    :return: (int) first peak level
    """
    # calculates the histogram of the roi
    hist = cv2.calcHist([sect_img], [0], None, [256], [0, 256])

    # init vars
    prev_lvl = 0
    i = 0

    # loops through histogram to find first peak. skips 0 to avoid black areas of the mask
    for i in range(1, 256):
        # if current level in histogram is lower then previous peak
        if prev_lvl > hist[i]:
            # exits loop
            break
        else:
            # updates peak level
            prev_lvl = hist[i] - 1
    return i


# noinspection PyUnusedLocal
def nothing(x):
    pass


def find_large_contours(thresh_img, min_area):
    """
    Finds the contours of a given binary image and filters them, returning only the ones larger than the specified
    minimum area

    :param thresh_img: binary image in which to find the contours
    :param min_area: minimum area to be used  to filter the areas of the contours
    :return: large_contours, areas - filtered contours, list of areas for each contour
    """
    contours, hierarchy = cv2.findContours(thresh_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours_s4 = filter(lambda contour: contour.size > 4, contours)
    large_contours = list(filter(lambda contour: cv2.contourArea(contour) >= min_area, contours_s4))
    areas = list(map(lambda contour: cv2.contourArea(contour), large_contours))
    return large_contours, areas


class BeadParams:
    def __init__(self):
        self.bead_valid = False
        self.mask_lines = []
        self.pu_areas = []
        self.min_contour_area = 0
        self.bead_name = ''
        self.roi_img_ratio = 20  # original: 25
        self.roi_size = 0
        self.rois_overlap = 0.67  # original: 0.5


class PUBead:
    def __init__(self, image=None, bead_name: str = ''):
        """
```

```
    :type image: bgr or grayscale cv2 image, to be used by the user on the creation of the new bead.
    :type bead_name: string - name of the new bead.
    """
    self.params = BeadParams()
    self._img = None
    self._mask = None
    self._thresh_img = None
    self._masked_img = None
    self._lock = threading.Lock()
    # if image argument is passed, run routine to create new bead
    if image is not None:
        self.create_new_bead(image, bead_name)

def clear_image_arrays(self):
    with self._lock:
        self._img = None
        self._mask = None
        self._thresh_img = None
        self._masked_img = None

def evaluate_bead(self, image) -> Tuple[bool, list]:
    """
    Evaluate the given image to check the quantity of pu beads present and their areas

    :param image: (cv2.mat) Grayscale image to be evaluated
    :return: True if bead approved. otherwise, false.
    """

    with self._lock:
        # loads the image to class attribute
        self.load_image(image)
        # if the loaded image is empty, return False
        if self._img is None:
            return False, []
        # build the mask image from the lines if it's empty
        if self._mask is None:
            self.create_mask_from_lines()
        # saves current image to the image log folder
        jpg_quality = 90
        file_name = f'./image_log/{self.params.bead_name}-{date_time()}.jpg'
        cv2.imwrite(file_name, self._img, [cv2.IMWRITE_JPEG_QUALITY, jpg_quality])
        # local copy of variables to release the thread lock sooner
        bead_name = self.params.bead_name
        area_tolerance = self.params.min_contour_area
        pu_areas = self.params.pu_areas
        # applies the threshold algorithm to segment the pu bead
        self.image_threshold()
        # finds the contours which areas are larger then min_contour_area
        contours, areas = find_large_contours(self._thresh_img, self.params.min_contour_area)
        # converts image to BGR to allow drawing the contours in color
        bgr_img = cv2.cvtColor(self._img, cv2.COLOR_GRAY2BGR)
    # draw the contours on the bgr image
    bgr_img = cv2.drawContours(bgr_img, contours, -1, (255, 0, 0), 4)
    # writes the image with contours to the disk
    file_name = './images/last_image.png'
    cv2.imwrite(file_name, cv2.resize(bgr_img, (800, 600)))

    # gets number of areas from the contours
    number_of_areas_found: int = len(areas)
    number_of_areas_expected: int = len(pu_areas)
    # return False if number of areas found different from expected
    if number_of_areas_found != number_of_areas_expected:
        logger.info(f"Part {bead_name} rejected. Incorrect number of areas. Expected: {number_of_areas_expected}. "
                    f"Found: {number_of_areas_found}")
        return False, areas
    # return False also if areas found are not within tolerance
    if not np.allclose(areas, pu_areas, 0, area_tolerance):
        logger.info(f"Part {bead_name} rejected. Beads areas out of tolerance. Expected: {pu_areas}. "
                    f"Found: {areas}")
        return False, areas
    # if nothing goes wrong, return True
    logger.info(f'Part {bead_name} approved. Areas found: {areas}')
    return True, areas

def create_mask_from_lines(self):
    """

    :return:
    """
    size = self._img.shape
    size_x = size[1]
    size_y = size[0]
    line_color = 255

    # (re)create blank black image of given size
    self._mask = np.zeros((size_y, size_x), np.uint8)
    # load the threshold image with a blank black image
    self._thresh_img = self._mask.copy()
```

```
        for line in self.params.mask_lines:
            # draw the lines on the mask
            cv2.line(self._mask, line.p1, line.p2, line_color, line.thickness)

        # binarize the image
        self.image_threshold()

    def image_threshold(self):
        # (re)apply the mask to the image
        self.create_masked_image()
        size = self._img.shape
        size_x = size[1]
        size_y = size[0]
        self._thresh_img = np.zeros((size_y, size_x), np.uint8)

        for line in self.params.mask_lines:
            # reapply the threshold for the rois along the lines and merge them to the thresh_img
            self._thresh_img = np.bitwise_or(self._thresh_img, self.roi_threshold(line.rois_p1))

    def create_masked_image(self):
        """
        Help function to apply the mask to the original image

        :return: self.masked_img
        """
        blur = cv2.medianBlur(self._img, 5)
        # create a masked image
        self._masked_img = np.bitwise_and(blur, self._mask)

    def roi_threshold(self, rois_p1):
        """
        Returns an image of the same size of self.img, with the threshold of the regions of interest on their
        original position

        :param rois_p1: list with the up left point (p1) of the regions of interest (ROIs)
        :return: (cv.mat) threshold of self.img applied on the given ROIs
        """
        # distance from the peak to threshold value
        distance_from_peak_level = 4
        # get number of rows and cols
        size = self._img.shape
        rows = size[0]
        cols = size[1]
        # create a blank black image
        blank_img = np.zeros((rows, cols), np.uint8)
        # initialize thresh_img with blank image
        partial_thresh_image = blank_img.copy()

        # loops through roi list
        for roi in rois_p1:
            # get row and col of current roi
            row = roi[0]
            col = roi[1]
            # get the section of image containing the roi
            sect_img = self._masked_img[row:row + self.params.roi_size, col:col + self.params.roi_size]
            # find first peak level from histogram section
            level = find_first_peak_from_histogram(sect_img) + distance_from_peak_level
            # thresholds the roi with the peak level
            ret, thresh_roi = cv2.threshold(sect_img, level, 255, cv2.THRESH_BINARY_INV)
            # blanks tmp image
            thresh_tmp = blank_img.copy()
            # paste roi to blank image
            thresh_tmp[row:row + self.params.roi_size, col:col + self.params.roi_size] = thresh_roi
            # performs 'or' operation with temp image and partial_thresh_image, to
            # update it with the current roi threshold
            partial_thresh_image = cv2.bitwise_or(thresh_tmp, partial_thresh_image)

        # re-apply the mask
        partial_thresh_image = np.bitwise_and(partial_thresh_image, self._mask)

        return partial_thresh_image

    def load_image(self, image):
        # checks if the image is empty
        if image is None:
            logger.error('Image is empty.')
            return
        # checks whether the image is grayscale
        if len(image.shape) > 2:
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            logger.warning('Image passed to the class was not grayscale. Consider changing it.')
        # loads the parameter to the function attribute
        self._img = image

    def create_new_bead(self, image, bead_name):
        self.load_image(image)
        self.params.bead_valid = False
```

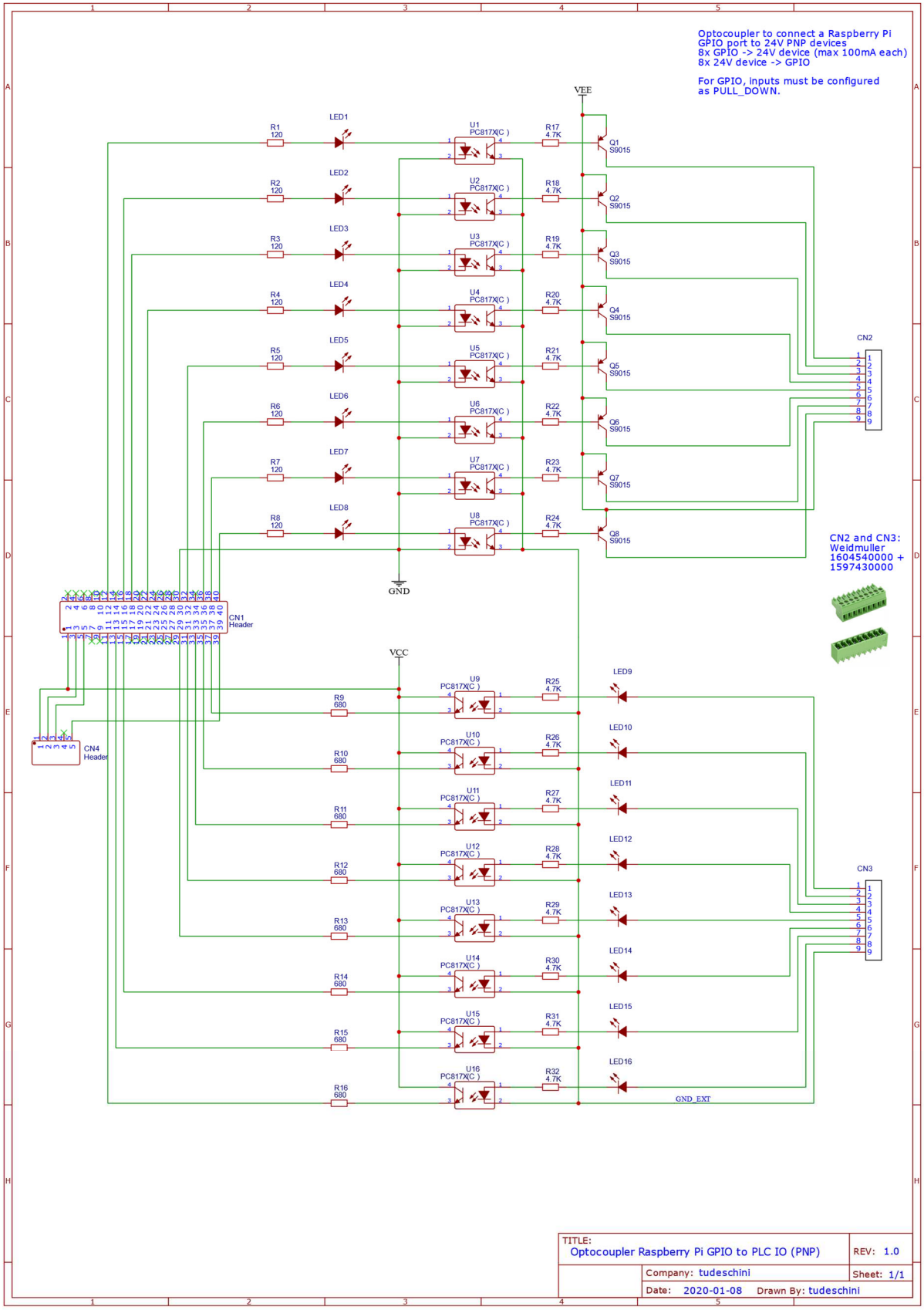
```
        if self._img is None:
            logger.error('Empty image passed to the create_new_bead routine.')
            return False

        if len(bead_name) == 0:
            logger.error('Bead name is empty. Cannot continue with bead creation.')
            return False

        # calculates size of regions of interest from image size and ratio
        dims = self._img.shape
        self.params.roi_size = int(max(dims[0], dims[1]) / self.params.roi_img_ratio)
        # create a blank black mask with the same size of the image
        self._mask = np.zeros((dims[0], dims[1]), np.uint8)
        # load the threshold image with a blank black image
        self._thresh_img = self._mask.copy()
        # opens GUI to allow user to draw mask
        self.draw_mask()
        # get quantity of mask lines and bead areas
        number_of_mask_lines = len(self.params.mask_lines)
        number_of_areas = len(self.params.pu_areas)
        # if there are no lines, return False
        if number_of_mask_lines == 0:
            logger.error('No line was drawn by the user. Bead creation aborted.')
            return False
        # if there are no areas found, return false
        if number_of_areas == 0:
            logger.error('No pu beads were detected on the drawn lines. Bead creation aborted.')
            return False
        # updates the bead name
        self.params.bead_name = bead_name
        # saves current image to the image log folder
        jpg_quality = 90
        file_name = f'./image_log/{self.params.bead_name}-ref-{date_time()}.jpg'
        cv2.imwrite(file_name, self._img, [cv2.IMWRITE_JPEG_QUALITY, jpg_quality])
        # sets bead_valid to True and returns True
        self.params.bead_valid = True
        logger.info(f'Bead {bead_name} created with {number_of_mask_lines} mask lines and '
                    f'{number_of_areas} beads.')
        return True

    def modify_bead(self, image):
        with self._lock:
            self.load_image(image)

            if self._img is None:
                logger.error('Empty image passed to the modify_bead routine. Exiting...')
                return False

            # calculates size of regions of interest from image size and ratio
            dims = self._img.shape
            self.params.roi_size = int(max(dims[0], dims[1]) / self.params.roi_img_ratio)

            # load the threshold image with a blank black image
            self._thresh_img = np.zeros((dims[0], dims[1]), np.uint8)
            # saves current mask_lines and pu_areas, to restore them in the case the user mess things up
            backup_mask_lines = self.params.mask_lines
            backup_pu_areas = self.params.pu_areas
            # build the mask image from the lines if it's empty
            if self._mask is None:
                self.create_mask_from_lines()
            # opens GUI to allow user to draw mask
            self.draw_mask()
            # get quantity of mask lines and bead areas
            number_of_mask_lines = len(self.params.mask_lines)
            number_of_areas = len(self.params.pu_areas)
            # if there are no mask lines, restore values anr return False
            if number_of_mask_lines == 0:
                logger.error('No line was drawn by the user. Bead modification aborted. Original values restored.')
                self.params.mask_lines = backup_mask_lines
                self.params.pu_areas = backup_pu_areas
                return False
            # if no area was found, restore values and return False
            if number_of_areas == 0:
                logger.error('No pu beads were detected on the drawn lines. Bead modification aborted. Original values '
                             'restored')
                self.params.mask_lines = backup_mask_lines
                self.params.pu_areas = backup_pu_areas
                return False
            # if everything goes well:
            # saves current image to the image log folder, as a reference
            jpg_quality = 90
            file_name = f'./image_log/{self.params.bead_name}-ref-{date_time()}.jpg'
            cv2.imwrite(file_name, self._img, [cv2.IMWRITE_JPEG_QUALITY, jpg_quality])
            # write log and return True
            logger.info(f'Bead {self.params.bead_name} modified. {number_of_mask_lines} mask lines and '
                        f'{number_of_areas} beads.')
            return True
```

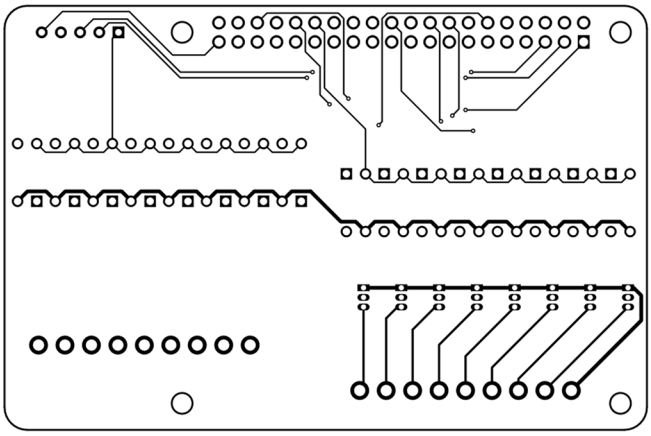
```
def rename_bead(self, bead_name):
    if bead_name != '':
        with self._lock:
            self.params.bead_name = bead_name
    else:
        logger.error('Name bead is empty. Cannot rename the bead.')

def draw_roi_squares(self, cv_color=(0, 0, 255)):
    """
    Draw squares representing the regions of interest (ROI) of the image

    :param cv_color: (tuple, uint8) - color in BGR. default: red (0, 0, 255)
    :return: (cv.mat) img_squares - original image with the squares representing the ROIs
    """
    # converts img to BGR
    img_bgr = cv2.cvtColor(self._img, cv2.COLOR_GRAY2BGR)

    img_squares = None

    # loop through rois list
    for line in self.params.mask_lines:
        for roi in line.rois_p1:
            # flips the row and col to adapt numpy array to opencv Mat
            cv_p1 = (roi[1], roi[0])
            # add roi_size
            cv_p2 = tuple(np.add(cv_p1, self.params.roi_size))
            # draw rectangle
            img_squares = cv2.rectangle(img_bgr, cv_p1, cv_p2, cv_color, 3)
    return img_squares

def draw_mask(self):
    """
    GUI to allow the user to draw the mask on the top of the original image. User gets visual feedback on the
    results of the application of the drawn mask.

    :return: nothing
    """
    # init vars
    drawing = False  # true if mouse is pressed
    ix, iy = -1, -1  # fist point of the line
    backup_of_mask = None  # temporary mask
    update_screen = True
    # gets image size
    sx, sy = self._img.shape
    # calculates the ratio to be used to relate the thickness chosen by the user with the actual line thickness.
    # number 600 was defined empirically resulting in optimal line thickness for the pu line thickness of the images
    # used during the development. then, it could be changed if the thickness of the PU bead changes on the image,
    # e.g., if the distance between the camera and the part changes.
    thickness_ratio = sy / 700
    # calculates the minimum area to be considered when filtering
    self.params.min_contour_area = int(sx * sy / 10000)
    # creates blank black BGR mask, to be used to draw the color mask lines
    mask_bgr = np.zeros((sx, sy, 3), np.uint8)
    # copies the B&W mask to the green channel of the BGR mask
    mask_bgr[:, :, 1] = self._mask
    # creates a BGR version of the original image, to match with the BGR mask
    img_bgr = cv2.cvtColor(self._img, cv2.COLOR_GRAY2BGR)

    # mouse callback function
    # noinspection PyUnusedLocal
    def draw_line(event, x, y, flags, param):
        nonlocal drawing, ix, iy, backup_of_mask, thickness, mask_bgr, reference_img, update_screen
        # selects actions from mouse events
        if event == cv2.EVENT_LBUTTONDOWN:  # left button pressed
            # sets flag to true
            drawing = True
            # updates first point of the line wit current mouse position
            ix, iy = x, y
            # copies the bgr mask to the backup of the mask
            backup_of_mask = mask_bgr.copy()
        elif event == cv2.EVENT_MOUSEMOVE:  # mouse movement
            # if drawing is set
            if drawing:
                # noinspection PyUnresolvedReferences
                # restores the mask to its backup
                mask_bgr = backup_of_mask.copy()
                # draws a line from the initial position to the current mouse cursor position
                cv2.line(mask_bgr, (ix, iy), (x, y), (0, 255, 0), thickness)
                # merges the mask and th image
                reference_img = cv2.addWeighted(mask_bgr, 0.5, img_bgr, 1.0, 30)
                # sets flag to true
                update_screen = True
        elif event == cv2.EVENT_LBUTTONUP:  # left button released
            # resets flag to false
            drawing = False
            # draws a line from the initial position to the current mouse cursor position
            cv2.line(mask_bgr, (ix, iy), (x, y), (0, 255, 0), thickness)
```

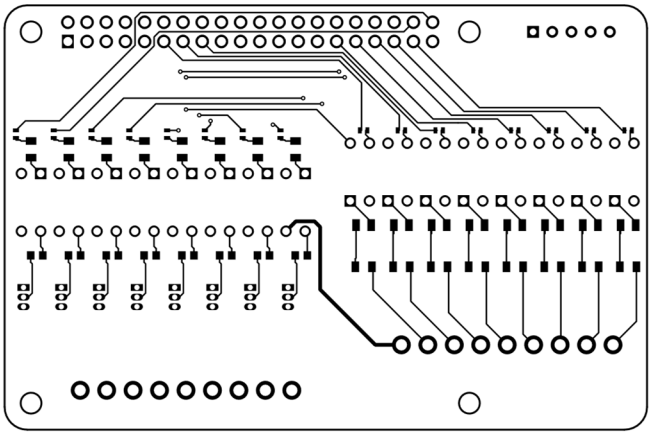
```
                # adds a MaskLine object with the current line settings
                self.params.mask_lines.append(MaskLine((ix, iy), (x, y), thickness,
                                                       self.params.roi_size, self.params.rois_overlap, self._img))
                # adds the added line to the mask
                self.add_line_to_mask()
                # updates the bgr mask and the image shown to the user
                update_mask()

            def update_mask():
                nonlocal mask_bgr, reference_img, update_screen
                # copy the B&W mask to the green channel of the BGR mask
                mask_bgr[:, :, 1] = self._mask
                # merge the original image with the drawn mask
                reference_img = cv2.addWeighted(mask_bgr, 0.5, img_bgr, 1.0, 30)
                large_contours, self.params.pu_areas = find_large_contours(self._thresh_img, self.params.min_contour_area)
                cv2.drawContours(reference_img, large_contours, -1, (255, 0, 0), 3)
                cv2.putText(reference_img, f"Beads found: {len(self.params.pu_areas)}", (100, 100),
                            cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 6)
                # sets flag to true
                update_screen = True

            # setup the window to be used by the user to draw the mask
            cv2.namedWindow('image', cv2.WINDOW_NORMAL)
            # setup the mouse callback routine and attaches it to the window
            cv2.setMouseCallback('image', draw_line)

            # create trackbar for thickness change
            lbl_thickness = 'Thickness'
            cv2.createTrackbar(lbl_thickness, 'image', 5, 10, nothing)

            reference_img = None
            update_mask()

            while 1:
                if update_screen:
                    cv2.imshow('image', reference_img)
                    update_screen = False
                k = cv2.waitKey(1) & 0xFF
                time.sleep(0.05)
                if k == ord('d'):
                    if len(self.params.mask_lines):
                        self.params.mask_lines.pop()
                        self.create_mask_from_lines()
                        update_mask()
                elif k == ord('-'):
                    pass
                elif k == 27:
                    break
                # exit loop if window is closed by OS' 'close' button
                if not cv2.getWindowProperty('image', cv2.WND_PROP_VISIBLE):
                    break
                # get current positions of trackbar
                t_bar_thickness = cv2.getTrackbarPos(lbl_thickness, 'image')
                thickness = int((t_bar_thickness + 8) * thickness_ratio)

            cv2.destroyAllWindows()

        def add_line_to_mask(self):
            """
            Updates the mask with a recently added line and updates the thresholded image with the new mask line

            """

            line_color = 255

            # draw the lines on the mask
            cv2.line(self._mask, self.params.mask_lines[-1].p1, self.params.mask_lines[-1].p2, line_color,
                     self.params.mask_lines[-1].thickness)
            self.create_masked_image()
            # reapply the threshold for the rois along the line and merge them to the thresh_img
            self._thresh_img = np.bitwise_or(self._thresh_img, self.roi_threshold(self.params.mask_lines[-1].rois_p1))


    class MaskLine:
        def __init__(self, p1, p2, thickness, roi_size, rois_overlap, image):
            self.p1 = p1
            self.p2 = p2
            self.thickness = thickness
            self.rois_p1 = []
            self.find_rois_from_line(roi_size, rois_overlap, image)

        def find_rois_from_line(self, roi_size: int, rois_overlap: float, image):
            rows, cols = image.shape

            half_box = int(roi_size / 2)
            distance = roi_size * (1 - rois_overlap)

            # calculates the angle of the line. constant 0.1 added to avoid division by zero
```

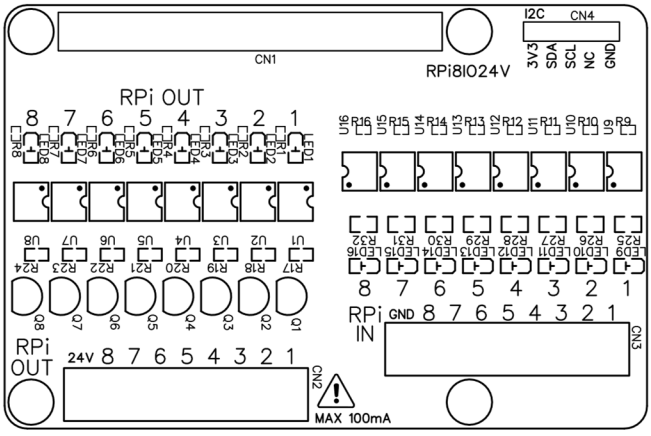
```
alpha = math.atan((self.p2[1] - self.p1[1]) / (self.p2[0] - self.p1[0] + 0.1))
# if x2 > x1, add pi to get the angle in the correct quadrant
if self.p2[0] < self.p1[0]:
    alpha += math.pi

# calculates the line length
line_length = math.hypot((self.p2[1] - self.p1[1]), (self.p2[0] - self.p1[0]))

# calculates the x and y steps
step_col = int(distance * math.cos(alpha))
step_row = int(distance * math.sin(alpha))
# calculates the number of boxes to cover the line
n_boxes = int(math.ceil((line_length + roi_size) / distance))

# init row and col
col = self.p1[0] - half_box
row = self.p1[1] - half_box

for box in range(n_boxes):
    box_col = col
    box_row = row
    # adjust the points to fall inside image limits
    if 0 > col:
        box_col = 0
    if col > (cols - roi_size):
        box_col = cols - roi_size
    if 0 > row:
        box_row = 0
    if row > (rows - roi_size):
        box_row = rows - roi_size
    # append to list
    self.rois_p1.append((box_row, box_col))

    # increment col and row
    col += step_col
    row += step_row
```

APÊNDICE D - PLACA PARA CONEXÃO ENTRE SINAIS 3,3V DO RASPBERRY PI E 24V DO CLP









Diagrama, desenhos da placa, lista de materiais e descrição do projeto estão disponíveis gratuitamente na internet e podem ser acessados através do link: https://easyeda.com/tudeschini/iocouplerrpi-24v_copy

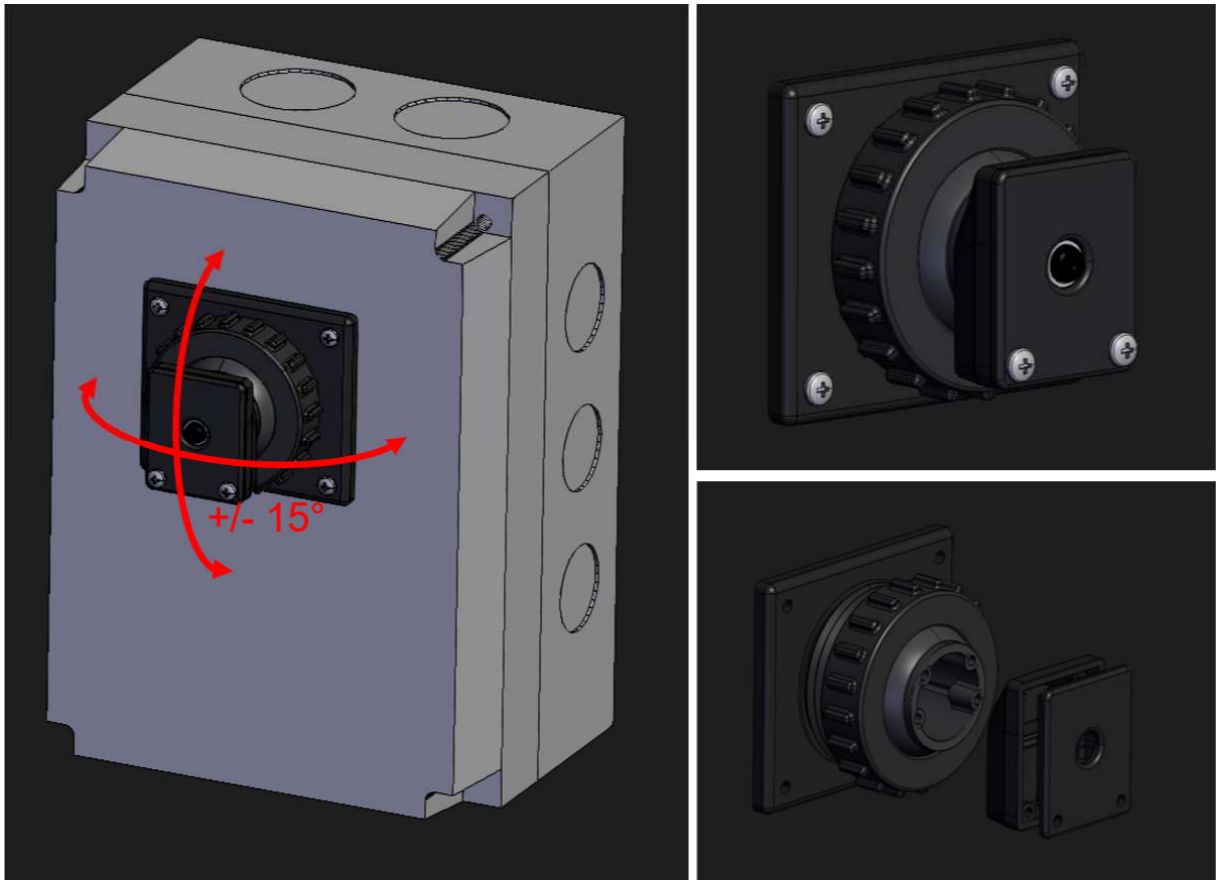
APÊNDICE E - DESENVOLVIMENTO DE UM SUPORTE COM ÂNGULO AJUSTÁVEL PARA MÓDULO DE CÂMERA DO RASPBERRY PI

Para possibilitar o ajuste de ângulo para posicionamento da câmera, um suporte foi especialmente desenvolvido para o módulo de câmera do *Raspberry Pi* com o uso do programa *FreeCAD* v0.18.

O suporte foi desenhado em cinco peças, para facilitar a impressão e sua montagem: uma base com rosca externa, um anel com rosca interna, uma esfera para ajuste do ângulo, uma caixa para o módulo de câmera e uma tampa. Além das peças impressas em plástico foram utilizados 10 parafusos 2,9x9,5mm.

Com este suporte, é possível movimentar a câmera em dois sentidos em um ângulo de até 15° em relação ao eixo central, possibilitando maior agilidade e flexibilidade em seu posicionamento.

Figura 43 – Detalhes do projeto 3D do suporte da câmera



Fonte: Elaborado pelo autor

Os arquivos editáveis do programa FreeCAD, bem como os em formato STEP e STL para impressão foram disponibilizados em um repositório de projetos 3D, e podem ser acessados em https://grabcad.com/library/raspberry-pi-camera-support-1.