CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Samyla de Souza Dutra

UTILIZAÇÃO DO ALGORITMO HAARCASCADE E TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS PARA AVALIAÇÃO DE JATOS PULVERIZADORES

Timóteo

2019

Samyla de Souza Dutra

UTILIZAÇÃO DO ALGORITMO HAARCASCADE E TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS PARA AVALIAÇÃO DE JATOS PULVERIZADORES

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Odilon Corrêa da Silva

Timóteo

2019

Samyla de Souza Dutra

UTILIZAÇÃO DO ALGORITMO HAARCASCADE E TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS PARA AVALIAÇÃO DE JATOS PULVERIZADORES

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, campus Timóteo, como requisito parcial para obtenção do título de Engenheiro de Computação.

Trabalho aprovado. Timóteo, 10 de Julho de 2019:

Prof. Me. Odilon Corrêa da Silva
Orientador

Prof. Dr. Elder de Oliveira Rodrigues
Professor Convidado

Prof. Dr. João Batista Queiroz Zuliani
Professor Convidado

Analista de Sistemas Cristóvão Nery Giacomin
Convidado

Timóteo
2019

Agradecimentos

Sou grata a Deus pela vida e saúde concedidos graciosamente todos os dias. Aos meus pais que estiveram comigo nessa trajetória me dando total apoio e suporte. À minha irmã Sulaany e minha sobrinha Nauany pela compreensão dos dias de renúncia do lazer com elas para a dedicação aos estudos. A todos os meus familiares que acreditam em mim e torcem para o meu sucesso profissional. Meu esforço e dedicação herdei de vocês!

Aos meus amigos do CEFET-MG: Aline, Ariadne, Athos, Fernanda, Juliana, Júlio, Rodrigo e Vitor que estiveram presentes e fizeram com que a caminhada da graduação pudesse ser ainda mais prazerosa. Aos meus professores que desde o ensino técnico se dedicaram em passar o conhecimento e ensinar. Tenho por vocês grande admiração por desempenharem essa profissão e entusiasmo de ser este o caminho para a construção de um mundo melhor. Meu muito obrigada!

Ao meu orientador pela paciência, disponibilidade e conhecimento compartilhado que fez jus ao nome de orientador nos momentos obscuros do desenvolvimento e foi também amigo quando necessário. Aos professores Jozelmo, Fabrício e João Batista pelos esclarecimentos. Aos alunos da monitoria de GAAV pelo incentivo e aprendizado mútuo. Aos funcionários do CEFET-MG que com gentileza e suporte facilitaram minha chegada até aqui. Passei uma vida com vocês e sou extremamente grata por tudo!

Aos meus amigos do estágio na USIMINAS. Aprendi um mundo com vocês! Obrigada pela troca de conhecimento e conversas produtivas. Um agradecimento especial ao Giacomin pela ideia inicial do projeto e ao Bruno pelas dicas e esclarecimentos.

”Diante da vastidão do tempo e da imensidão do universo, é um prazer para mim dividir uma época com vocês!” Carl Sagan.

“A alegria está no outro lado do trabalho árduo. Esse é um elemento básico de todo crescimento. Parte da maturidade está em compreender o princípio da satisfação protelada. Se você não pode aceitar o sofrimento do aprendizado e quer satisfação imediata, você perde as maiores recompensas da vida.”.

John Piper

Resumo

A aplicação de sistemas de visão computacional para a identificação de objetos e determinação de suas características básicas é cada vez mais frequente nas indústrias para otimização do processo em larga escala. Porém, fazer com que o computador seja capaz de reconhecer objetos em imagens ou sequências de imagens ainda é um dos grandes desafios da computação. O objetivo desse trabalho é detectar e extrair características de jatos pulverizadores utilizados em indústrias para o processo siderúrgico. As imagens foram obtidas a partir de uma bancada de teste e submetidas ao algoritmo *Haarcascade* que utiliza a técnica de *machine learning* para a detecção da região de interesse. Posteriormente foram submetidas à técnicas de suavização utilizando filtro Gaussiano e de mediana, segmentação pelo operador de Canny e a transformada de Hough da biblioteca OpenCV implementada na linguagem Python. Com as características básicas da imagem processada, implementou-se um algoritmo utilizando relações trigonométricas para extrair características fundamentais para avaliação do jato como o ângulo e a cobertura teórica. Como resultado, o algoritmo *Haarcascade* apresentou um percentual de acerto de 80,77% para a detecção da região de interesse e o algoritmo desenvolvido para o cálculo do ângulo e da cobertura teórica apresentou 92,06% de acerto. Como resultado do teste realizado com vídeo de 30 *frames*/segundo foi possível processar 50% da taxa de *frames*/segundo do vídeo original e o algoritmo para cálculo dos parâmetros apresentou um percentual de 76,22%, 77,57% e 88,12% de acerto de acordo com as classificações do jato.

Palavras-chave: Machine learning, Transformada de Hough, Jato pulverizador, Siderurgia.

Abstract

The application of computer vision systems to the identification of objects and determination of their basic characteristics are increasingly frequent in the industries for large scale process optimization. However, making the computer capable of recognizing objects in images or sequences of images is still one of the great challenges of computing. The objective of this work is to detect and extract characteristics of spray nozzles used in industries for the steel industry process. The images were obtained from the test bench and submitted to the Haarcascade algorithm for the detection of the region of interest and later to the smoothing techniques using Gaussian and median filters, segmentation by the Canny operator and the Hough transform of the OpenCV library implemented in Python language. With the basic characteristics of the processed image, an algorithm was implemented using trigonometric relations to extract fundamental characteristics for jet evaluation such as angle and theoretical coverage. For the images submitted to the test Haarcascade presented a percentage of correctness of 80.77% for the detection of the region of interest and the algorithm developed for the calculation of the angle and the theoretical coverage presented 92.06% of correctness. For the test performed with video of 30 frames/second it was possible to process 50% rate of the original frame/second and the algorithm for calculation of the parameters presented a percentage of 76,22%, 77,57% and 88,12% of correctness according to the jet classifications.

Keywords: Machine Learning, Hough Transform, Spray nozzle.

Lista de ilustrações

Figura 1 – Processos que utilizam bicos pulverizadores na indústria siderúrgica 14
Figura 2 – Desenho esquemático da linha de laminação a quente de uma siderurgia e os principais pontos da linha que utilizam bicos pulverizadores para diferentes funções. 15
Figura 3 – Desenho esquemático da utilização do bico pulverizador para o processo de descarepação (remoção de carepa) na laminação de tiras a quente em uma siderurgia. 16
Figura 4 – Imagem exemplo dos jatos emitidos pelos bicos sendo utilizados na área de descarepação da laminação de tiras a quente. 16
Figura 5 – Cobertura e ângulo de pulverização do jato utilizado no descarepador. 17
Figura 6 – Passos fundamentais para o processamento digital de imagens. 18
Figura 7 – (A) Imagem representada como uma matriz de intensidade visual. (B) Imagem representada como uma matriz numérica 2-D (0, .5 e 1 correspondem ao preto, cinza e branco, respectivamente). 19
Figura 8 – Funcionamento da filtragem espacial linear utilizando uma máscara 3x3. . . 21
Figura 9 – Exemplo de correlação unidimensional. 21
Figura 10 – Produto ponto a ponto. 22
Figura 11 – Exemplo de correlação e convolução. 22
Figura 12 – (a) função Gaussiana bidimensional com média $(0, 0)$ e $\sigma = 1$. (b) matriz com pesos Gaussianos para serem aplicados em uma imagem 24
Figura 13 – Aplicação do filtro Gaussiano em uma imagem real utilizando $\sigma = 3$ e uma janela deslizante 5 x 5. 24
Figura 14 – Exemplo de aplicação do filtro de mediana. 25
Figura 15 – Aplicação do filtro de mediana com janela deslizante de ordem 3, aplicada a uma imagem sob o ruído sal e pimenta. 25
Figura 16 – Usando o gradiente para determinar a intensidade e a direção da borda em um ponto. A borda é perpendicular à direção do vetor gradiente no ponto onde o gradiente é computado. Cada quadrado da figura representa um pixel. 26
Figura 17 – Algoritmo para determinação de pontos de borda em uma imagem. 27
Figura 18 – Aplicação do filtro de Sobel para o eixo x e eixo y. 28
Figura 19 – Aplicação do filtro de Laplace. 28
Figura 20 – Aplicação do filtro de Canny com limiar mínimo de 100 e máximo de 200. . . 29
Figura 21 – Aplicação da segmentação por binarização com limiar adaptativo sendo o Gaussiano da biblioteca OpenCV. 30
Figura 22 – Princípio da transformada de Hough. (a) plano da imagem; (b) espaço de parâmetros. 31
Figura 23 – (a) (ρ, θ) Parametrização do plano xy. (b) Curvas senoidais no plano $\rho\theta$; (c) Divisão do plano $\rho\theta$ em células acumuladoras. 31
Figura 24 – Funcionamento em cascata do classificador. 33

Figura 25 – Filtros de Haar: Os pixels das regiões brancas são subtraídos dos pixels da região preta correspondente. Por exemplo, a máscara 1(a) apenas subtrai o valor de um pixel do seu vizinho à direita. A máscara 2(c) subtrai o pixel central dos pixels logo acima e abaixo. A máscara 3(a) subtrai um pixel central dos seus 8-vizinhos. 33
Figura 26 – Exemplos de atributos Haar calculadas sobre uma imagem. 34
Figura 27 – Exemplo de código LBP calculado para um pixel de uma imagem 16x16. . . 34
Figura 28 – Sistemas de Reconhecimento de Padrões 35
Figura 29 – Aprendizagem supervisionada 36
Figura 30 – Etapas de um sistema de visão computacional genérico. 38
Figura 31 – Fluxo de processos do sistema contador de telhas. 39
Figura 32 – Fluxograma proposto para a etapa de pré-processamento 40
Figura 33 – Metodologia utilizada para implementação da proposta. 43
Figura 34 – Etapas para o funcionamento do haarcascade 44
Figura 35 – Esquema para obtenção das imagens dos jatos emitidos pelo bico descarepador 45
Figura 36 – Imagem do jato em bancada de testes com a ROI delimitada pelo *bounding box* em verde. 46
Figura 37 – Imagens positivas utilizadas para treinamento do classificador 47
Figura 38 – Imagens da ROI para o parâmetro bgthresh igual a 100 48
Figura 39 – Imagens da ROI para o parâmetro bgthresh igual a 20 48
Figura 40 – Detecção do algoritmo haarcascade para o primeiro experimento utilizando 400 imagens negativas 50
Figura 41 – Detecção de falsos positivos detectados pelo algoritmo haarcascade. 51
Figura 42 – Detecção do algoritmo haarcascade para o segundo experimento 51
Figura 43 – Detecção do algoritmo haarcascade para o experimento com o limiar para scaleFactor e minNeighbors especificados 52
Figura 44 – Imagem da ROI original (a), com filtro mediana (b), Gaussiano (c) e filtro Gaussiano e mediana (d) aplicado. 54
Figura 45 – Imagens de bordas em declives submetidas a um ruído gaussiano, os resultados da aplicação da derivada de primeira ordem, os resultados da aplicação da derivada de segunda ordem e os respectivos perfis de intensidade para cada caso: (a) Ruído com desvio padrão de 0,0; (b) Ruído com desvio padrão de 0,1; (c) Ruído com desvio padrão de 1,0; (d) Ruído com desvio padrão de 10,0. 55
Figura 46 – Imagem da ROI e os resultados do processamento após aplicação de filtro Gaussiano (a); após aplicação de filtro de mediana (b) e filtro Gaussiano e mediana aplicados simultaneamente (c). 56
Figura 47 – Resultado após aplicação do filtro Gaussiano, Mediana e operador de Canny para delimitar contorno do jato com uma ROI mais próxima do observador (a) e uma ROI mais distante do observador (b). 58

Figura 48 – Primeira imagem é o jato real, a segunda do jato com delimitação de bordas e terceira com a transformada de Hough para detecção das retas segundo os critérios estabelecidos. 59
Figura 49 – Posição relativa entre duas retas concorrentes. 59
Figura 50 – Relação trigonométrica para o cálculo da abertura do jato e da cobertura teórica. 60
Figura 51 – Resultado da aplicação da transformada de Hough com detecção de retas colineares (a) e retas que não são de interesse para extração de característica (b). 61
Figura 52 – Resultado da aplicação da do operador de Canny (a) e da transformada de Hough (b) para o vídeo da classe DNEX 1328. 62
Figura 53 – Resultado dos ângulos e cobertura teórica calculados pelo algoritmo desenvolvido para classe do jato 3040E. 63
Figura 54 – Resultado dos ângulos e cobertura teórica calculados pelo algoritmo desenvolvido para classe do jato DNEX 1328. 64
Figura 55 – Resultado do processamento da ROI não delimitada corretamente pelo Haarcascade da imagem 23 para a detecção de bordas (a) e detecção de retas (b). 64
Figura 56 – Resultado do processamento para a detecção de bordas (a) e detecção de retas (b) em uma imagem ROI não delimitada corretamente na etapa do *Haarcascade*. 65
Figura 57 – Resultado dos ângulos e cobertura teórica calculados pelo algoritmo desenvolvido para classe do jato 3040EN PRO. 66
Figura 58 – Resultado da aplicação do método desenvolvido para os *frames* da classe DNEX 1328, TC3040E e TC3040EN PRO. 69

Lista de abreviaturas e siglas

ROI	*Region Of Interest*
OpenCV	*Open Source Computer Vision Library*
RGB	*Red, Green, Blue*
APC	Análise de Componentes Principais
MLP	*Perceptron* de Múltiplas Camadas
SOM	Redes Neurais Auto-Organizáveis
LBP	*Local Binary Patterns*
KNN	*K — Nearest Neighbors*
2D	Duas dimensões
PDI	Processamento Digital de Imagens
GPU	Unidade de Processamento Gráfico

Sumário

1 INTRODUÇÃO 13
1.1 Motivação 15
1.2 Objetivos 17
1.2.1 Objetivos Específicos 17
1.3 Justificativa 17

2 REFERENCIAL TEÓRICO 18
2.1 Processamento Digital de Imagens 18
2.2 Pré-processamento 20
2.2.1 Filtragem no Domínio Espacial 20
2.2.2 Suavização 21
2.2.2.1 Filtro de média 23
2.2.2.2 Filtro Gaussiano 23
2.2.2.3 Filtro de Mediana 24
2.3 Segmentação 25
2.3.1 Operadores de Gradiente 26
2.3.2 Operador Sobel 27
2.3.3 Operador Laplaciano 28
2.3.4 Detector de bordas de Canny 29
2.3.5 Segmentação por binarização adaptativa 29
2.3.6 Transformada de Hough 30
2.3.6.1 Transformada Probabilística de Hough 32
2.4 Representação e descrição 32
2.4.1 Algoritmo HaarCascade 32
2.5 Reconhecimento e interpretação 34
2.5.1 Aprendizagem supervisionada 35
2.5.2 AdaBoost 35
2.6 Biblioteca OpenCV 36
2.7 Plataforma Anaconda 37

3 TRABALHOS RELACIONADOS 38

4 MATERIAIS E MÉTODOS 41

5 APLICAÇÃO DO ALGORITMO HAARCASCADE 44
5.1 Aquisição de imagens 44
5.2 Definição da região de interesse 45
5.3 Definição das imagens negativas 46
5.4 Definição das imagens positivas 46

5.4.1 Geração do vetor de positivas 47
5.5 Treinamento do cascade 49
5.6 Resultados obtidos 50
5.6.1 Experimento 1 50
5.6.2 Experimento 2 51

6 ALGORITMO PROPOSTO PARA EXTRAÇÃO DE CARACTERÍSTICAS . . 53
6.1 Pré-processamento 53
6.1.1 Aplicação do filtro Gaussiano 53
6.1.2 Aplicação do filtro de mediana 54
6.2 Segmentação 57
6.2.1 Operador de Canny 57
6.2.2 Transformada de Hough 58
6.3 Algoritmo proposto para extração das características 59
6.3.1 Cálculo dos parâmetros de interesse 60
6.3.2 Restrições aplicadas para o cálculo do ângulo 61
6.3.3 Testes realizados em vídeo 61
6.4 Avaliação do processo 62

7 CONCLUSÃO 70

REFERÊNCIAS 72

1 Introdução

“Aquele que conhece o inimigo e a si mesmo,
lutará cem batalhas sem perigo de derrota.”
Sun Tzu

Na computação, é vigente a demanda em conseguir simular as capacidades humanas para a realização de funções relacionadas à análise de imagens (MARENGONI; STRINGHINI, 2009). Essa simulação deve ter a habilidade em extrair apenas informações importantes a partir de uma cena repleta de informações, entre as quais muitas irrelevantes, habilidade para fazer inferências a partir de informações incompletas e, principalmente, conseguir identificar um objeto ou padrão em uma imagem com a maior independência possível em relação a fatores como mudanças de posição, tamanho, orientação e variações no ponto de vista (ARAÚJO, 2009).

A aplicação desses sistemas computacionais para a identificação de objetos e determinação de suas características básicas são cada vez mais frequentes nas indústrias para otimização do processo em larga escala. O conjunto de técnicas utilizadas para captura, aprimoramento, representação e transformação de imagens com o auxílio do computador é denominada Processamento Digital de Imagens (PDI). Em geral, entre os pesquisadores não existe um consenso de onde o PDI termina e começam outras áreas relacionadas como a análise de imagens e a visão computacional (PEDRINI; SCHWARTZ, 2008). Sistemas que utilizam essas técnicas vem se tornando cada vez mais comum no ambiente fabril. Eles substituem sistemas e procedimentos de alto custo, que demandem tempo ou que não oferecem a exatidão necessária (SCOMAZZON, 2014).

Sistemas de visão computacional também estão sendo utilizados na área da medicina para detecção de manchas de pele auxiliando no diagnóstico de câncer (OLIVEIRA et al., 2012) (SOARES, 2008) e avaliação de tipos sanguíneos dentre os tipos ABO e os fatores Rh (OLIVEIRA et al., 2012); na área geográfica para obter informações a partir de sensoriamento remoto (GRIGIO, 2003); na área agrícola para detecção e reconhecimento de variedades de plantações e vegetações (LULIO, 2011), caracterização física de grãos (GUEDES et al., 2011), reconhecimento de padrões de plantas (SILVA, 2014) e variedades de soja (KHATCHATOURIAN; PADILHA, 2008); na área de microscopia para classificação microestrutural de elementos químicos (FERNANDES et al., 2012), análise microestrutural em asfaltos para determinar propagação de defeitos (ZHANG; REN; LU, 2016); na área de reconhecimento facial (MANUEL, 2015) e detecção facial (MANUEL et al., 2016), detecção de objetos (PINTO, 2015); na área industrial para reconhecimento de falhas em processos (KUROIWA; CARRO, 2015) e automatização de processos lentos e demorados (SILVA, 2014); na área de trânsito detectando tráfego de automóveis, sinais de trânsito, pedestres e demais elementos (NEU, 2014).

No ambiente industrial as aplicações mais comuns para sistemas de visão computacional são divididas por Golnabi e Asadpour (2007), em quatro categorias:

- Monitoramento e controle de processos (PFEIFER; WIEGERS, 2000);
- Reconhecimento e classificação de peças (RUDEK; COELHO; JR, 2001), (BENTO, 2016);
- Inspeção (ALMEIDA; CORSO; JUNIOR, 2009), (YOON; CHUNG, 2004);
- Orientação e controle de manipuladores robóticos e mecanismos (MENG; ZHUANG, 2007).

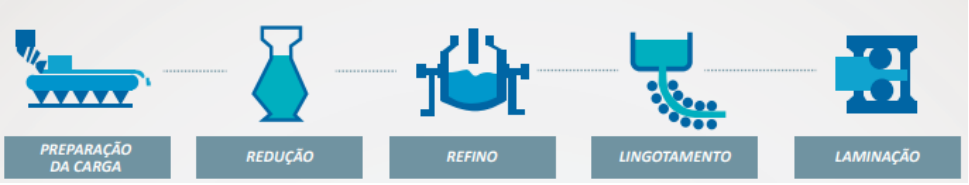
Para a área siderúrgica, sistemas de visão computacional já foram utilizados para automatizar o processo de inspeção de defeitos em chapas e classificá-los no processo de fabricação do aço (QUEIROZ, 2015), (MARTINS, 2010), analisar irregularidades na produção (BENTO, 2016), auxiliar operadores para análise nos sistemas de automação (SALIS; PEREIRA, 2007) e avaliar o desempenho eletromagnético de aço elétrico (FILHO et al., 2017).

Atualmente, para a produção do aço, o processo siderúrgico dispõe de objetos responsáveis por realizar a remoção de óxidos de ferro presente na composição química do material, emissão de água ou ar para a mistura química, pulverização nas áreas primárias e resfriamento do material processado. Esses objetos são chamados de bicos de pulverização.

Em um cenário generalístico, os bicos de pulverização são utilizados em escala industrial emitindo jatos de água, elementos químicos, ou ar sob grande pressão em diversos ramos. Os principais ramos são o mercado de alimentos, farmácia e biologia, lavagem de automóveis, madeira composta, papel e celulose, petroquímica, produtos químicos, siderurgia, agropecuária dentre outras. Na indústria siderúrgica, os bicos pulverizadores são utilizados principalmente no processo de laminação de tiras a quente. Neste processo os bicos são utilizados para a remoção de óxidos agregados no aço, denominados carepa. A siderurgia utiliza água na maioria de seus processos produtivos e a eficiência na utilização desse recurso é um fator muito importante na atividade industrial tanto do ponto de vista econômico quanto ambiental.

Como mostrado na figura 1, os bicos de spray são utilizados na área de refino primário para amenizar a emissão de particulados onde a matéria prima é armazenada; na área de redução onde o processo de aquecimento da solução nos alto-fornos precisa ser controlada; na área de refino para emissão de elementos nos convertedores; na área de lingotamento para resfriamento e solidificação do aço em placas e por fim na área de laminação a quente onde a emissão de água com grande pressão dos bicos de spray são responsáveis por realizar a descarepação na chapa que está sendo laminada.

Figura 1 – Processos que utilizam bicos pulverizadores na indústria siderúrgica



Fonte: Spraying Systems Co

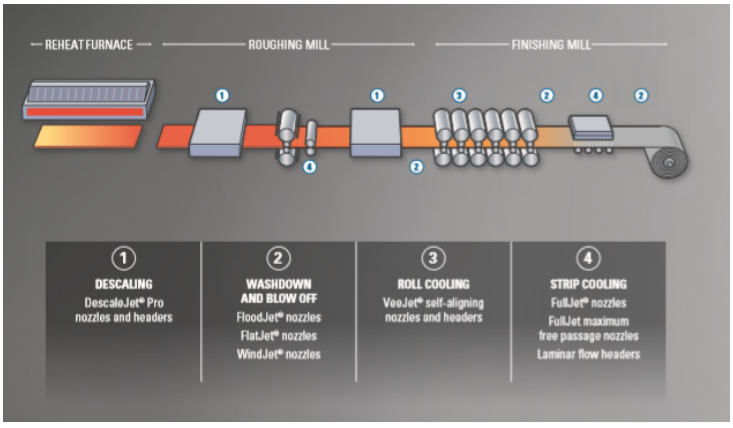
Atualmente, a avaliação da qualidade do bico pulverizador utilizado na laminação de tiras a quente é realizada manualmente. Um operador utiliza métodos físicos para medição e observação visual para determinar a qualidade do jato emitido.

Contudo, essa avaliação está suscetível a interferências humanas uma vez que um operador pode ser mais rigoroso que outro. Ocasionalmente, fatores ambientais, físicos e comportamentais (como estresse, fadiga, sono, fome) podem causar divergências na avaliação.

1.1 Motivação

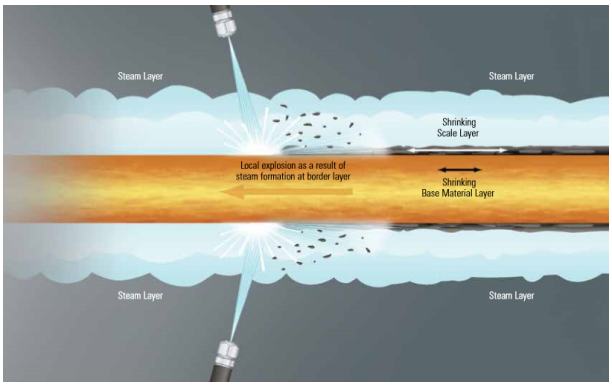
O aço, produto da siderurgia, é um elemento fundamental para quase todos os segmentos da economia. É matéria-prima essencial para a construção civil, o setor automotivo e a produção de máquinas e equipamentos. Para garantir a produtividade, economia nos processos e a qualidade do produto final, é fundamental que a indústria siderúrgica utilize sistemas de pulverização precisos. O uso dos bicos de pulverização certos e em bom estado está diretamente ligado à qualidade do aço. Uma bobina de aço com defeito, por exemplo, será inutilizada, representando um prejuízo de aproximadamente R$ 80 mil para a usina (SPRAYINGSYSTEMS, a). Uma parada na produção provocada por problemas com os bicos de pulverização também será extremamente onerosa (SPRAYINGSYSTEMS, b). A figura 2 mostra a principal área que utiliza os bicos pulverizadores para controle de temperatura do material e remoção de impurezas (figura 3).

Figura 2 – Desenho esquemático da linha de laminação a quente de uma siderurgia e os principais pontos da linha que utilizam bicos pulverizadores para diferentes funções.



Fonte: Spraying Systems Co

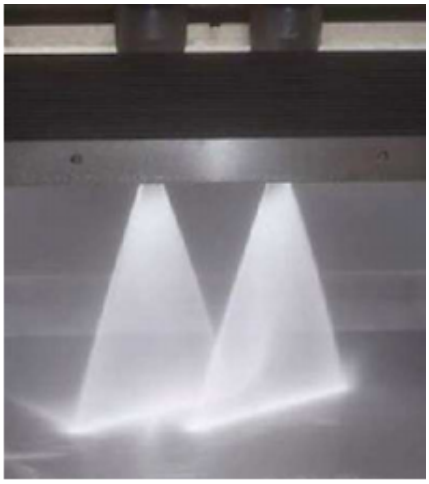
Figura 3 – Desenho esquemático da utilização do bico pulverizador para o processo de descarepação (remoção de carepa) na laminação de tiras a quente em uma siderurgia.



Fonte: Spraying Systems Co catálogo B628

A automatização do processo de avaliação do jato é importante, uma vez que implica em ganho de tempo para determinação da qualidade e cumprimento de um padrão fixo estabelecido. Consequentemente aumenta-se o lucro no processo em que o bico é utilizado já que não apresentará defeitos na linha de produção uma vez que já terá sido avaliado na bancada de testes. A figura 4 mostra um exemplo dos jatos sendo utilizados em bancada de teste.

Figura 4 – Imagem exemplo dos jatos emitidos pelos bicos sendo utilizados na área de descarepação da laminação de tiras a quente.



Fonte: Lechler Co

Sendo assim, de quais formas as técnicas de visão computacional e processamento digital de imagens podem ser utilizadas para auxiliar no processo de avaliação dos jatos emitidos pelos bicos pulverizadores?

1.2 Objetivos

Este trabalho tem como objetivo geral desenvolver um método para detectar e extrair características de imagens ou sequências de imagens de jatos emitidos por um bico pulverizador.

1.2.1 Objetivos Específicos

Para alcançar o objetivo geral proposto, foram definidos os seguintes objetivos específicos:

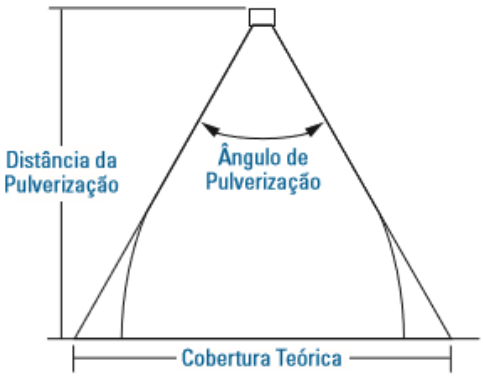
- Detectar áreas de interesse na imagem utilizando algoritmo de aprendizagem supervisionado;
- Extrair características das imagens como o ângulo e a cobertura teórica do jato;
- Avaliar o formato do jato em relação a uma escala de especificações técnicas pré-definidas;
- Avaliar o desempenho de acertos das classificações do método a partir de um estudo de caso utilizando os jatos em bancada de teste.

1.3 Justificativa

A avaliação dos jatos de bicos pulverizadores, atualmente, é realizada de forma manual a partir de medições físicas por profissionais da área. Podendo sofrer assim, influência de fatores externos e oscilações do comportamento humano impedindo uma padronização do método e sua confiabilidade.

A avaliação consiste em determinar parâmetros essenciais para a validação positiva/-negativa do jato. As principais, como mostrada na figura 5, são a determinação do ângulo de incidência de pulverização, a distância da pulverização e a comparação entre a cobertura teórica e real obtida.

Figura 5 – Cobertura e ângulo de pulverização do jato utilizado no descarepador.



Fonte: Spraying Systems Co

2 Referencial Teórico

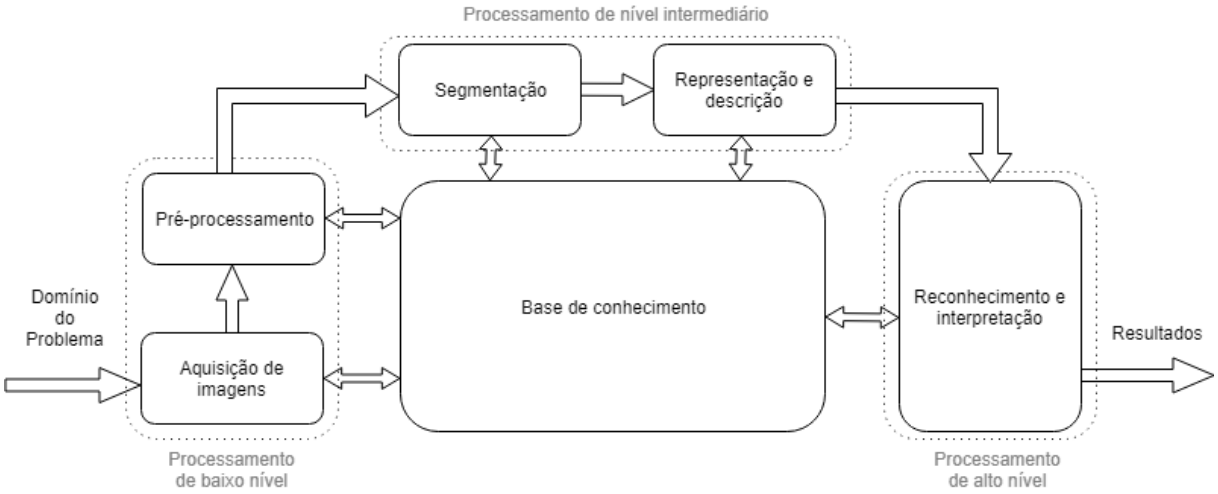
“Conheça o inimigo e a si mesmo e você obterá vitória sem qualquer perigo; conheça o terreno e as condições da natureza, e você será sempre vitorioso.”
Sun Tzu

2.1 Processamento Digital de Imagens

O processamento digital de imagens consiste no conjunto de técnicas empregadas para a captura, aprimoramento, representação e transformação de imagens com o auxílio do computador (PEDRINI; SCHWARTZ, 2008).

Gonzales e Woods (2010) divide conceitualmente esse processo em três áreas básicas como sendo (1) processamento de baixo nível, (2) processamento de nível intermediário e (3) processamento de alto nível. O processamento de nível baixo envolvem operações primitivas, como pré-processamento de imagens para reduzir o ruído, o realce de contraste e o aguçamento de imagens. Um processo de nível baixo é caracterizado pelo fato de tanto a entrada quanto a saída serem imagens. O processamento de nível médio envolve tarefas como a segmentação (separação de uma imagem em regiões ou objetos), a descrição desses objetos para reduzi-los a uma forma adequada para o processamento computacional e a classificação (reconhecimento) de objetos individuais. Um processamento de nível médio é caracterizado pelo fato de suas entradas, em geral, serem imagens, mas as saídas são atributos extraídos dessas imagens (bordas, contornos e a identidade de objetos individuais). Por fim, o processamento de nível alto envolve "dar sentido"a um conjunto de objetos reconhecidos, como na análise de imagens e realizar as funções cognitivas normalmente associadas à visão. Essas etapas são ilustradas conforme a imagem a seguir.

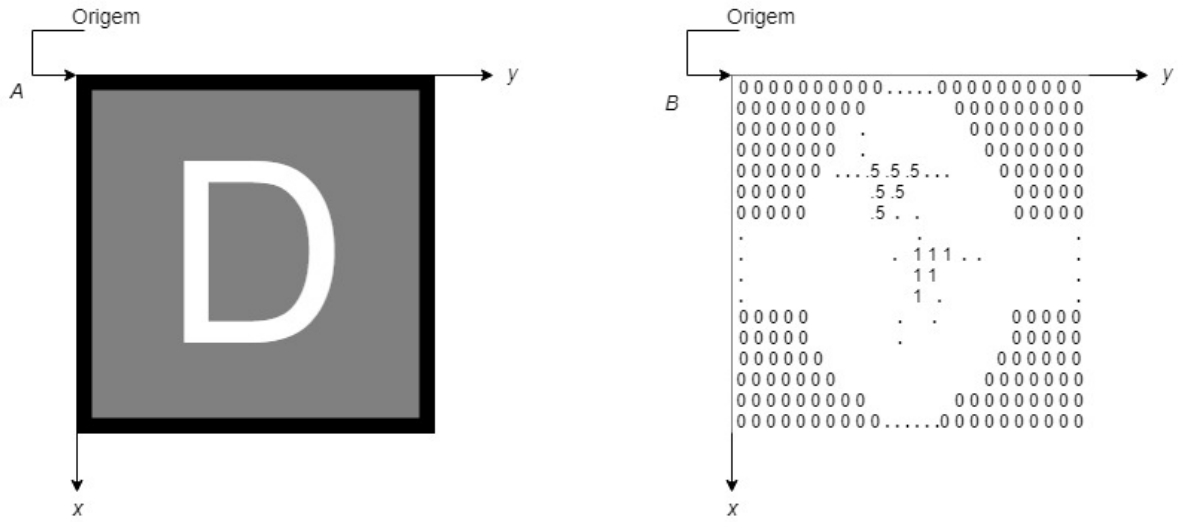
Figura 6 – Passos fundamentais para o processamento digital de imagens.



Fonte: Adaptado (GONZALES; WOODS, 2010)

A representação de uma imagem, segundo Gonzales e Woods (2010) pode ser feita graficamente como uma matriz de intensidade visual ou uma matriz numérica. A figura 7 ilustra essa definição:

Figura 7 – (A) Imagem representada como uma matriz de intensidade visual. (B) Imagem representada como uma matriz numérica 2-D (0, .5 e 1 correspondem ao preto, cinza e branco, respectivamente).



Fonte: Adaptado (GONZALES; WOODS, 2010)

Gonzales e Woods (2010) define uma imagem digital como sendo

> *uma função bidimensional, f(x,y), em que x e y são coordenadas espaciais (plano), e a amplitude de f em qualquer par de coordenadas (x,y) é chamada de intensidade ou nível de cinza da imagem nesse ponto. Quando x, y e os valores de intensidade de f são quantidades finitas e discretas, chamamos de imagem digital. O campo do processamento digital de imagens se refere ao processamento de imagens digitais por um computador digital. (GONZALES; WOODS, 2010, p. 36).*

Portanto, uma imagem digital pode ser escrita como uma matriz numérica MxN como mostrada na equação 2.1.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & ... & f(0,M-1) \\ f(1,0) & f(1,1) & ... & f(1,M-1) \\ . & & & \\ . & & & \\ . & & & \\ f(N-1,0) & f(N-1,1) & ... & f(N-1,M-1) \end{bmatrix} \tag{2.1}$$

Os dois lados dessa equação são formas equivalentes de representar quantitativamente uma imagem digital. No lado direito, cada elemento dessa matriz é chamado de elemento de imagem, elemento pictório, pixel ou pel. Pixel é o termo mais utilizado para representar os elementos em uma imagem digital. Sendo assim, para o processamento, tratamento

e extração de características das imagens, são utilizadas operações matemáticas. A seguir são descritas as formas principais para o tratamento e as operações utilizadas.

2.2 Pré-processamento

A etapa de pré-processamento tem como objetivo aprimorar a qualidade da imagem e prepará-la para as etapas seguintes de processamento (QUEIROZ, 2015). É nesta etapa que as imagens são suavizadas dilatadas, reduzidas ou recortadas a fim de reduzir ruídos, suprimir ou evidenciar características.

2.2.1 Filtragem no Domínio Espacial

Segundo Pedrini e Schwartz (2008), o domínio de imagem refere-se ao conjunto de pixels que compõe uma imagem, ao próprio plano da imagem. No domínio espacial, o nível de cinza de um ponto $f(x, y)$ após a transformação depende do valor do nível de cinza da vizinhança desse ponto $f(x, y)$.

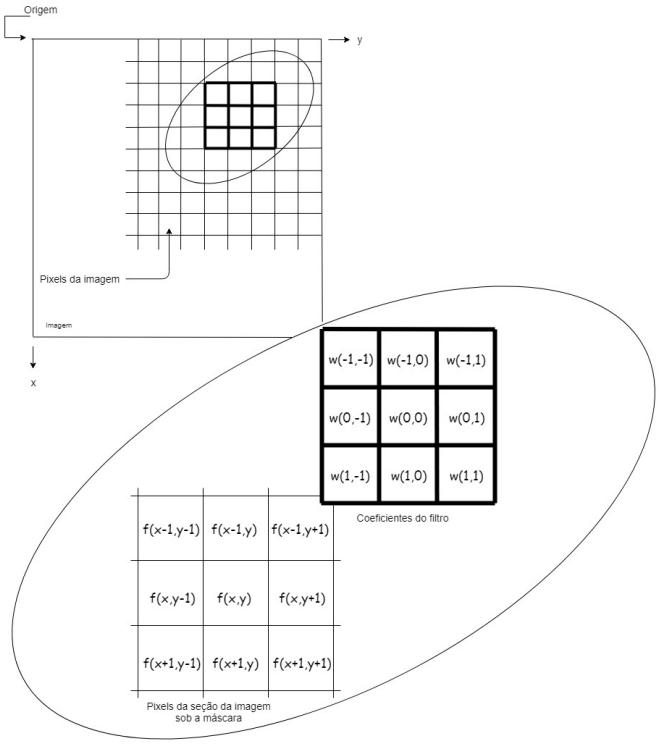
A operação de filtragem produz uma nova imagem a partir da imagem de entrada. Entretanto, cada pixel da imagem resultante depende apenas dos pixels da imagem original, ou seja, os resultados de uma operação efetuada não afetam os resultados dos outros pixels.

A figura 8 ilustra o funcionamento da filtragem espacial linear utilizando uma vizinhança 3 x 3. Em qualquer ponto (x, y) da imagem, a resposta $g(x, y)$, do filtro é a soma dos produtos dos coeficientes do filtro com os pixels da imagem englobados pelo filtro como mostrado pela equação 2.2:

$$g(x,y) = w(-1,-1)f(x-1,y-1)+w(-1,0)f(x-1,y)+...+w(0,0)f(x,y)+...+w(1,1)f(x+1,y+1) \tag{2.2}$$

Em geral, a máscara que percorre a imagem pode ter ordem m x n em que m e n são inteiros positivos ímpares. Entretanto, por uma questão de simetria, a literatura sugere a utilização de matrizes quadradas e por questões de eficiência computacional, normalmente são selecionados valores pequenos para n.

Figura 8 – Funcionamento da filtragem espacial linear utilizando uma máscara 3x3.



Fonte: (GONZALES; WOODS, 2010)

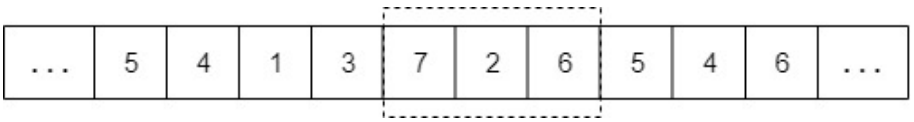
2.2.2 Suavização

A etapa de suavização das imagens consiste em aplicar esses filtros espaciais resultantes do pré-processamento. Dois conceitos estão relacionados à suavização por meio da filtragem espacial, a correlação e a convolução.

Segundo Gonzales e Woods (2010), a correlação é o processo de mover uma máscara pela imagem e calcular a soma dos produtos em cada posição. O funcionamento da convolução é o mesmo, exceto o fato de o primeiro filtro ser rotacionado a 180º. A melhor forma de compreender esses conceitos é por meio de exemplos.

Considere uma imagem unidimensional **f** representada pelo vetor da figura 9 em que uma operação de correlação pela filtragem da média é aplicada. Essa operação consiste em substituir cada pixel da imagem pela média de seu nível de cinza e de seus dois vizinhos.

Figura 9 – Exemplo de correlação unidimensional.



Fonte: Adaptado de (PEDRINI; SCHWARTZ, 2008)

Considerando uma janela de $n = 3$ temos que o cálculo da média para o pixel com o valor 2, por exemplo, produzirá o valor 5, resultado da média aritmética entre 7, 2 e 6. Para o caso da filtragem da média, é possível realizar o deslocamento de uma máscara com pesos iguais a $1/3$, em que cada um dos valores dos pixels é multiplicado por esse peso e então somados. A máscara $(1/3, 1/3, 1/3)$ forma o filtro, a aplicação desse filtro a cada um dos pixels da imagem corresponde ao processo de correlação.

A correlação para o caso bidimensional é similar. Considerando que a imagem e o filtro possuem agora duas dimensões, a correlação é definida como a soma da operação ponto a ponto entre as matrizes como ilustra a figura 10

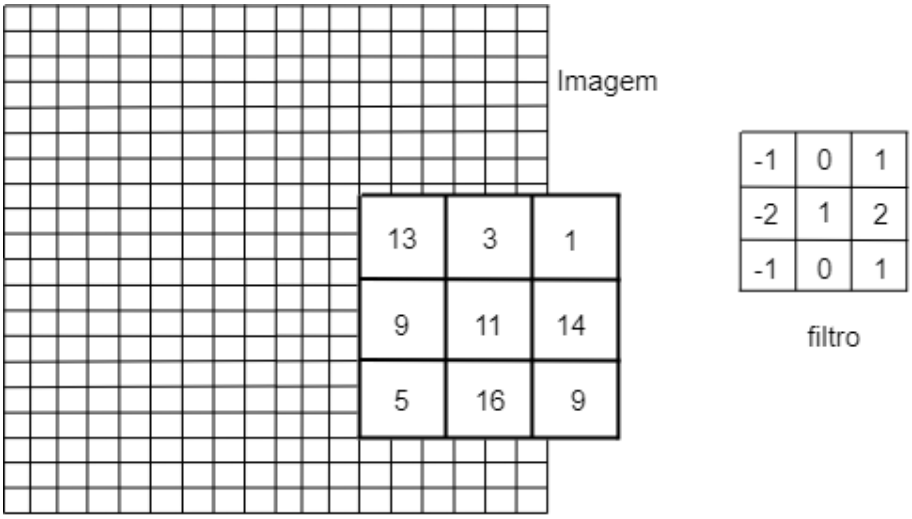
Figura 10 – Produto ponto a ponto.

13	3	1		-1	0	1		-13	0	1
9	11	14	*	-2	1	2	=	-18	11	24
5	6	9		-1	0	1		-5	0	9

Fonte: Elaborado pela autora.

A convolução consiste em um processo similar à correlação, com a diferença de que o filtro percorrido na região deve sofrer uma rotação de 180º antes de ser aplicado à imagem. A figura 11 ilustra um exemplo de correlação e convolução. O resultado da correlação para a região em destaque é igual a $13*(-1)+3*0+1*1+9*(-2)+11*1+14*2+5*(-1)+6*0+9*1 = 22$, o resultado da convolução, por sua vez, é igual a $13 * 1 + 3 * 0 + 1 * (-1) + 9 * 2 + 11 * 1 + 14 * (-2) + 5 * 1 + 6 * 0 + 9 * (-1) = 9$

Figura 11 – Exemplo de correlação e convolução.



Fonte: Adaptado de (PEDRINI; SCHWARTZ, 2008)

É válido destacar que a correlação e a convolução são idênticas quando o filtro é simétrico.

Os filtros podem ser classificados como lineares ou não lineares. A diferença entre eles é o tipo de operação realizada entre a máscara e o pixel $f(x, y)$ da imagem. Filtros lineares calculam o valor resultante do pixel $f'(x, y)$ como uma combinação linear dos níveis de cinza em uma vizinhança local do pixel $f(x, y)$ na imagem original. Ou seja, a operação é facilmente revertida tais como filtro de média, filtro de operação ponto a ponto, etc. Já os filtros não lineares não podem ser revertidos. Tais como filtros de mediana, filtro de mínimo, máximo e Gaussiano.

2.2.2.1 Filtro de média

O filtro de média é um filtro linear, classificado como passa-baixas, ou seja, para suavização de imagens. Ele substitui cada pixel da imagem pelo valor médio de sua vizinhança. Quanto maior a ordem da matriz que representa a máscara, maior será o número de pixels vizinhos considerados, logo, mais intenso será o efeito de suavização.

Na prática, apenas o filtro de média aplicado sozinho não é muito utilizado. Ele serve de base para os filtros Gaussiano e Mediana.

2.2.2.2 Filtro Gaussiano

O filtro gaussiano é um filtro não linear passa-baixas, isto é, atenua as altas frequências que estão relacionadas com a informação de detalhes da imagem. Este apresenta bons resultados no tratamento de imagens com ruído gaussiano. Diferente do filtro de média, o filtro gaussiano possui um parâmetro que define o grau de suavização. Esse parâmetro, denominado σ (sigma) refere-se ao desvio padrão da função gaussiana. Na prática, quanto maior o valor desse parâmetro, mais intensa será a suavização. A equação 2.3 mostra a função Gaussiana discreta onde o valor de σ é definido por um número inteiro positivo. .

$$G(x, y) = \frac{1}{2\pi\sigma^2} exp \frac{-(x^2 + y^2)}{2\sigma^2} \tag{2.3}$$

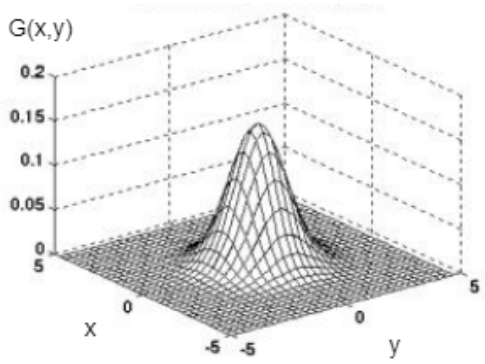
O filtro Gaussiano segue o comportamento da função Gaussiana ilustrada na figura 12 (a) que em notação matricial pode ser descrita como mostrada na figura 12 (b). Assim, é possível obter como resultado, uma imagem suavizada de maneira que os pontos de borda não sofram alterações bruscas.

Segundo Pedrini e Schwartz (2008), podemos destacar as características do filtro gaussiano como:

- Em duas dimensões, funções Gaussianas são simétricas com relação à rotação. Isso significa que o grau de suavização realizado pelo filtro será o mesmo em todas as direções;
- A suavização da imagem é realizada por meio da substituição de cada pixel por uma média ponderada dos pixels vizinhos, tal que o peso dado a um vizinho decresce monotonicamente com a distância do pixel central;

Figura 12 – (a) função Gaussiana bidimensional com média $(0, 0)$ e $\sigma = 1$. (b) matriz com pesos Gaussianos para serem aplicados em uma imagem

(a)



(b)

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

Fonte: Adaptado de (PEDRINI; SCHWARTZ, 2008)

- O grau de suavização do filtro Gaussiano está relacionado com o parâmetro σ. Quanto maior o valor de σ, maior a largura do filtro Gaussiano e maior o grau de suavização.

Como ilustrado na figura 13, o efeito *borramento* permite suprimir ruídos presentes na imagem (como a rua e os telhados) mas sem perder completamente as bordas principais da imagem. A aplicação do filtro Gaussiano nesta imagem foi utilizando utilizando $\sigma = 3$ e uma janela deslizante de 5 x 5.

Figura 13 – Aplicação do filtro Gaussiano em uma imagem real utilizando $\sigma = 3$ e uma janela deslizante 5 x 5.

Fonte: Elaborado pela autora.

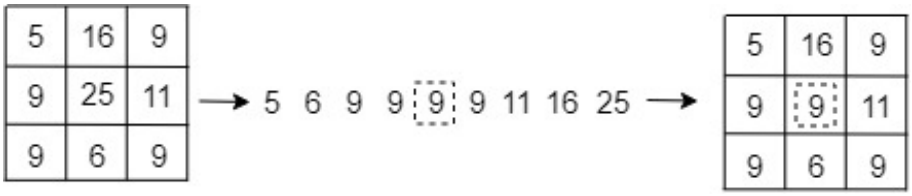
2.2.2.3 Filtro de Mediana

O filtro de mediana é não linear e apresenta bons resultados no tratamento de ruído do tipo *sal e pimenta*. Por ser um filtro passa-baixa, ele atua suavizando as transições abruptas da imagem e utilizando o conceito de estatística (cálculo da mediana), evita a suavização próximo às bordas preservando, assim, regiões interessantes de contorno.

Matematicamente, este filtro atua alterando o valor de cada pixel-alvo pela mediana estatística dos valores dos pixels vizinhos. Como mostrado na imagem 21, a operação matri-

cial do filtro de mediana consiste em realizar a operação estatística de mediana dos valores originais da região a ser aplicada na imagem e inserir o resultado no pixel-alvo.

Figura 14 – Exemplo de aplicação do filtro de mediana.



Fonte: Elaborado pela autora

A figura 15 ilustra o resultado do filtro de mediana aplicado utilizando a janela deslizante de ordem 3. A primeira imagem é a imagem original, a segunda com ruído do tipo *sal e pimenta* e a terceira o resultado após aplicação do filtro de mediana.

Figura 15 – Aplicação do filtro de mediana com janela deslizante de ordem 3, aplicada a uma imagem sob o ruído sal e pimenta.

Fonte: Elaborado pela autora

2.3 Segmentação

A etapa de segmentação consiste em delimitar as regiões de borda, localização e a forma dos objetos. Segundo Pedrini e Schwartz (2008), processar uma imagem de modo a segmentar um número de objetos, possivelmente em diferentes posições e com diferentes tamanhos e formas, é uma tarefa difícil e extremamente dependente da correta extração de características de objetos, especialmente em imagens ruidosas.

A maioria dos algoritmos de segmentação baseiam-se em uma das seguintes propriedades básicas de valores de intensidade: descontinuidade e similaridade. Na primeira categoria, a abordagem consiste em dividir uma imagem com base nas mudanças bruscas de intensidade, como as bordas. As abordagens da segunda categoria são baseadas na divisão da imagens em regiões de acordo com um conjunto de critério pré-denidos, como acontece no caso de limiarização de imagens (GONZALES; WOODS, 2010).

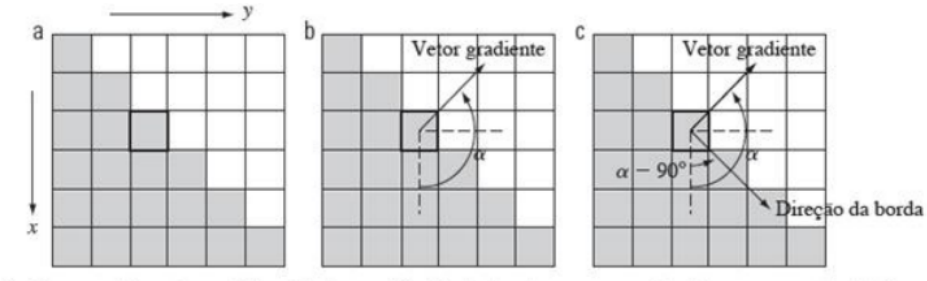
2.3.1 Operadores de Gradiente

A detecção de bordas é parte principal da etapa de segmentação de imagens. As mudanças significativas dos tons de cinza em uma imagem revela fortes evidências de presença de borda. Segundo Pedrini e Schwartz (2008), essas mudanças podem ser descritas por meio do conceito de derivadas. Como uma imagem depende de duas coordenadas espaciais, as bordas da imagem podem ser expressas por derivadas parciais. O operador de gradiente é comumente utilizado em diferenciação de imagens, uma vez que esse vetor indica os locais nos quais os níveis de cinza sofrem maior variação.

A figura 16 mostra uma seção ampliada de uma imagem contendo um segmento de borda reto. Cada quadrado corresponde a um pixel e o objetivo é obter a intensidade e a direção da borda no ponto destacado com um quadrado em negrito. Os pixels em cinza possuem o valor 0 e os pixels brancos 1. Como mostrado na equação 2.5 e 2.6, para calcular as derivadas nas direções x e y utilizando uma vizinhança 3 x 3 centrada sobre um ponto, basta subtrarir o conjunto de pixels localizados na linha superior dessa vizinhança dos pixels localizados na linha inferior encontramos a derivada parcial na direção x. Da mesma forma, subtrair os pixels na coluna esquerda dos pixels na coluna da direita encontramos a derivada parcial na direção y (GONZALES; WOODS, 2010).

A figura 16 mostra também que a borda de um ponto é ortogonal ao sentido do vetor gradiente naquele ponto. a direção do gradiente é sempre perpendicular à direção tangente da borda.

Figura 16 – Usando o gradiente para determinar a intensidade e a direção da borda em um ponto. A borda é perpendicular à direção do vetor gradiente no ponto onde o gradiente é computado. Cada quadrado da figura representa um pixel.



Fonte: (GONZALES; WOODS, 2010)

O vetor gradiente $\nabla f(x, y)$ de uma imagem na posição (x, y) pode ser calculado pelas derivadas parciais como mostrado na equação 2.4.

$$\nabla f(x, y) = \frac{\partial f(x, y)}{\partial x}\mathbf{i} + \frac{\partial f(x, y)}{\partial y}\mathbf{j} \tag{2.4}$$

Em que **i** e **j** são vetores unitários nas direções x e y, respectivamente. Dessa forma, uma rápida variação de $f(x, y)$ ao longo da direção x e lenta ao longo da direção y indica a presença de uma borda praticamente vertical.

A identificação de pontos de borda baseada no operador de gradiente pode ser definida como sendo a magnitude do vetor gradiente. Esse vetor tem a importante propriedade geométrica de apontar no sentido da maior taxa de variação de f no local (x, y). Esta, é definida como a distância entre dois pontos um pertencente à região interna e o outro a região externa, abordados como as derivadas parciais, tendo como resultado a direção do vetor gradiente mostrando assim, um possível ponto de borda. A determinação desses pontos de borda é descrito pelo algoritmo mostrado na figura 17 que utiliza uma imagem de entrada **f** com dimensões M x N pixels e um limiar T. Os algoritmos utilizados para a detecção de borda nesse trabalho são baseados no cálculo do gradiente.

Figura 17 – Algoritmo para determinação de pontos de borda em uma imagem.

1: entradas: uma imagem de entrada **f** com dimensões $M \times N$ pixels e um limiar T.
2: **for** $x = 0$ até $M - 1$ **do**
3: **for** $y = 0$ até $N - 1$ **do**
4: // calcular a magnitude do gradiente $\nabla f(x, y)$
5: $\nabla f(x, y) = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$
6: // efetuar a limiarização
7: **if** $\nabla f(x, y) > T$ **then**
8: (x, y) é um ponto da borda
9: **end if**
10: **end for**
11: **end for**

Fonte: (PEDRINI; SCHWARTZ, 2008)

É importante ressaltar que, como estamos lidando com quantidades digitais, uma aproximação digital das derivadas parciais em uma vizinhança sobre um ponto é necessária. Portanto, segundo Gonzales e Woods (2010), podemos definir as derivadas parciais como sendo:

$$\frac{\partial f(x,y)}{\partial x} = f(x+1, y) - f(x, y) \tag{2.5}$$

$$\frac{\partial f(x,y)}{\partial y} = f(x, y+1) - f(x, y) \tag{2.6}$$

2.3.2 Operador Sobel

O operador de Sobel, também nomeado de filtro de Sobel, é uma operação usada para realçar contornos em imagens. Esse filtro não linear realça linhas verticais e horizontais mais escuras que o fundo, sem realçar pontos isolados. Além dessa característica, ele possibilita realçar as arestas independente da direção: a filtragem pode ser realizada tanto vertical quanto horizontalmente. As regiões destacadas por esse procedimento resultam em bordas mais "grossas"comparadas ao resultado obtido com outras técnicas. Por isso, o operador de Sobel não é tão utilizado para realçar bordas em imagens, sendo mais comum usá-lo para detecção de bordas. Mesmo assim, esse operador continua sendo de grande importância, pois

outros filtros para realçar imagens utilizam-no como base. A figura 18 mostra a aplicação do filtro de Sobel aplicado no eixo x e no eixo y.

Figura 18 – Aplicação do filtro de Sobel para o eixo x e eixo y.

Fonte: Elaborado pela autora

2.3.3 Operador Laplaciano

O operador laplaciano é um dos mais populares e usados para realçar bordas. Assim como o operador de Sobel, o laplaciano também é um filtro espacial. Ele possui uma máscara de ordem 3, que percorre toda a imagem alterando o pixel-alvo pela média ponderada dos pixels vizinhos, e depois eleva ao quadrado o valor obtido. A imagem 19 mostra a aplicação do filtro de Laplace.

Figura 19 – Aplicação do filtro de Laplace.

Fonte: Elaborado pela autora

Para solucionar o problema com ruídos, um filtro passa-baixas, como o gaussiano, pode ser aplicado antes de a imagem ser submetida ao filtro laplaciano. Esse procedimento é tão eficaz que um filtro conhecido como laplaciano de gaussiano foi implementado a fim de aprimorar essa técnica conjunta. O filtro laplaciano de gaussiano unifica essa ideia aplicando uma única máscara e reduzindo consideravelmente o número de operações.

2.3.4 Detector de bordas de Canny

O detector de bordas de Canny é um dos mais eficientes algoritmos para detectar bordas em imagens. Esse método foi desenvolvido por John Canny (1986) que propôs inicialmente suavizar a imagem por um filtro Gaussiano e em seguida, a magnitude e a direção do gradiente são calculadas utilizando aproximações baseadas em diferenças finitas para as derivadas parciais. Canny definiu que um bom detector deveria respeitar três características principais:

1. O algoritmo deve ser capaz de identificar todas as bordas possíveis;
2. Todas as bordas detectadas devem estar próximas das bordas originais da imagem;
3. Bordas falsas não podem ser criadas, ou seja, cada borda deve ser definida uma única vez.

Segundo Pedrini e Schwartz (2008), após o cálculo do gradiente, a borda é localizada tomando-se apenas os pontos cuja magnitude seja localmente máxima. Entretanto, a borda pode ainda conter certos fragmentos causados pela presença de ruído ou textura fina. Uma solução para remover esses fragmentos é utilizar um limiar. Entretanto, a escolha desse limiar é uma tarefa complexa, podendo acarretar em bordas falsas caso o limiar seja muito baixo, ou em perda de fragmentos da borda caso o limiar seja muito alto.

Para solucionar esse problema, o operador de Canny utiliza dois limiares diferentes, T_1 e T_2 com $T_1 > T_2$. Na proposta do algoritmo, Canny sugere que a proporção desse limiar seja de dois para um ou três para um. A figura 20 ilustra um exemplo da aplicação do filtro de Canny em uma imagem após o tratamento em escala de cinza. Foi utilizado o valor de $T_1 = 100$ e $T_2 = 200$ seguindo a recomendação de proporcionalidade de dois para um, obteve-se o melhor resultado se comparado com outros valores testados. O resultado final com o filtro de Canny é dado pela terceira imagem à direita.

Figura 20 – Aplicação do filtro de Canny com limiar mínimo de 100 e máximo de 200.

Fonte: Elaborado pela autora

2.3.5 Segmentação por binarização adaptativa

A segmentação por binarização também é conhecida como aplicação de limiar de intensidade. A separação do objeto de interesse por meio desse método ocorre pela definição

do valor de um limiar. Quando uma imagem é submetida a esse procedimento, os pixels representados por valores maiores que o limiar são estabelecidos como o objeto de interesse, sendo redefinidos para a cor preta ou branca. Do contrário, os pixels representados por valores menores que o limiar são estabelecidos como o segundo plano, sendo redefinidos para a cor oposta à do objeto de interesse. A segmentação por binarização resulta em uma imagem binária, na qual geralmente o objeto de interesse é representado pela cor branca e o segundo plano pela preta.

A figura 21 ilustra a aplicação do filtro binarizado após aplicação do filtro de mediana com janela deslizante de ordem 7, e limiar adaptativo como sendo o Gaussiano da biblioteca OpenCV. A imagem à esquerda apresenta o arquivo original carregado antes de ser convertido para tons de cinza, e a mais a direita, a imagem após sofrer o procedimento de binarização, com os contornos da igreja segmentados.

Figura 21 – Aplicação da segmentação por binarização com limiar adaptativo sendo o Gaussiano da biblioteca OpenCV.

Fonte: Elaborado pela autora

2.3.6 Transformada de Hough

Para compreendermos melhor a transformada de Hough, é necessário rever alguns conceitos algébricos referentes a equação da reta. A equação da reta pode ser definida como

$$y = mx + b \tag{2.7}$$

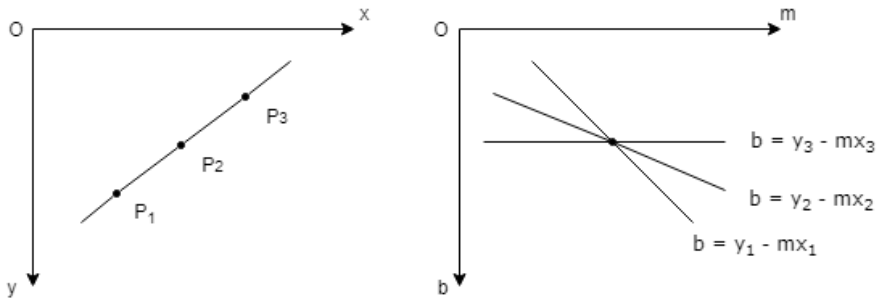
em que m é o coeficiente angular e b é o coeficiente linear. Segundo Pedrini e Schwartz (2008), para valores diferentes de m e b, infinitas retas passam por um ponto $p_1(x_1, y_1)$, todas elas satisfazendo a equação $y_1 = mx_1 + b$. Analogamente, infinitas retas que passam por um ponto $p_2(x_2, y_2)$ podem ser expressas pela equação $y_2 = mx_2 + b$.

Reorganizando a equação 2.7 de tal forma que m e b sejam os parâmetros e x e y sejam as constantes, tem-se que $b = y - mx$. O plano mb é conhecido como *espaço de parâmetros*. Todas as retas que passam pelo ponto p_1 são representadas no espaço de parâmetros pela equação $b = y_1 - mx_1$. Do mesmo modo, a equação $b = y_2 - mx_2$ representa o conjunto de todas as retas que passam pelo ponto p_2 no plano da imagem.

O ponto (m, b) no espaço de parâmetros é comum a essas duas retas associadas aos pontos p_1 e p_2. De fato, todos os pontos que são colineares no plano da imagem se

interceptam em um mesmo ponto no espaço de parâmetros. A figura 22 ilustra o mapeamento entre o plano da imagem e o espaço de parâmetro. As coordenadas no espaço (m, b) do ponto de intersecção fornecem os parâmetros da reta no espaço (x, y) que contém esses pontos.

Figura 22 – Princípio da transformada de Hough. (a) plano da imagem; (b) espaço de parâmetros.



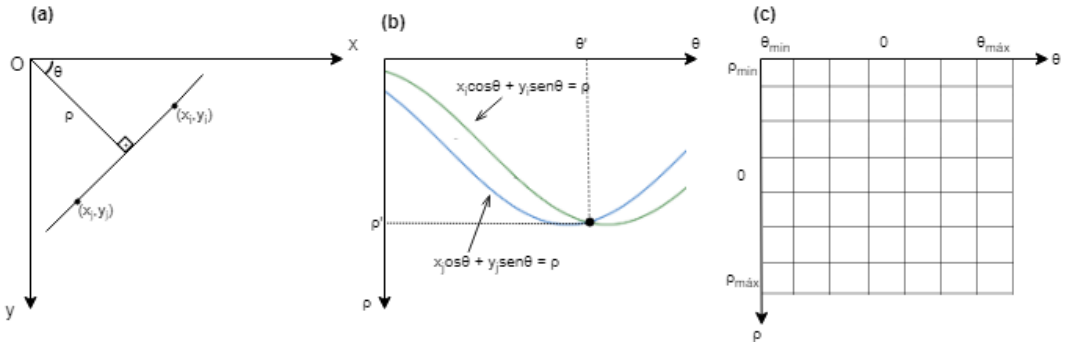
Fonte: (PEDRINI; SCHWARTZ, 2008)

Esse conceito forma a base da transformada de Hough para a detecção de retas. Uma dificuldade prática com essa abordagem, porém, é que m se aproxima do infinito conforme a reta se aproxima da direção vertical. Uma maneira de contornar essa dificuldade é utilizar a representação normal de uma reta em coordenadas polares:

$$xcos\theta + ysen\theta = \rho \tag{2.8}$$

A atratividade computacional da transformada de Hough surge da subdivisão do espaço de parâmetros $\rho\theta$ nas chamadas "células acumuladoras" (GONZALES; WOODS, 2010). Como mostrado na figura 23(a) uma reta no plano xy é transformada utilizando a equação 2.8 em curvas senoidais da figura 23(b) e é contabilizado a quantidade de vezes que um ponto xy é colinear, isto é, está na mesma célula $A(i, j)$ da figura 23(c). Assim, no final do processo um valor em cada célula em $A(i, j)$ representa pontos colineares das curvas senoidais e consequentemente uma potencial reta no plano xy.

Figura 23 – (a) (ρ, θ) Parametrização do plano xy. (b) Curvas senoidais no plano $\rho\theta$; (c) Divisão do plano $\rho\theta$ em células acumuladoras.



Fonte: (GONZALES; WOODS, 2010)

Uma dúvida que pode surgir ao estudar a transformada de Hough é ”como saber que o ponto considerado em 2.8 de fato pertence a reta do plano xy?” Ora, se manipularmos a equação 2.8 teremos:

$$y = \frac{-cos(\theta)}{sen(\theta)} * x + \frac{\rho}{sen(\theta)} \tag{2.9}$$

Assim, como mostrado na figura 23(b) ρ e θ serão constantes uma vez que diversos valores de x e y resultará sempre nos mesmos pontos colineares. Dessa forma, o coeficiente angular e o coeficiente linear da equação 2.9 serão sempre os mesmos levando assim a uma única reta para n pontos estabelecidos de x e y.

2.3.6.1 Transformada Probabilística de Hough

A transformada probabilística de Hough é uma forma simplificada de obtenção dos parâmetros da reta. Introduzida em 1990 por N. Kiryati, Y. Eldar e A. M. Bruckshtein, a transformada probabilística de Hough afirma que para descobrir objetos é suficiente computar a transformada de Hough só de uma proporção α $(0\% < \alpha \leq 100\%)$ de pixels na imagem. Esses pixels são randomicamente escolhidos de acordo com uma função de densidade de probabilidade uniforme, definida em cima da imagem. Isto é equivalente a computar a transformada padrão de uma amostra da imagem (JAMUNDÁ, 2019). A biblioteca OpenCV retorna dois pontos pertencentes a reta de forma simplificada e rápida, assim, para a implementação deste trabalho optou-se por utilizar a transformada probabilística de Hough.

2.4 Representação e descrição

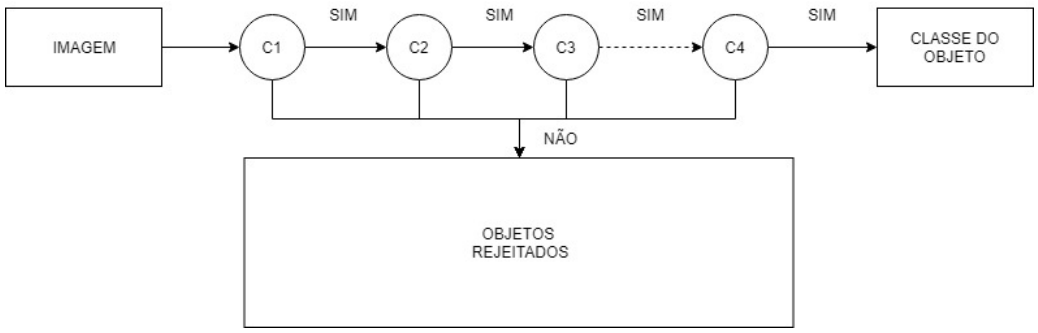
2.4.1 Algoritmo HaarCascade

Os detectores baseados em cascade (cascata) são chamados assim pois treinam uma árvore de decisão em que cada nível analisa um conjunto de atributos diferentes e avalia se esses atributos representam ou não o objeto de interesse.

Cada nível dessa árvore é chamado de estágio. Cada estágio é composto por um mais classificadores fracos (*weak classifiers*). Esses classificadores fracos são, em geral, treinados com algoritmos baseados em métodos de *boosting*, como o *Adaboost*, até que eles atinjam uma taxa mínima de verdadeiros positivos e, ao mesmo tempo, uma taxa máxima de falsos positivos. Um detalhe importante é que os exemplos incorretamente classificados são automaticamente escolhidos para a próxima iteração do algoritmos, só que com pesos maiores. Ou seja, na próxima iteração, o algoritmo deve dar preferência a acertar os exemplos que ele errou anteriormente. A figura 24 ilustra o funcionamento.

Na prática, o sistema usa o esquema de janela deslizante sobre a imagem. Cada janela correspondente é classificado pelo *cascade* como positivo ou negativo. Outro detalhe importante é: para que um objeto seja considerado positivo, ele deve passar por todos os estágios. Isso é importante para permitir o algoritmo economizar tempo, já que se um único estágio dizer que o objeto pertence a classe negativa, o algoritmo já passa para próxima janela

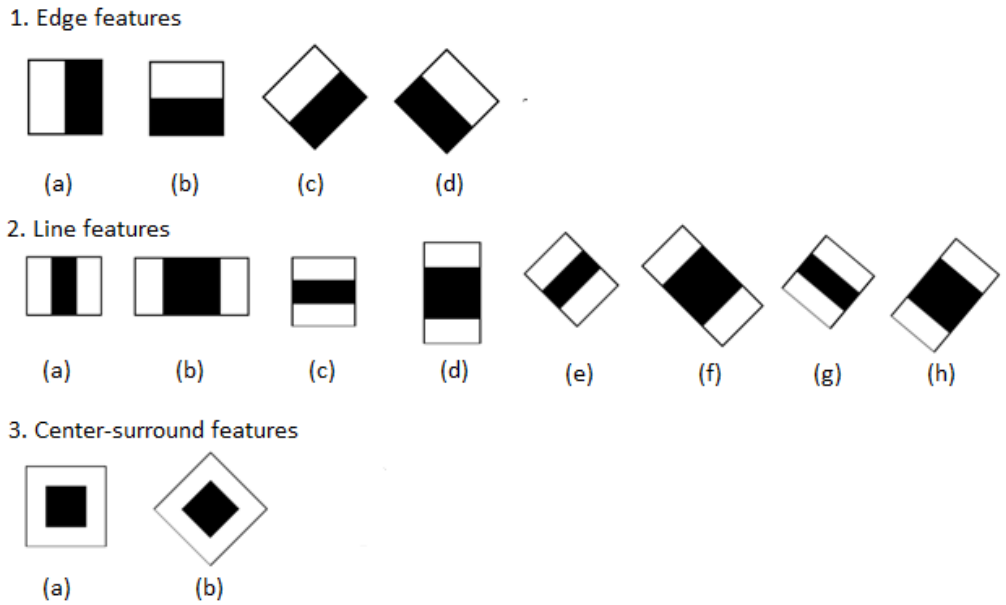
Figura 24 – Funcionamento em cascata do classificador.



Fonte: Adaptado (BASTOS-FILHO CARMELO JA E SILVA, 2014)

e recomeça o processo. Todo esse processo é feito automaticamente pelo *framework* utilizado. É necessário apenas se preocupar em treinar o detector. A OpenCV disponibiliza dois tipos de *features* para treinamento de *cascades*: *Haar* e *LBP* (Local Binary Pattern). Atributos *Haar* são extraídos subtraindo diferentes pixels da imagem de acordo com as "máscaras" mostradas na figura 25. Este foi o escolhido para o desenvolvimento deste trabalho.

Figura 25 – Filtros de Haar: Os pixels das regiões brancas são subtraídos dos pixels da região preta correspondente. Por exemplo, a máscara 1(a) apenas subtrai o valor de um pixel do seu vizinho à direita. A máscara 2(c) subtrai o pixel central dos pixels logo acima e abaixo. A máscara 3(a) subtrai um pixel central dos seus 8-vizinhos.



Fonte: (GRANATYR, 2018)

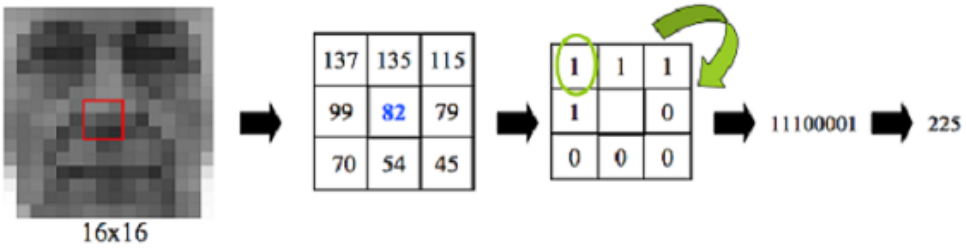
A figura 26 mostra um exemplo da janela deslizante sobre uma imagem. Os *Haar* utilizam os filtros mostrados na figura 25 para percorrer toda a imagem positiva procurando o padrão que se encaixe

Figura 26 – Exemplos de atributos Haar calculadas sobre uma imagem.

Fonte: (GRANATYR, 2018)

Por outro lado, atributos LBP são calculados da seguinte forma: para cada pixel da imagem, compare o pixel central com sua vizinhança de 8-vizinhos, binarizando-os. Depois, transforma o código binário correspondente em um valor decimal. A figura 27 exemplifica esse processo.

Figura 27 – Exemplo de código LBP calculado para um pixel de uma imagem 16x16.

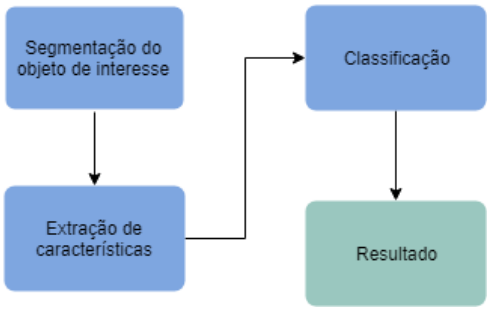


Fonte: (GRANATYR, 2018)

2.5 Reconhecimento e interpretação

A capacidade do ser humano de reconhecer padrões e classificar objetos sempre impressionou cientistas dos mais diversificados campos de estudo, principalmente os que se dedicam a desenvolver tecnologias capazes de imitar a natureza humana. O reconhecimento de padrões é a técnica científica utilizada para descrever padrões de objetos a fim de classificá-los. Nos sistemas baseados em Visão Computacional, as técnicas de reconhecimento de padrões são essenciais, pois possibilitam a classificação automática de um objeto de interesse (BARELLI, 2018). Assim, essas técnicas permitem que computadores enxerguem o mundo à nossa volta, reconhecendo placas, peças, caracteres, faces humanas, objetos e outros. O reconhecimento de objetos detectados por nós humanos é possível graças às características peculiares de cada um. Semelhantemente, para realizar essa tarefa, os sistemas executam basicamente três procedimentos apresentados na figura 28.

Figura 28 – Sistemas de Reconhecimento de Padrões



Fonte: Adaptado - (BARELLI, 2018)

Segundo Barelli (2018), para a classificação, os algoritmos mais populares em Visão Computacional são os Bayesianos, o K-NN (K-Nearest Neighbor), a Lógica Nebulos, as RNA (Redes Neurais Artificiais). Os classificadores podem ser avaliados quanto à velocidade de execução e à capacidade de fazer classificações corretas e outros parâmetros. Para que os classificadores consigam detectar os objetos em classes, sem que tenham sido explicitamente programados, eles precisam aprender alguns critérios para executar essa tarefa. Existem dois métodos de aprendizagem de máquina bastante usados: aprendizagem supervisionada e a não supervisionada.

2.5.1 Aprendizagem supervisionada

Nesse modelo de aprendizagem, o classificador é treinado para reconhecer padrões a partir de objetos já conhecidos. Um conjunto de caracteristicas de objetos e suas respectivas classes são apresentados ao classificador. A partir desses dados, ele é treinado para identificar automaticamente padrões nessas informações, tornando-se capaz de classificar novos objetos. Justamente por necessitar de um agente externo, responsável por apresentar ao classificador dados para o treinamento, esse método é conhecido como supervisionado. Geralmente, os dados usados na etapa de treinamento são definidos por algum especialista no problema. A figura a seguir apresenta um processo de aprendizagem supervisionada. Um conjunto de objetos está representado em um espaço de características bidimensional. Todos os objetos coloridos possuem classes definidas e indicadas na legenda; eles representam a base de conhecimento do classificador. A figura 29 à esquerda exibe um elemento de classe desconhecida, representado na cor preta. Os classificadores baseados nesse tipo de aprendizagem são capazes de classificar esse elemento a partir das informações sobre as características dos elementos conhecidos (já classificados).

2.5.2 AdaBoost

O *AdaBoost* (do inglês *Adaptive Boosting*) é um algoritmo de aprendizado supervisionado do tipo *boost*. O *AdaBoost* combina um conjunto de funções simples de classificação, denominadas classificadores fracos para formar um classificador forte.

Segundo Bastos-Filho Carmelo JA e Silva (2014) um classificador forte é composto de

Figura 29 – Aprendizagem supervisionada

Fonte: (BARELLI, 2018)

um conjunto de classificadores fracos, associados a pesos que classificam de forma precisa dois conjuntos de dados, onde as características com pesos maiores são mais significativas para a classificação de exemplos definidos como parte de um certo conjunto. Na Equação 2.10 é formalizada a criação de um classificador forte através de um algoritmo de *Boosting*.

$$H(x) = \alpha_1 h_1 + \alpha_2 h_2 + ... + \alpha_n h_n \tag{2.10}$$

onde $H(x)$ é um classificador forte, α_i é o peso associado ao classificador h_i.

Dado uma base de dados de entrada, a função do *AdaBoost* é encontrar o conjunto de características que irão formar o classificador forte para uma melhor classificação do conjunto de entrada.

É importante ressaltar que, no presente trabalho, o processo de detecção e extração das características da região de interesse foi utilizado o *AdaBoost* juntamente com o classificador *HaarCascade*. Obtendo assim, o aprendizado supervisionado por meio do *AdaBoost* e a procura da característica requerida por meio das *features* delimitadas pelo *HaarCascade*.

2.6 Biblioteca OpenCV

Open Source Computer Vision Library (OpenCV) é uma biblioteca desenvolvida pela Intel no ano 2000. É uma biblioteca multiplataforma de código aberto para desenvolvimento na área de visão computacional e aprendizado de máquinas. Possui módulos de Processamento de Imagens e Vídeo de entrada e saída, Estrutura de dados, Álgebra Linear, GUI. Todas as operações citadas neste trabalho, filtros de imagem, detecção, reconhecimento existem na biblioteca.

A OpenCV foi escrita nativamente em C++ e hoje possui interface com várias linguagens, como C, Python, Java, Visual Basic, Ruby, dentre outras. A biblioteca possui mais de 2.500 algoritmos otimizados para detecção e reconhecimento de faces, identificação de objetos, extração de modelos de objetos em 3D, nuvens de pontos utilizando câmeras estéreo,

etc. A OpenCV conta com mais de 47 mil pessoas como usuários na comunidade e número estimado de downloads superior a 6 milhões.

2.7 Plataforma Anaconda

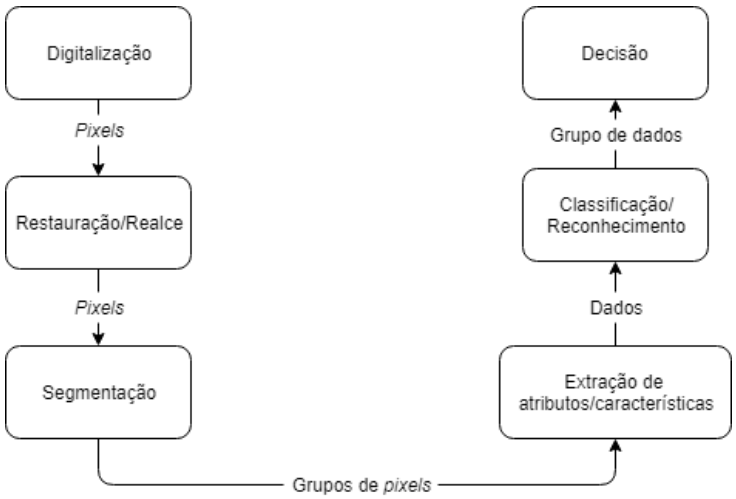
Conda é um sistema de gerenciamento para pacotes *open source* e ambientes de desenvolvimento que executam em Windows, macOS e Linux. Conda instala, executa e atualiza os pacotes e suas dependências necessárias para integrar a aplicação local com as ferramentas utilizadas. Ele foi criado para programas em Python, mas se expandiu e também está disponível para as principais linguagens que implementam *machine learning* como R, Lua, Scala, Ruby e também Java, C e C++.

3 Trabalhos Relacionados

“Considere e analise a situação do inimigo e onde ele deseja batalhar, você pode ter uma compreensão clara das suas chances de sucesso.”
Sun Tzu

Na literatura, foram encontrados diversos trabalhos relacionados à soluções desenvolvidas para atribuir ao computador a capacidade de processar e interpretar imagens. Grande parte dos trabalhos relacionados à visão computacional seguem as etapas proposta por Azevedo, Conci e Leta (2010) mostradas na figura 30. Em seguida são listados os principais trabalhos relacionados.

Figura 30 – Etapas de um sistema de visão computacional genérico.



Fonte: Adaptado (AZEVEDO; CONCI; LETA, 2010)

Queiroz (2015) propõe um método para classificar amostras obtidas a partir de um processo metalúrgico utilizando técnicas de processamento digital de imagens. Para isso, ela utiliza algoritmos de limiarização e histograma para o pré-processamento e para a segmentação utiliza um algoritmo que implementa o método de Canny para aplicação de ltros e detecção de bordas na imagem. Esse método apresentou um desempenho melhor se comparado com método de Sobel e ltros de média, Gaussiano e bilateral. Ainda para a etapa de segmentação, é utilizado limiar adaptativo e transformada de Hough.

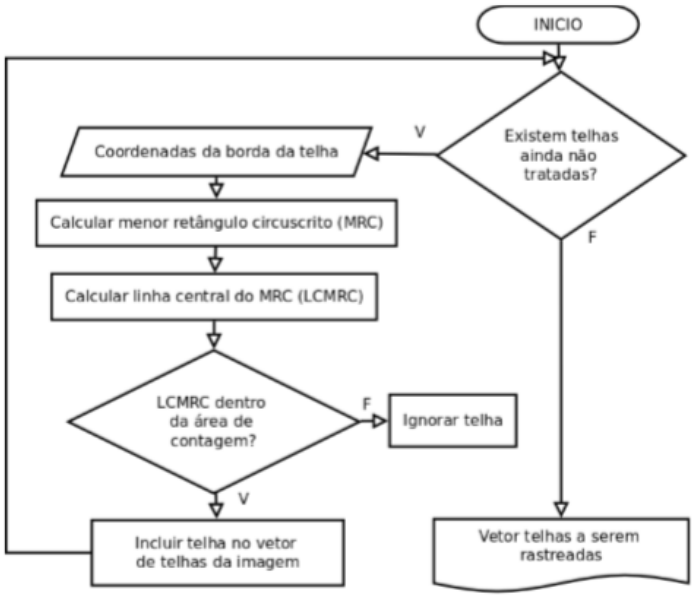
Martins (2010) desenvolve um mecanismo de calibração da câmera para análise e captura das imagens retiradas da área operacional e utiliza a transformada de Hough, na segmentação e detecção de retas e círculos. Para a extração de características é utilizado métodos de textura baseada em estatísticas como o Análise de Componentes Principais (ACP) e Descritores de Haralick. O método APC consiste em encontrar um novo sistema de coordenadas (a partir de transformações de matrizes em um espaço ortogonal) para representar o conjunto

de dados que está sendo trabalhado. O método de Haralick consiste em extrair as características das imagens a partir dos níveis de cinza apresentados no histograma da imagem. Para a classificação, é utilizado redes neurais implementadas com o algoritmo de múltiplas camadas (MLP) e redes neurais auto-organizáveis (SOM).

BENTO (2016) propõe um método de identificação não supervisionada de propagação de trincas por meio de técnicas de processamento digital de imagens. Para a etapa de pré-processamento foi utilizado técnicas de limiarização e operações matemáticas morfológicas. Para a etapa de segmentação foi utilizado algoritmo de Sobel para remoção de ruídos e eliminação de fundo; para detecção de bordas algoritmo de Canny; para detecção de retas a bentonsformada de Hough.

SILVA (2014) desenvolve uma ferramenta para contagem automática de telhas utilizando dispositivo mobile. Para isso, ele utiliza métodos exatos para a detecção de objetos. Esse método se baseia em delimitar características de objetos como comprimento, perímetro, raio, distância ao centro de massa, altura, rotação, etc. Podendo ser utilizada uma ou mais característica combinada para definir se determinado objeto pertence ou não a uma classe. Para a etapa de pré-processamento ele utiliza representação em escala de cinzas, limiarização e operadores morfológicos (erosão e dilatação); na etapa de segmentação foi utilizado um algoritmo de detecção de bordas externas às telhas; e por fim aplicação do chamado filtro de Kalman para predição e correção da posição do objeto de interesse; algoritmo CONDENSATION para o rastreamento de objetos em cenas ambíguas. Para a etapa de classificação utiliza redes neurais que é treinada de forma supervisionada através do algoritmo backpropagation. O fluxo do sistema desenvolvido é mostrado na figura 31.

Figura 31 – Fluxo de processos do sistema contador de telhas.



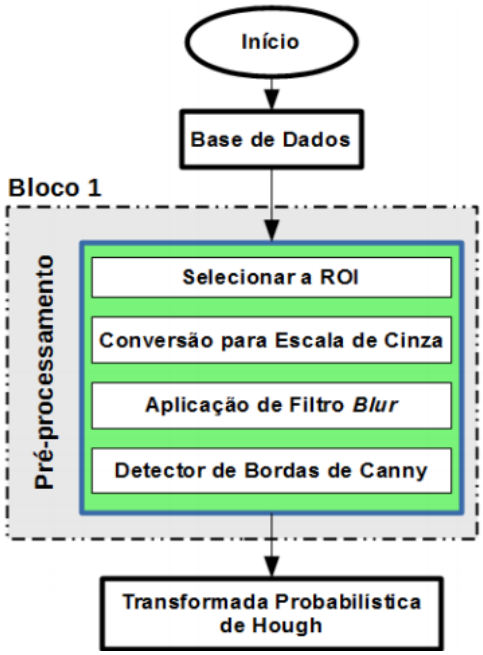
Fonte: Bruno Ramon de Oliveira e Silva, 2014

Salis e Pereira (2007) propõe no trabalho Contagem Automática de Tarugos a criação de um sistema que monitore a formação dos pacotes de tarugos produzidos por uma siderúr-

gica utilizando visão computacional. Foi desenvolvido um sistema de obtenção das imagens utilizando a rede já instaurada na empresa em estudo e utilizando duas metodologias o autor extrai as características de interesse para cada imagem. Para eliminar a distorção causada pela iluminação dos tarugos ainda quentes, na primeira metodologia foi aplicado o filtro *topHat* que consiste na subtração de uma imagem aberta da sua original. Em seguida, é feito a binarização da imagem e operações morfológicas (erosão) para remoção de ruídos e filtros de média. Logo após, sucessivas operações de erosão e dilatação são realizadas separar os elementos de interesse e deixá-los visíveis. Na segunda metodologia foi realizada uma adaptação e melhoria para detecção dos tarugos que se encontram com temperaturas mais baixas e portanto com coloração diferenciada. Para isso, sucessivas operações morfológicas aplicadas em ordem diferentes das aplicadas na metodologia 1 foram executadas além de filtros de média. Assim, o resultado a taxa pontual de acerto apresentada por este trabalho é de 96% com a metodologia 1 e 100% para a metodologia 2.

Júnior (2016) propõe identificar e classificar a sinalização em autovias com o objetivo de auxiliar o motorista no trânsito e assim reduzir o índice de acidentes causados. Utilizando técnicas morfológicas, operadores de segmentação como filtro gaussiano e Canny ele consegue reduzir ruídos, identificar área de interesse a ser trabalhada e em seguida aplica a transformada probabilística de Hough para identificar as faixas de sinalização e calcular a posição do veículo na faixa e os obstáculos ou outros veículos ao redor. O processo desenvolvido para a primeira etapa utilizado neste trabalho é ilustrado na figura 32

Figura 32 – Fluxograma proposto para a etapa de pré-processamento



Fonte: (JÚNIOR, 2016)

4 Materiais e Métodos

“Administrar um exército grande é, em princípio, igual a administrar um pequeno: é uma questão de organização.”
Sun Tzu

Os materiais utilizados como referência para o estudo e entendimento de técnicas de processamento digital de imagens e visão computacional incluem a leitura de artigos relacionados ao tema, livros, tutoriais e materiais disponíveis na internet, cursos lecionados na plataforma digital, demais trabalhos de conclusão de curso, além de teses de mestrado e doutorado.

A principal biblioteca de implementação para o desenvolvimento do algoritmo para reconhecimento é a OpenCV versão 4.0.1 utilizando a linguagem de programação Python versão 3.7.3 no ambiente de desenvolvimento PyCharm versão 2018.2.2, a plataforma Anaconda para gerenciamento do OpenCV e instalação das bibliotecas que foram utilizadas.

A implementação foi realizada em um notebook DELL *Special Edition* com as seguintes especificações técnicas:

- Processador Intel Core i7 5500U CPU 2.40Ghz;
- Memória RAM 8,00GB;
- Sistema Operacional Windows 10 x64;
- Disco Rígido 1TB.

Para a etapa de aquisição de imagens as especificações técnicas do celular Zenfone 4 são:

- Processador 2.2GHz
- Memória RAM 4GB
- Câmera Traseira Principal 12MP com abertura $f/1.8$

As imagens obtidas seguiram a seguinte configuração da câmera:

- Tempo de exposição $1/30$s para classe TC 3040E e 3040ENPRO
- Tempo de exposição $1/24$s para classe DNEX 1328
- Para a classe Classe TC 3030E ISO de 82 139 163 174
- Para a classe 3040ENPRO ISO de 135 158 168
- Para a classe DNEX 1328 ISO faixas de 230 238 245 253 278 296 315 325

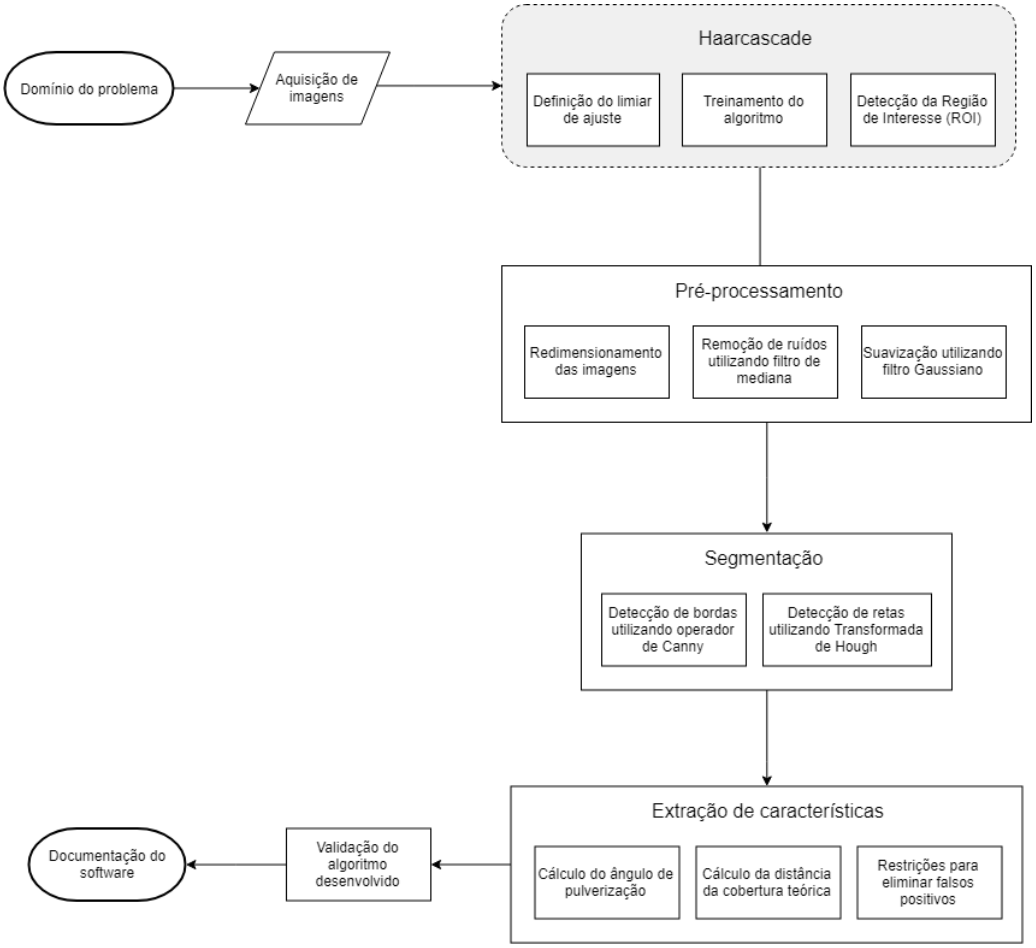
- Sem flash
- Distância focal da lente da câmera de 3.94mm

As principais atividades realizadas durante o projeto foram:

1. Estudo sobre detecção de objetos de interesse utilizando *machine learning* e técnicas para Processamento Digital de Imagens;
2. Aquisição das imagens em bancada de teste;
3. Definição das características importantes das imagens obtidas a serem avaliadas e detectadas pelo *Haarcascade*;
4. Treinamento do algoritmo com a imagem de interesse;
5. Extração das características;
6. Elaboração de testes para validação do algoritmo com as imagens do estudo de caso.

Azevedo, Conci e Leta (2010) subdivide os sistemas que utilizam processamento digital de imagens em passos fundamentais: domínio do problema, obtenção da imagem, restauração/realce (pré-processamento), segmentação, extração de características, classificação e decisão. Com essa divisão proposta e com os passos acima delimitados, tem-se os passos definidos pelo fluxograma apresentado na figura 33.

Figura 33 – Metodologia utilizada para implementação da proposta.



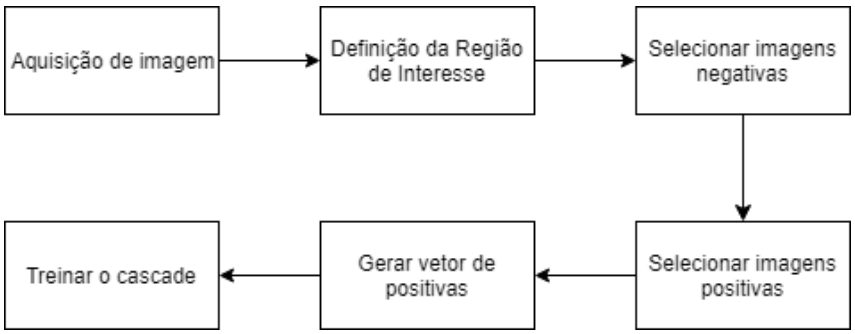
Fonte: Elaborado pela autora

5 Aplicação do algoritmo Haarcascade

“É difícil, porque se trata de transformar um tortuoso caminho em uma estrada reta, transformar uma desvantagem em vantagem”.
Sun Tzu

Para o desenvolvimento do algoritmo *haarcascade* foi necessário a etapa de aquisição de imagens, definição a região de interesse a ser detectada; escolha das imagens negativas, isto é, todas as imagens que não contêm o objeto de interesse; escolha das imagens positivas que são aquelas que contêm o objeto de interesse; geração das máscaras *Haar Features* e por fim a etapa de treinamento do algoritmo. É na etapa de treinamento que o algoritmo consegue armazenar as informações da imagem correta a ser detectada das imagens a serem descartadas. A figura 34 mostra em um fluxo esse funcionamento do *haarcascade*.

Figura 34 – Etapas para o funcionamento do haarcascade



Fonte: Elaborado pela autora

5.1 Aquisição de imagens

As imagens foram obtidas a partir da bancada de teste em uma siderurgia como mostra o esquema da figura 35. Uma câmera móvel foi posicionada na frente da bancada de testes com o jato em funcionamento a uma distância x. Mediu-se a distância do bico emissor do jato até a superfície (distância de pulverização y) para ser informada ao método desenvolvido e a pressão da água em que o bico foi submetido. As imagens obtidas formam três classes de objetos como mostrado na tabela 1.

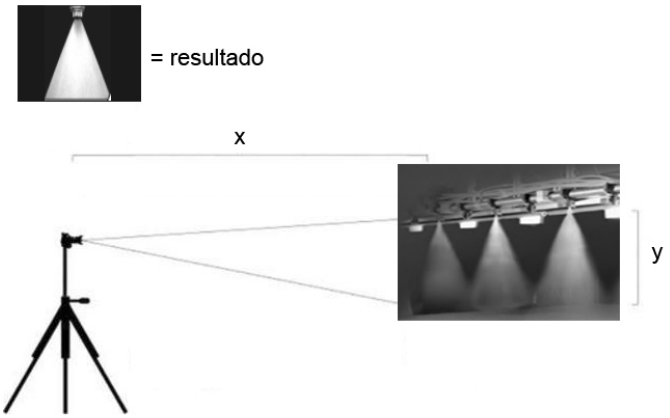
Tabela 1 – Imagens utilizadas para os testes realizados.

Classe	**Pressão**	**Tipo arquivo**	**Fabricante**
DNEX 1328	10Kgf/cm^2	37 imagens vídeo com 14s de exposição	*Kyoritsu*
TC 3040 E	10Kgf/cm^2	18 imagens vídeo com 13s exposição	*Spraying Systems*
TC 3040EN PRO	10Kgf/cm^2	23 imagens vídeo com 16s de exposição	*Spraying Systems*

Fonte: Elaborado pela autora

O esquema mostrado na figura 35 para obtenção das imagens, ilustra que as imagens obtidas são em 2D. Ou seja, para o contexto desse trabalho, não é possível analisar e avaliar os jatos de acordo com a forma que ele atinge a superfície. É válido lembrar que, como mostra no trabalho de Peterson et al. (2015), a avaliação do desempenho dos jatos bem como a área de contato com a superfície é realizada a partir de estudos considerando a pressão, fluido emitido, tempo de emissão do fluido, etc. Sendo assim, este trabalho visa detectar e delimitar o formato do jato para auxiliar no processo de avaliação da qualidade em termos visuais.

Figura 35 – Esquema para obtenção das imagens dos jatos emitidos pelo bico descarepador



Fonte: Elaborado pela autora

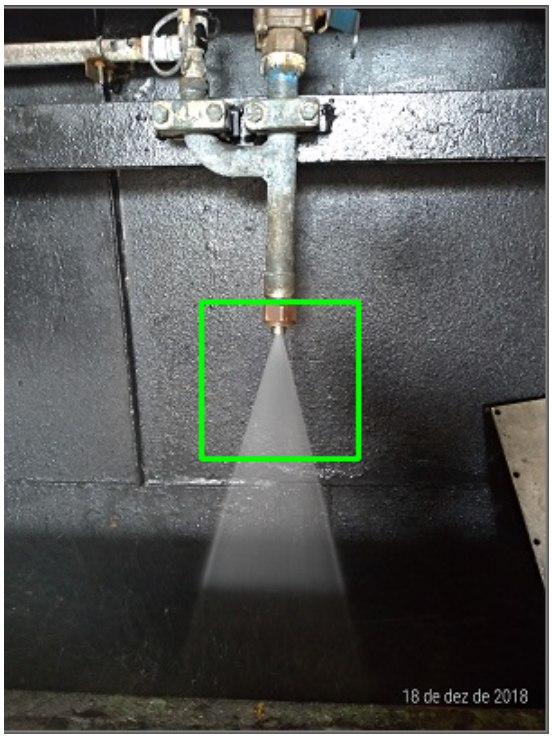
5.2 Definição da região de interesse

As imagens que compõem a base de dados exibem o fundo da bancada de teste, a base suspensa para o bico e algumas irregularidades da chapa traseira da bancada como mostra a figura 36. Essas características da bancada de teste geram ruídos e prejudicam a análise nas etapas de pré-processamento, segmentação e reconhecimento de retas utilizando a transformada de Hough.

Para amenizar esse tipo de problema, optou-se por selecionar uma Região de Interesse, ROI (*Region of Interest*), localizada na parte central da imagem de forma a detectar

apenas o início do jato.

Figura 36 – Imagem do jato em bancada de testes com a ROI delimitada pelo *bounding box* em verde.



Fonte: Elaborado pela autora

5.3 Definição das imagens negativas

Para representar as imagens que não possuem os objetos de interesse, foram utilizadas três mil imagens arbitrárias e distintas. Todas com dimensões 100x100 pixels para não comprometer o desempenho durante o treinamento do algoritmo.

As imagens negativas são enumeradas em um arquivo especial externo (bg.txt). Esse arquivo criado manualmente contêm o nome das imagens negativas e sua extensão. Na etapa de treinamento do haarcascade esse arquivo é referenciado para mapeamento das imagens negativas a serem consideradas.

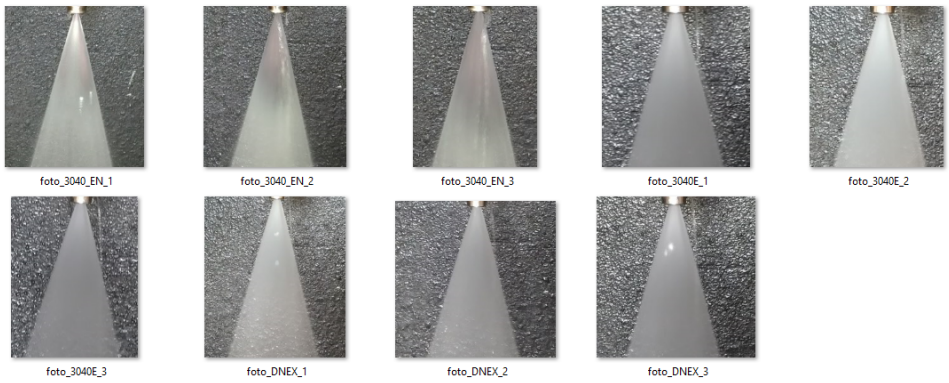
5.4 Definição das imagens positivas

Para as imagens positivas é necessário definir imagens contendo o objeto e a região de interesse desejada. É a partir dessas imagens que o *haarcascade* conseguirá buscar a região de interesse nas imagens da base de dados.

Para o primeiro experimento, foram escolhidas 3 imagens de cada classe das imagens obtida. Para o segundo experimento, em uma tentativa de aumentar o índice de acertos e diminuir os falsos positivos, foram escolhidas 5 imagens de cada classe.

A figura 37 ilustra as imagens positivas que foram submetidas para o primeiro experimento.

Figura 37 – Imagens positivas utilizadas para treinamento do classificador



Fonte: Elaborado pela autora

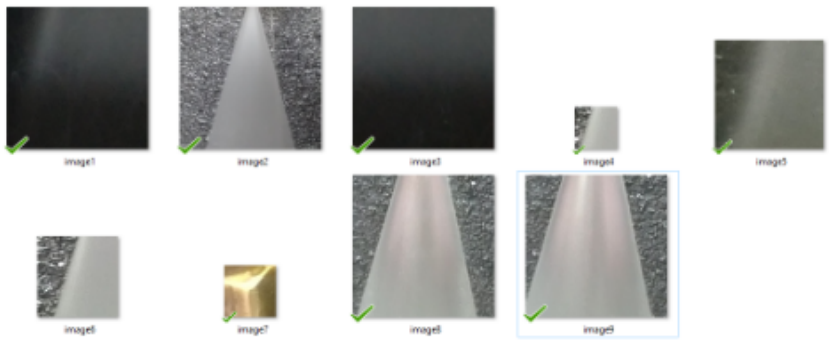
5.4.1 Geração do vetor de positivas

Após definir a ROI, criou-se o vetor de positivas para cada imagem positiva mostrada na figura 37 com a utilização do componente *create_samples* do OpenCV. Esse componente do OpenCV cria um arquivo contendo cada imagem negativa e a posição da imagem positiva sobreposta. Para essa criação, é necessário especificar alguns parâmetros como:

- $maxxangle$: máximo valor de rotação que a imagem positiva terá no eixo x;
- $maxyangle$: máximo valor de rotação que a imagem positiva terá no eixo y;
- $maxzangle$: máximo valor de rotação que a imagem positiva terá no eixo z;
- $bgcolor$: valor da cor em RGB do fundo da imagem que se deseja tornar transparente;
- $bgthresh$: quantidade de tolerância que o parâmetro bgcolor tem para ser transformado em transparente.

Os parâmetros $maxangle$ para os eixos x, y e z foram definidos como 0.5 para terem a mesma inclinação. O ajuste desses parâmetros permite que o algoritmo consiga reconhecer o objeto de interesse mesmo que esteja posicionado a alguns ângulos diferentes. Os primeiros testes foram feitos utilizando o valor de $bgcolor$ em 0 (uma vez que o objetivo é tornar o fundo preto da imagem em transparente) e o valor para $bgthresh$ em 100. Porém, os resultados obtidos nas imagens da região de interesse não foram satisfatórias. Assim, foi preciso diminuir o valor de $bgtresh$. A figura 38 ilustra o resultado do teste realizado para obtenção da ROI.

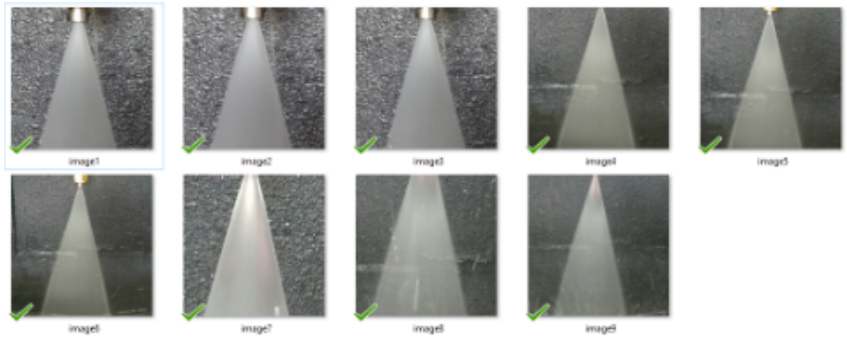
Figura 38 – Imagens da ROI para o parâmetro bgthresh igual a 100



Fonte: Elaborado pela autora

Após alguns testes verificamos que o melhor valor para bgtresh seria de 20. Assim, obtivemos as seguintes regiões de interesse mostradas na figura 39

Figura 39 – Imagens da ROI para o parâmetro bgthresh igual a 20



Fonte: Elaborado pela autora

Após a criação de todos os vetores de imagens positivas sobrepostas nas imagens negativas, um único arquivo compacto foi gerado para que o treinamento fosse realizado considerando todas as especificações contidas em cada imagem trabalhada anteriormente. É válido ressaltar que a dimensão das imagens utilizadas foi de 24x24 pixels. Esse foi o limiar encontrado de melhor custo X benefício. Entre o melhor desempenho de processamento com o equipamento utilizado e o resultado aceitável para o proposto trabalho. Valores maiores que 24 pixels se tornaria muito custoso e com resultados não muito distintos se comparado aos obtidos.

Objetivando resultados melhores para a detecção da região de interesse foram realizados testes utilizando mais imagens positivas para a criação do cascade. Portanto, ao invés de 3 imagens para cada classe das imagens obtidas, o classificador foi criado utilizando 5 imagens de cada classe. Dessa forma, e com o ajuste dos parâmetros *scaleFactor* e *minNeighbors*, o cascade apresentou menos falsos positivos e melhor qualidade na detecção da ROI. Os resultados obtidos são descritos na sessão ”Experimento 2”.

5.5 Treinamento do cascade

Na etapa de treinamento do cascade, utilizou-se o componente *opencv_traincascade* da biblioteca OpenCV. Para isto, especificou-se a quantidade de imagens positivas e negativas a serem criadas. O resultado do treinamento são arquivos em XML contendo os estágios de processamento, um arquivo contendo os parâmetros e um arquivo contendo as especificações do classificador. Este, é utilizado no programa desenvolvido em Python para detectar a região de interesse. A imagem abaixo com o código em XML mostra as *tags* contendo detalhes dos estágios criados, a dimensão das imagens, etc.

Um destaque especial para os parâmetros *minHitRate* e *maxFalseAlarmRate*. O *minHitRate* refere-se à taxa de acerto mínima desejada para cada estágio do classificador. Quanto mais próximo de 1 melhor. De acordo com experimentos, o melhor valor a ser utilizado é 0.998. O *maxFalseAlarmRate* refere-se à taxa máxima de alarme falso desejado. Esse valor é para definir quantas *features* precisam ser adicionadas ao processo. A intenção é fazer com que o algoritmo encontre o máximo de janelas positivas e remova as negativas o mais rápido possível. O valor especificado foi de 0.5 que, de acordo com experimentação e recomendações, foi o melhor valor para remover as janelas negativas mais rapidamente. O tempo para processamento final do treinamento do cascade foi cerca de 15 minutos. Entretanto, para a criação dos vetores de imagens positivas demorou cerca de 3 minutos para cada imagem positiva submetida. Considerando as 15 imagens positivas, A criação do vetor de classificação demorou 45 minutos. Resultando em um tempo total de treinamento de 1 hora.

```
1  <?xml version="1.0"?>
2  <opencv_storage>
3  <cascade>
4    <stageType>BOOST</stageType>
5    <featureType>HAAR</featureType>
6    <height>24</height>
7    <width>24</width>
8    <stageParams>
9      <boostType>GAB</boostType>
10     <minHitRate>9.9500000476837158e-01</minHitRate>
11     <maxFalseAlarm>5.0000000000000000e-01</maxFalseAlarm>
12     <weightTrimRate>9.4999999999999996e-01</weightTrimRate>
13     <maxDepth>1</maxDepth>
14     <maxWeakCount>100</maxWeakCount></stageParams>
15   <featureParams>
16     <maxCatCount>0</maxCatCount>
17     <featSize>1</featSize>
18     <mode>BASIC</mode></featureParams>
19   <stageNum>6</stageNum>
20   <stages>
21     <!-- stage 0 -->
22       <maxWeakCount>1</maxWeakCount>
23       <stageThreshold>8.6046510934829712e-01</stageThreshold>
24       <weakClassifiers>
```

```
25            <internalNodes>
26              0 -1 6 -2.4327360093593597e-02</internalNodes>
27            <leafValues>
28              8.6046510934829712e-01 -1.</leafValues></_></weakClassifiers
                  ></_>
29      <!-- stage 1 -->
30        <maxWeakCount>1</maxWeakCount>
31        <stageThreshold>5.3256702423095703e-01</stageThreshold>
32        <weakClassifiers>
33            <internalNodes>
34              0 -1 11 -1.6071248683147132e-04</internalNodes>
35        .
36        .
37        .
38  </opencv_storage>
```

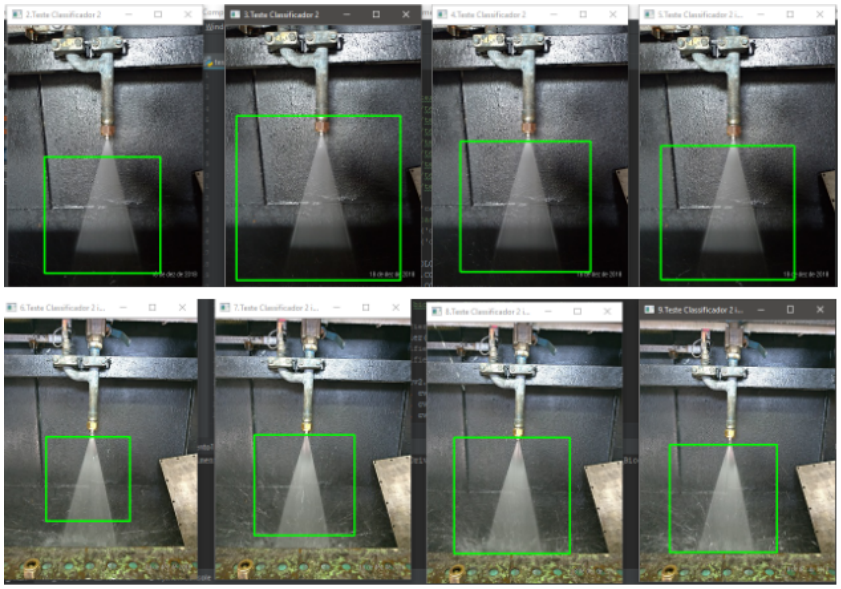
Fonte: Elaborado pela autora

5.6 Resultados obtidos

5.6.1 Experimento 1

Nos primeiros experimentos, o algoritmo foi treinado utilizando imagens positivas com a ROI maior. Isto é, considerando uma região mais ampla. Para esse cascade, foram utilizadas 400 imagens positivas para geração do arquivo com a listagem das imagens negativas e positivas sobrepostas e 3 imagens positivas de acordo com as classificações das imagens da base de dados. Para esse cascade obtivemos o resultado conforme mostrado na figura 40.

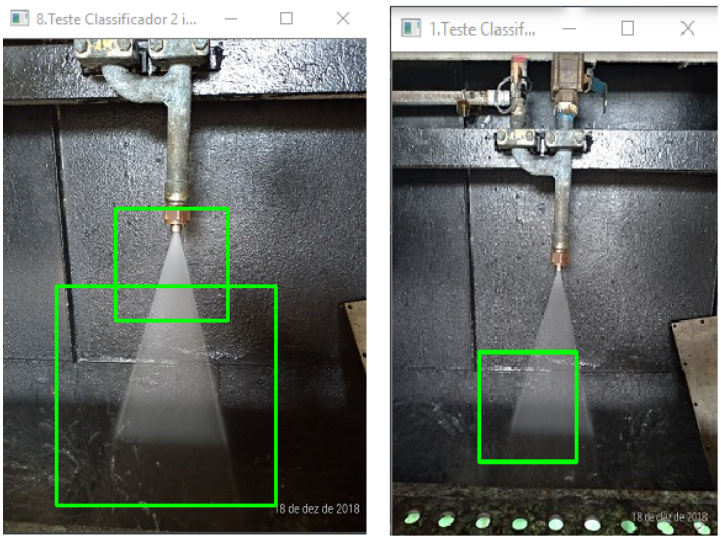
Figura 40 – Detecção do algoritmo haarcascade para o primeiro experimento utilizando 400 imagens negativas



Fonte: Elaborado pela autora

Em alguns casos, como mostrado na figura 41 observou-se a presença de falsos positivos. Nesses casos, é possível ajustar o resultado da classificação alterando os parâmetros *scaleFactor* e *minNeighbors*. O primeiro parâmetro especifica quanto o tamanho da imagem é reduzido em cada escala da imagem. Isso significa que quanto menor o valor passado, mais detalhadamente o algoritmo fará a busca do objeto de interesse. Isso implica em demora de processamento, uma vez que ele fará a busca mais lenta. O segundo parâmetro refere-se a quantos vizinhos cada retângulo candidato deve ter para manter-se como uma região em potencial que contêm o objeto. Valores muito altos significam detecções menores, porém, apresenta maior qualidade de detecção.

Figura 41 – Detecção de falsos positivos detectados pelo algoritmo haarcascade.

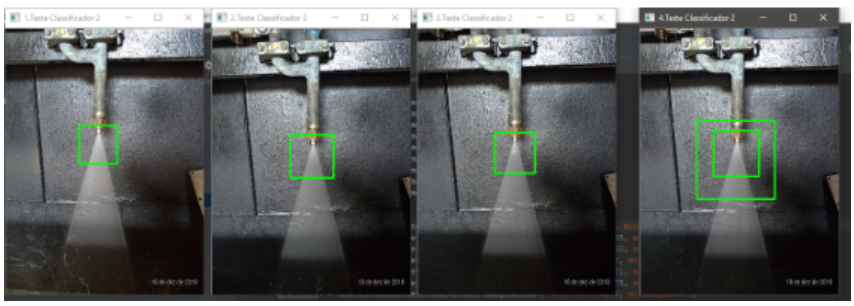


Fonte: Elaborado pela autora

5.6.2 Experimento 2

Para o segundo experimento, o algoritmo foi treinado com a ROI de tamanho menor. A criação do cascade foi refeita e definiu-se 5 novas imagens positivas de cada classe (mais próximas do início da emissão do jato) e com 500 imagens negativas com a sobreposição das imagens positivas. Assim, obteve-se o resultado mostrado na figura 42.

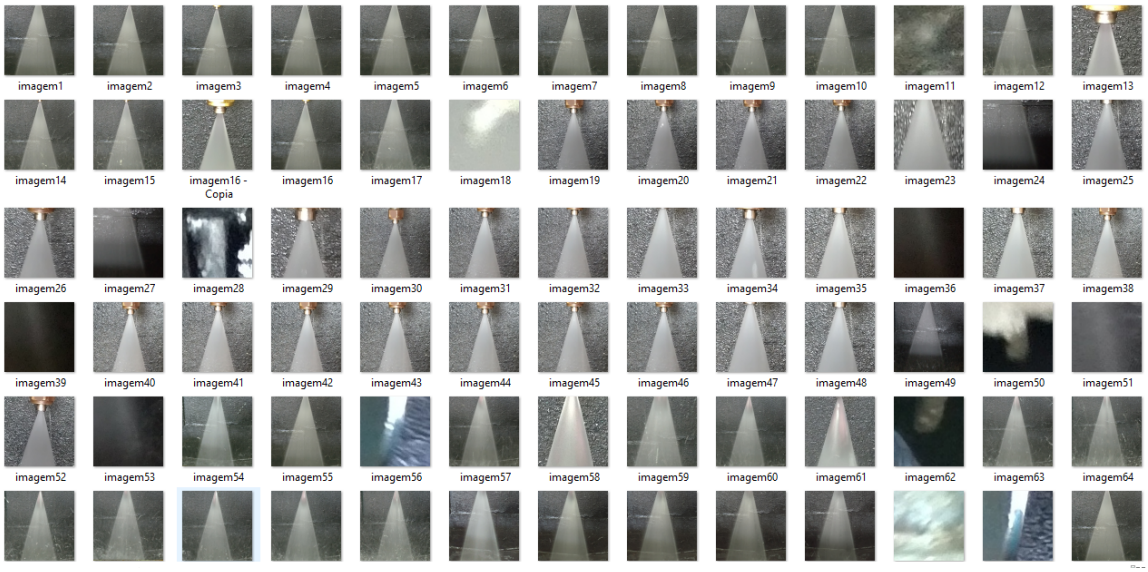
Figura 42 – Detecção do algoritmo haarcascade para o segundo experimento



Fonte: Elaborado pela autora

Os primeiros experimentos descritos anteriormente foram realizados considerando imagem por imagem para visualizar a influência dos parâmetros *scaleFactor* e *minNeighbors* de forma minuciosa. Entretanto, o algoritmo foi implementado de forma otimizada para processar as n imagens contidas na base de dados. Dessa forma, foi necessário escolher um limiar para o melhor resultado, isto é, para o menor número de falsos positivos possíveis. Após vários testes experimentais, o melhor desempenho foi encontrado como sendo $scaleFactor = 1.8$ e $minNeighbor = 8$ como mostra a figura 43

Figura 43 – Detecção do algoritmo haarcascade para o experimento com o limiar para scaleFactor e minNeighbors especificados



Fonte: Elaborado pela autora

6 Algoritmo proposto para extração de características

“Se você iniciar o fogo antes da linha do vento, nunca ataque contra o vento.
É provável que o vento que soprou constantemente
durante o dia, se acalme à noite”.
Sun Tzu

6.1 Pré-processamento

A imagem a ser tratada na etapa de pré-processamento é resultado da detecção da região de interesse processada pelo *Haarcascade* já cortada apenas com a região de interesse e em escala de cinza. Sendo assim, o primeiro passo foi redimensionar a imagem recebida para uma escala menor. Após experimentos realizados, a melhor dimensão para trabalhar com a imagem foi 200x300 pixels. Posteriormente, foram utilizados filtros de suavização aplicados nas imagens da base de dados.

Os filtros de suavização são utilizados para borramento e redução de ruído. Aplicando as operações de borramento nas imagens, foi possível remover os pequenos detalhes da imagem antes da extração de objetos e conectar pequenas descontinuidades nas linhas apresentadas pelas imagens que futuramente foram extraídas na segmentação.

6.1.1 Aplicação do filtro Gaussiano

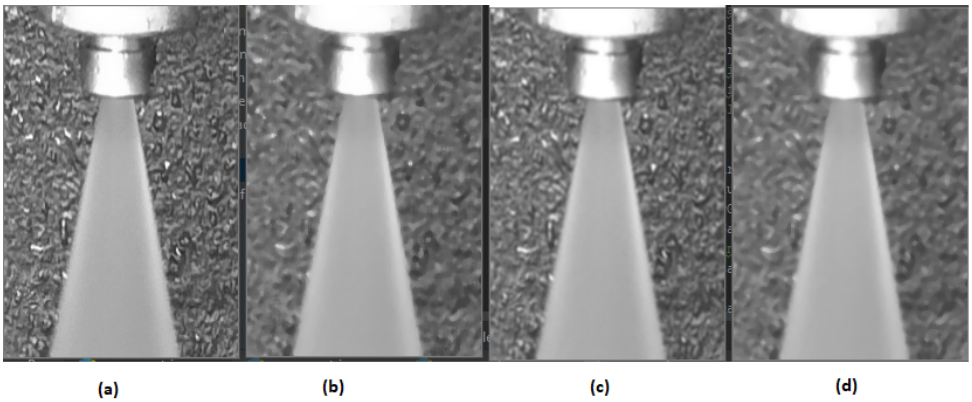
A primeira etapa consistiu em aplicar o filtro gaussiano para suavizar a imagem. Segundo Gonzales e Woods (2010), a ordem da matriz precisa ser um número ímpar e, por questões de simetria, uma matriz quadrada. Assim, o resultado da operação da máscara deslizante será aplicada no pixel central da imagem.

Como descrito em detalhes no referencial teórico deste trabalho, a largura de um filtro Gaussiano, ou seja, seu grau de suavização está relacionado com o parâmetro σ. Quanto maior o valor de σ, maior a largura do filtro Gaussiano e maior o seu grau de suavização. Para o desenvolvimento, optou-se por utilizar dois filtros de suavização distintos, o gaussiano e o de mediana. Assim, utilizou-se para o filtro Gaussiano, uma matriz de ordem $(3, 3)$ e o valor de $\sigma = 3$. Apenas o filtro Gaussiano aplicado o resultado é como mostra a figura 44 (c).

O filtro Gaussiano é um filtro passa-baixa e, como tal, um dos principais problemas relacionados à eliminação de ruído em imagens por meio desses filtros é a supressão de detalhes finos e bordas da imagem. Entretanto, por ser um filtro não linear, o filtro Gaussiano realiza a suavização de forma mais atenuada seguindo a distribuição normal. Assim, o filtro aplicado na imagem possui uma matriz com pesos distintos seguindo a função Gaussiana.

Como a aplicação do filtro espacial não interfere na detecção de bordas, é possível

Figura 44 – Imagem da ROI original (a), com filtro mediana (b), Gaussiano (c) e filtro Gaussiano e mediana (d) aplicado.



Fonte: Elaborado pela autora

aplicá-lo iterativamente. Assim, a aplicação do filtro de mediana permitiu remover o restante dos ruídos não removidos pelo filtro Gaussiano.

6.1.2 Aplicação do filtro de mediana

O filtro de mediana utilizado é não linear e consiste em substituir a intensidade de cada pixel pela mediana das intensidades na vizinhança do pixel. Segundo Pedrini e Schwartz (2008), considerando uma máscara de ordem n, sendo n ímpar, existe uma vizinhança n x n pixels em que a mediana das intensidades ordenadas encontra-se na posição $(n^2 + 1)/2$.

Segundo Gonzales e Woods (2010), os filtros de mediana proporcionam excelentes resultados na redução de ruído, com borramento consideravelmente menor do que filtros lineares de suavização de tamanho similar. Os filtros de mediana são eficazes na presença de ruído impulsivo, também chamado de ruído *sal e pimenta* em razão de sua aparência, como pontos brancos e pretos sobrepostos em uma imagem. Embora os ruídos na imagem em contexto não sejam essencialmente do tipo *sal e pimenta*, por ser passa-baixa e por isto combinar com o filtro Gaussiano, o filtro de mediana foi escolhido para suavizar as imagens do jato.

Após alguns experimentos e verificação na literatura, o limiar definido para o *kernel* (ordem da matriz) do filtro de mediana para o presente trabalho foi definido como $n = 5$. É válido ressaltar que quanto maior o valor definido para o *kernel*, maior o borramento realizado pela máscara e, portanto, mais detalhes finos podem ser removidos da imagem. Tendo em vista o contexto das imagens da base, quanto maior o borramento, melhor o resultado para a detecção das retas na etapa de segmentação. Portanto, o valor de $n = 3$ não foi o suficiente e $n = 5$ permitiu maior redução do ruído gerado pelo fundo das imagens.

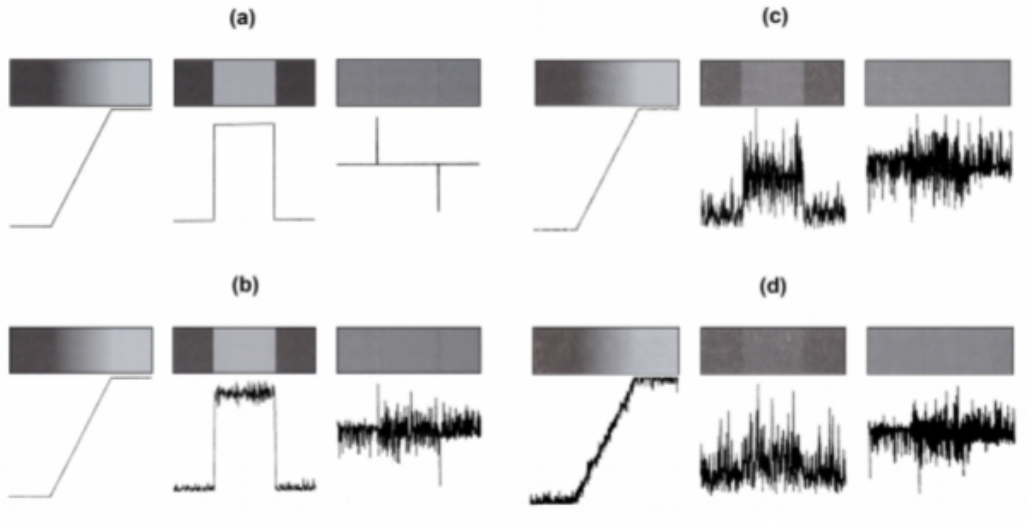
Nesta etapa, o interesse era reduzir os ruídos apresentados ao fundo da imagem do jato para que não fossem detectados como parte da região onde as características seriam extraídas e ao mesmo tempo que não suavizasse as bordas entre o jato e a parede. Os resultados foram como mostra a figura 44 (b).

A detecção de borda realizada com o operador de Canny, explicado na próxima seção,

é muito sensível aos pequenos ruídos presentes na imagem. Isso ocorre devido ao cálculo da derivada parcial e para isto a oscilação das cores em escala de cinza presente na imagem tem grande influência na detecção das bordas. A figura 45 ilustra o comportamento da derivada parcial de acordo com as variações das bordas. A primeira coluna em 45 (a) não apresenta ruído. As demais imagens 45 (b), (c) e (d) estão corrompidas por um ruído gaussiano aditivo com média zero e σ igual a 0,1 1,0 e 10,0 níveis de intensidade respectivamente. O gráfico abaixo de cada imagem é um perfil de intensidade horizontal onde o nível 0 representa o preto e o 255 o branco.

A segunda coluna de cada imagem é o resultado da aplicação da primeira derivada e os respectivos perfis de intensidade. Conforme acrescenta-se o ruído, percebe-se que o gráfico da derivada se torna cada fez mais diferente se comparada à derivada original da figura 45(a). A terceira coluna mostra as imagens resultantes da segunda derivada e os perfis de intensidade. Neste caso, percebe-se que a segunda derivada é ainda mais sensível ao ruído. A única imagem que se assemelha mais ao comportamento original é aquela correspondente a um ruído com $\sigma = 0,1$. As demais imagens de segunda derivada seria difícil detectar seus componentes positivos e negativos, que são características úteis da segunda derivada para a detecção de borda.

Figura 45 – Imagens de bordas em declives submetidas a um ruído gaussiano, os resultados da aplicação da derivada de primeira ordem, os resultados da aplicação da derivada de segunda ordem e os respectivos perfis de intensidade para cada caso: (a) Ruído com desvio padrão de 0,0; (b) Ruído com desvio padrão de 0,1; (c) Ruído com desvio padrão de 1,0; (d) Ruído com desvio padrão de 10,0.



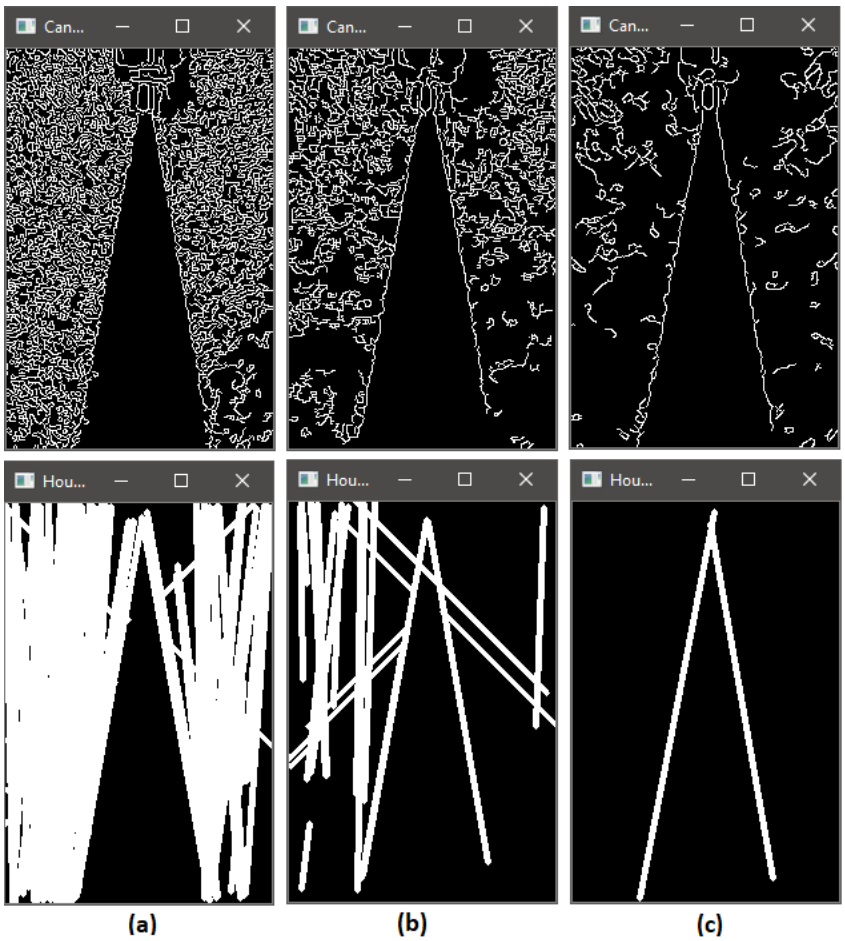
Fonte: (GONZALES; WOODS, 2010)

Portanto, convencionou-se que a utilização de dois filtros de suavização permitiria a melhor detecção das bordas. Assim, após a aplicação do filtro Gaussiano e em seguida o filtro de mediana, o resultado obtido é mostrado na figura 44 (d). Visualmente, a figura 44(b), (c) e (d) não apresentam diferenças exorbitantes, porém, os pixels de fundo estão relativamente mais suaves, isto é, sem mudanças bruscas na coloração como na imagem original e ao mesmo tempo, os pixels do jato contrastando com os pixels do fundo. As imagens com essas

características obtiveram um excelente resultado na etapa de delimitação das retas para o cálculo do ângulo.

A figura 46 ilustra o impacto que os filtros Gaussiano e de mediana geram para o resultado final das imagens. Como explicado pela figura 45, a delimitação das bordas em uma imagem a partir das derivadas parciais se torna muito sensível a presença de ruídos. Durante os experimentos realizados, isso se comprovou como mostrado na imagem 46. Na figura 46 (a) é mostrado a aplicação apenas de filtro Gaussiano, figura 46 (b) aplicação apenas de filtro mediana e 46 (c) aplicação da combinação dos filtros Gaussiano e mediana.

Figura 46 – Imagem da ROI e os resultados do processamento após aplicação de filtro Gaussiano (a); após aplicação de filtro de mediana (b) e filtro Gaussiano e mediana aplicados simultaneamente (c).



Fonte: Elaborado pela autora

Tendo em vista as características das imagens que compõem a base de dados, a metodologia da etapa de pré-processamento foi voltada a suprimir ruídos e delimitar as retas que são o formato do jato.

6.2 Segmentação

Na etapa de segmentação o objetivo é identificar corretamente a forma dos objetos presentes na imagem. Baseando-se principalmente na variação dos valores de intensidade dos pixels da imagem, detecção de descontinuidades e técnicas de limiarização.

Nesta etapa, as imagens resultantes da suavização dos filtros Gaussiano e mediana foram submetidos à operações para delimitação do jato. Essa operação foi realizada utilizando primeiramente o operador de Canny para detecção das bordas na imagem e a transformada de Hough para a obtenção das retas que parametrizam o formato do jato.

6.2.1 Operador de Canny

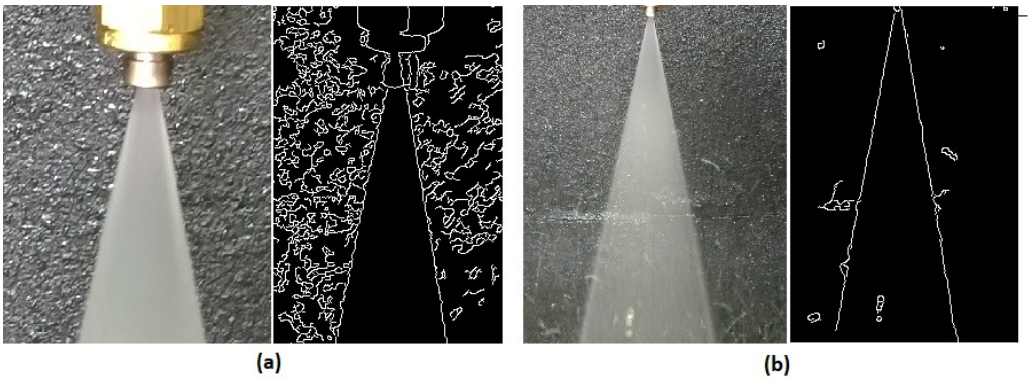
O principal gargalo para a aplicação de filtros segmentadores é a definição do limiar para reduzir falsos pontos de borda. O algoritmo de Canny tenta melhorar essa situação utilizando a limiarização por *histeresse* que, de forma simplificada e prática, utiliza dois limiares: um limiar baixo T_L e um limiar alto T_H. Canny sugeriu que a razão do limiar alto para o baixo deve ser de dois ou três para um.

Sendo assim, de acordo com experimentos realizados e seguindo a faixa estabelecida por Canny, utilizou-se o $T_L = 50$ e $T_H = 150$. Definiu-se, também, o valor do *kernel* igual a 3 para a ordem da janela deslizante.

A figura 47 mostra o resultado do tratamento após aplicação do filtro de suavização gaussiano e mediana e em seguida a aplicação do operador de Canny em duas ROI. Nessa figura é possível verificar que as imagens resultantes do pré-processamento foram diferentes. Na figura 47(a) a ROI está delimitada em uma região menor, já na figura 47(b) a ROI foi detectada com uma região maior. Isso implicou diretamente no resultado do operador de Canny. Uma vez que os ruídos da primeira ROI ficaram mais expostos.

Casos com essas características dificultaram na escolha do melhor limiar para a utilização de Canny. Uma vez que não poderia ser muito baixo (se não, para o caso de ROI menor apresentaria muito ruído e consequentemente dificultaria na extração de características) e nem muito alto, visto que poderia suprimir partes importantes da imagem caso fosse uma ROI com uma região maior.

Figura 47 – Resultado após aplicação do filtro Gaussiano, Mediana e operador de Canny para delimitar contorno do jato com uma ROI mais próxima do observador (a) e uma ROI mais distante do observador (b).



Fonte: Elaborado pela autora

6.2.2 Transformada de Hough

A transformada de Hough foi utilizada para a delimitação de retas formadas pelas bordas do jato como mostrada na figura 47. Para isso, a biblioteca OpenCV disponibiliza a transformada probabilística de Hough. Ela retorna de forma simplificada dois pontos para cada reta encontrada na imagem. Assim, é possível trabalhar algebricamente com a reta extraindo as informações pertinentes como o coeficiente angular.

Os principais parâmetros da transformada de Hough são listados a seguir juntamente com os valores convencionados para a detecção no algoritmo:

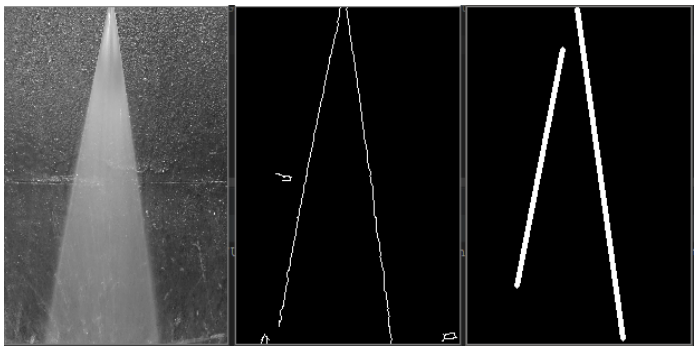
- $rho = 1$: distância do acumulador em pixels;
- $theta = \pi/180$: ângulo de resolução do acumulador em radianos;
- $threshold = 80$: valor do limiar do acumulador. Só serão selecionadas retas que possuírem quantidades de votos maiores que esse limiar;
- $min\_line\_length = 5$: tamanho mínimo para uma reta. Segmentos de reta menores que este atributo serão desconsideradas;
- $max\_line\_gap = 20$: distância máxima permitida entre pontos de uma mesma reta. Valores menores que este farão com que os pontos fiquem vinculados.

É válido ressaltar que definir o valor de limiar para max_line_gap não foi uma tarefa trivial, uma vez que foi necessário estabelecer um valor para todas as imagens a serem processadas. Após experimentos realizados, observou-se que o valor padrão de 20 foi o melhor valor para que não resultasse em pequenos fragmentos de retas (caso em que max_line_gap assume valores baixos) e não resultasse em retas falsas (caso em que max_line_gap assume valores altos os pontos mais distantes são vinculados).

Outro parâmetro igualmente importante definido é o limiar $threshold$. Ele representa a quantidade de *votos* contabilizados no plano acumulador (ρ, θ) e que assim possuem mais

chances de representarem uma reta no plano (x, y). Estabelecer esse valor também não foi uma tarefa trivial, uma vez que valores baixos faz com que muitas retas sejam detectadas e valores altos que retas pertinentes sejam desconsideradas. A figura 48 ilustra o resultado do processamento após a aplicação do filtro de Canny e a transformada de Hough para um caso bem sucedido.

Figura 48 – Primeira imagem é o jato real, a segunda do jato com delimitação de bordas e terceira com a transformada de Hough para detecção das retas segundo os critérios estabelecidos.



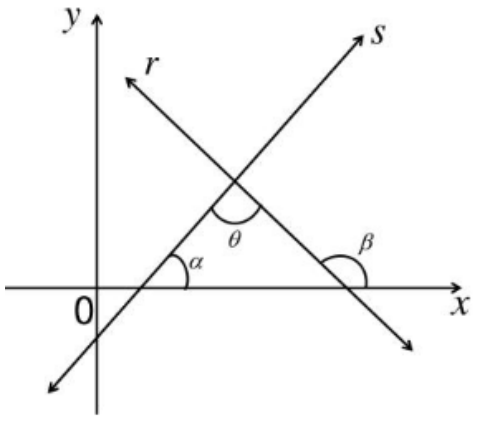
Fonte: Elaborado pela autora

6.3 Algoritmo proposto para extração das características

O algoritmo proposto consiste em determinar o ângulo que o jato apresenta e obter características como a distância de pulverização. Assim, após as etapas de detecção da ROI, pré-processamento e a delimitação das retas que originam o jato, foi necessário aplicar algum método matemático para definição do ângulo e a distância teórica de pulverização.

A ideia central para o cálculo do ângulo entre as duas retas delimitadas é a relação trigonométrica existente entre duas retas concorrentes em um plano. A figura 49 ilustra a relação entre duas retas que se interceptam formando um ângulo θ entre si.

Figura 49 – Posição relativa entre duas retas concorrentes.



Fonte: Elaborado pela autora

6.3.1 Cálculo dos parâmetros de interesse

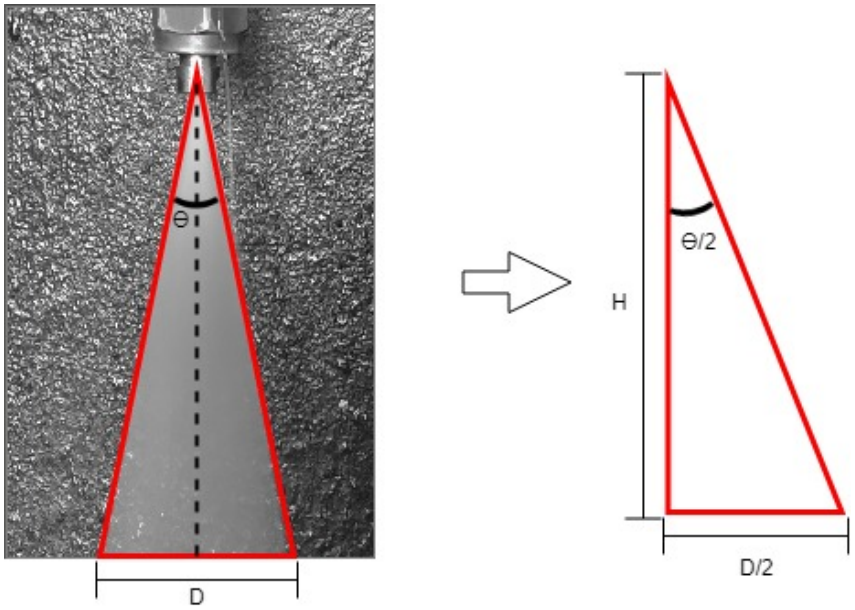
Por relação geométrica, é possível inferir que $\beta = \alpha + \theta$, aplicando relação de tangente em ambos os lados, temos que $tg(\theta) = tg(\beta - \alpha)$. Considerando m_r a inclinação da reta r e portanto a tangente do ângulo β e m_s a inclinação da reta s a tangente do ângulo α obtemos a relação trigonométrica mostrado na equação 6.1, consequentemente, o valor do ângulo θ é conforme mostrado equação 6.2.

$$tg(\theta) = \frac{m_r - m_s}{1 + m_r m_s} \tag{6.1}$$

$$\theta = arctg(\frac{m_r - m_s}{1 + m_r m_s}) \tag{6.2}$$

Com o ângulo de referência calculado, foi possível obter a distância de pulverização com uma aproximação esquemática do modelo real. Como mostra a figura 50 a imagem do jato pode ser aproximada de um triangulo retângulo com um ângulo $\theta/2$, cateto adjacente como a distância de pulverização informada pelo usuário (H), o cateto oposto a metade da cobertura teórica procurada $(D/2)$ e a hipotenusa uma das retas encontradas pela transformada de Hough.

Figura 50 – Relação trigonométrica para o cálculo da abertura do jato e da cobertura teórica.



Fonte: Elaborado pela autora

Dessa forma, o valor da cobertura teórica procurado pode ser escrito como 6.4

$$D = (tg(\frac{\theta}{2}) * H) * 2 \tag{6.3}$$

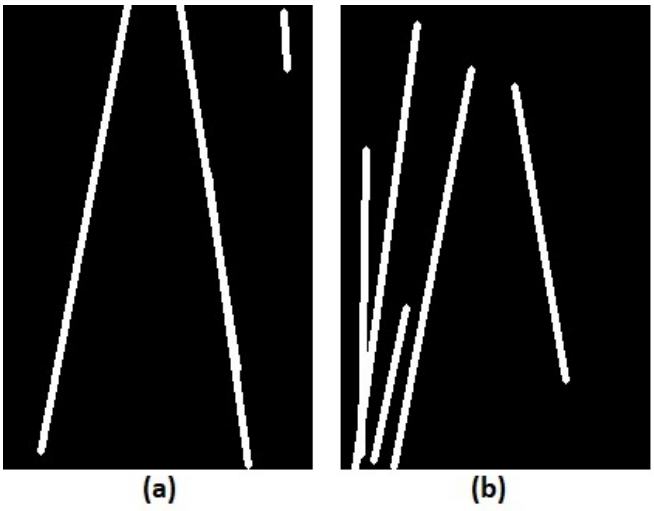
$$d = tg(\alpha) * H \tag{6.4}$$

6.3.2 Restrições aplicadas para o cálculo do ângulo

Para reduzir essa taxa de erro resultante das etapas anteriores, definiu-se uma faixa aceitável para os ângulos encontrados entre as retas encontradas pela transformada de Hough. Retas que apresentarem ângulos menores que 1^o ou maiores que 90^o são desconsideradas e o algoritmo busca a próxima reta que se encaixe na faixa $1^o < \theta < 90^o$.

Dessa forma, para os casos como mostrado na figura 51(a) não são consideradas válidas as retas colineares ou retas que formam ângulo próximo a zero (b). É importante ressaltar que devido ao valor do $threshold$ estabelecido na transformada de Hough (tendo em vista o resultado da detecção das bordas pelo operador de Canny) as primeiras retas a serem delimitadas são sempre as retas de interesse.

Figura 51 – Resultado da aplicação da transformada de Hough com detecção de retas colineares (a) e retas que não são de interesse para extração de característica (b).



Fonte: Elaborado pela autora

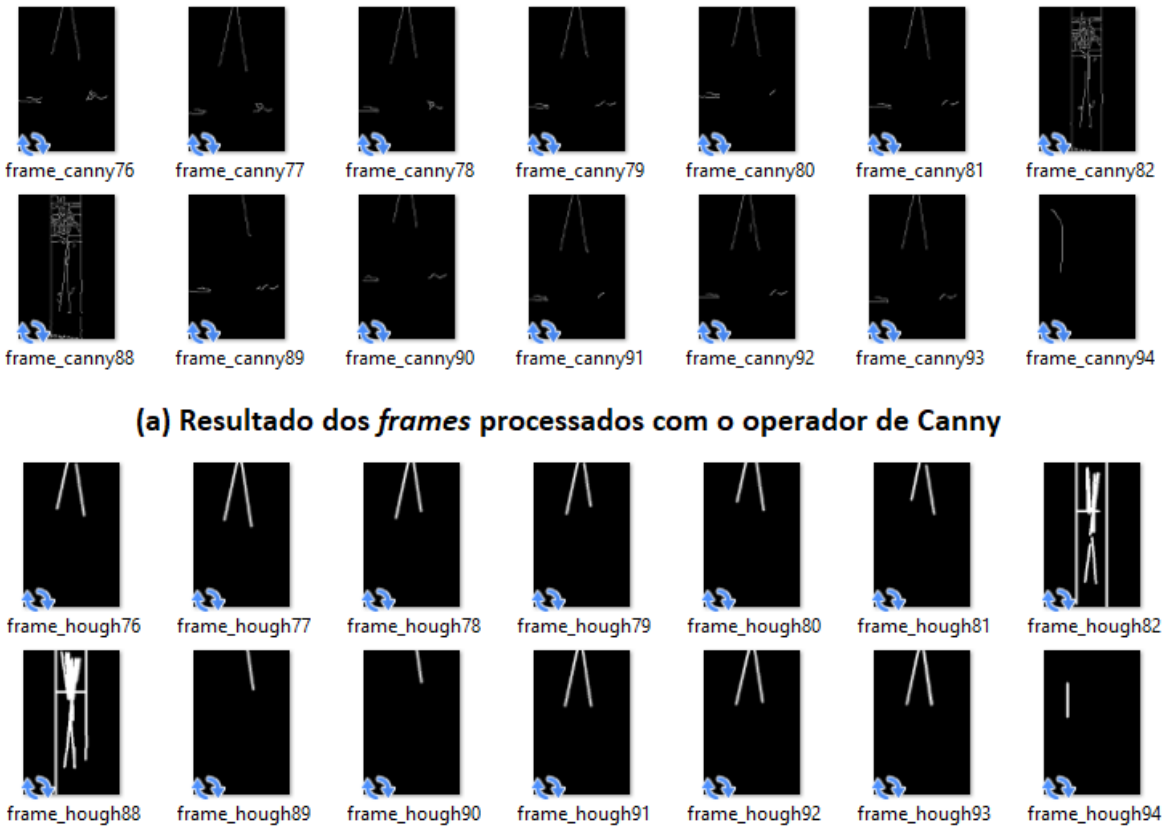
6.3.3 Testes realizados em vídeo

Utilizando o mesmo algoritmo desenvolvido para detecção do jato e extração das características foram realizados experimentos para captura de *frames* nos vídeos gravados na etapa de aquisição de imagens. O vídeo processado contêm 30 *frames* por segundo e resolução em HD (1280x720 pixels)

Para cada classe do bico adquirido os parâmetros do *Haarcascade* foram reajustados. Para a classe DNEX 1328 o *scaleFactor* = 2 e *minNeighbors* = 16; para a classe TC3040E *scaleFactor* = 1.9 e *minNeighbors* = 15 e para a classe TC 3040ENPRO *scaleFactor* = 3 e *minNeighbors* = 15. Dessa forma foi possível obter melhores resultados sem a detecção de falsos positivos ou falsos negativos.

A figura 52(a) mostra o resultado do processamento após a utilização do operador de Canny e a figura 52 (b) após a aplicação da transformada de Hough.

Figura 52 – Resultado da aplicação da do operador de Canny (a) e da transformada de Hough (b) para o vídeo da classe DNEX 1328.



Fonte: Elaborado pela autora

6.4 Avaliação do processo

O experimento foi realizado para três classificações de jato. Um jato tipo 3040E, um jato DNEX 1328 e outro 3040EN PRO. As tabelas a seguir da figura 53, figura 54, figura 57 mostra os resultados calculados pelo algoritmo de cada classe de imagem considerando a distância de pulverização como um parâmetro informado de 30cm.

As imagens que apresentaram *erro* foram originados de falsos positivos detectados pela primeira etapa com o haarcascade e por isso não foi possível detectar bordas com o operador de Canny e consequentemente traçar retas com a transformada de Hough como é o caso da imagem 18.

Figura 53 – Resultado dos ângulos e cobertura teórica calculados pelo algoritmo desenvolvido para classe do jato 3040E.

Imagem	Ângulo Calculado	Cobertura teórica
1	19,79	10,16
2	20,52	10,52
3	20,57	10,54
4	19,07	9,80
5	21,08	10,79
6	19,64	10,08
7	20,03	10,28
8	20,71	10,61
9	19,90	10,21
10	19,77	10,15
11	erro	erro
12	20,36	10,44
13	21,84	11,16
14	20,15	10,34
15	21,00	10,75
16	20,57	10,54
17	20,71	10,61
18	erro	erro

Fonte: Elaborado pela autora

Alguns casos especiais onde o ângulo resultou em 0^o como na imagem 23 (mostrado na figura 54) não foi possível traçar a reta pela transformada de Hough seguindo os valores estabelecidos nos parâmetros (valor de $threshold$ nesse caso foi alto demais para conseguir detectar uma reta). A figura 55 (a) mostra o resultado do processamento do operador de Canny aplicado na imagem 23 e a figura 55 (b) o resultado da reta detectada pela transformada de Hough.

Figura 54 – Resultado dos ângulos e cobertura teórica calculados pelo algoritmo desenvolvido para classe do jato DNEX 1328.

Imagem	Ângulo Calculado	Cobertura teórica	Imagem	Ângulo Calculado	Cobertura teórica
19	22,06	11,27	38	20,15	10,33
20	20,84	10,67	39	erro	erro
21	21,81	11,15	40	19,90	10,21
22	20,95	10,73	41	20,24	10,38
23	0,00	0,00	42	21,14	10,82
24	erro	erro	43	20,21	10,36
25	35,15	17,27	44	21,01	10,76
26	8,15	4,26	45	19,85	10,19
27	erro	erro	46	19,99	10,25
28	erro	erro	47	21,62	11,05
29	21,83	11,16	48	12,03	6,25
30	21,87	11,18	49	21,55	11,02
31	20,02	10,27	50	erro	erro
32	20,97	10,73	51	erro	erro
33	20,99	10,75	52	21,14	10,82
34	1,35	0,71	53	erro	erro
35	20,19	10,35	54	17,86	9,20
36	erro	erro	55	21,08	10,79
37	20,16	10,34			

Fonte: Elaborado pela autora

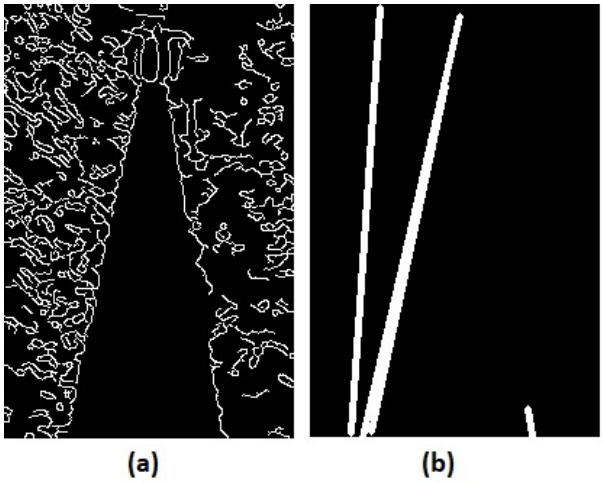
Figura 55 – Resultado do processamento da ROI não delimitada corretamente pelo Haarcascade da imagem 23 para a detecção de bordas (a) e detecção de retas (b).

Fonte: Elaborado pela autora

Outro caso para ser avaliado é a imagem 26 e imagem 34. A imagem 26 é um caso semelhante à imagem 23 como explicado anteriormente. Como mostrado na figura 56 (a), a detecção de bordas de Canny resultou em uma ruptura da lateral direita do jato que perdeu informações importantes da imagem devido aos filtros aplicados. Como o parâmetro de Hough considera como uma reta valores em que o acumulador seja maior que 80, não foi possível estabelecer uma reta na lateral direta do jato (figura 56(b)). Esse erro também poderia ter sido

solucionado caso o classificador tivesse detectado uma ROI melhor.

Figura 56 – Resultado do processamento para a detecção de bordas (a) e detecção de retas (b) em uma imagem ROI não delimitada corretamente na etapa do *Haarcascade*.



Fonte: Elaborado pela autora

Já a imagem 34 é um erro do próprio algoritmo para a definição de qual reta utilizar. Como o ângulo encontrado estava dentro da faixa de aceite permitido $1^o < \theta < 90^o$ o resultado foi tido como válido. Esse resultado faz parte da margem de erro previsto pelo algoritmo desenvolvido.

Para a classe de imagens jato 3040EN PRO não houveram casos especiais que fossem diferentes dos citados anteriormente. Os resultados estão listados na tabela da figura 57.

Figura 57 – Resultado dos ângulos e cobertura teórica calculados pelo algoritmo desenvolvido para classe do jato 3040EN PRO.

Imagem	Ângulo Calculado	Cobertura teórica
56	erro	erro
57	18,12	9,33
58	18,84	9,69
59	18,80	9,67
60	18,94	9,74
61	19,01	9,77
62	erro	erro
63	19,25	9,89
64	17,98	9,26
65	19,11	9,82
66	18,77	9,66
67	19,84	10,18
68	18,69	9,61
69	18,22	9,38
70	17,80	9,17
71	17,88	9,21
72	0,00	0,00
73	18,70	9,62
74	18,83	9,69
75	erro	erro
76	0,00	0,00
77	20,03	10,28
78	erro	erro

Fonte: Elaborado pela autora

A partir das figuras em tabelas mostradas anteriormente chegamos ao valor médio de aproximadamente 20º para o ângulo calculado pelo algoritmo e cobertura teórica de pulverização de 10,22cm para uma distância de pulverização igual a 30cm informada pelo usuário. Esses dados são mostrados na tabela 2.

Tabela 2 – Valores médios para o ângulo e a cobertura teórica calculada

Valor médio do ângulo calculado	**Valor médio da cobertura calculada**
19,95º	10,22cm

Fonte: Elaborada pela autora.

Tendo em vista a base de dados com 78 imagens no total, o classificador *cascade* conseguiu identificar corretamente 63 imagens dos bicos. Totalizando assim 19,23% de erro e 80,77% de acerto como mostrado na tabela 3.

Tabela 3 – Resultados para a detecção com *Haarcascade*.

Falsos positivos	15
Total de imagens na base de dados	78
Percentual de erro	19,23%
Percentual de acerto	80,77%

Fonte: Elaborada pela autora.

Para a etapa de pré-processamento as 63 imagens resultantes da detecção com o *Haarcascade* foram submetidas e dessas 5 imagens foram *outliers* apresentando incoerências tanto ao ângulo calculado quanto à cobertura teórica. Dessa forma, como mostrado na tabela 4 considerando 100% as 63 imagens submetidas às técnicas de suavização, segmentação e extração de características pelo algoritmo desenvolvido, foi obtido 92,06% de acertos e 7,94% de erros como mostrado na tabela 5.

Tabela 4 – Resultados gerais para a etapa de pré-processamento

Imagens que não apresentaram erros	58
Outliers removidos	5
Total de imagens válidas processadas	63

Fonte: Elaborada pela autora.

Tabela 5 – Percentual de erros e acertos do pré-processamento

Percentual de erros	**Percentual de acertos**
7,94%	92,06%

Fonte: Elaborada pela autora.

Para a avaliação do vídeo, o método desenvolvido foi aplicado a uma sequência de imagens, com as mesmas características das imagens isoladas, obtido na etapa de aquisição de imagem. O vídeo contêm originalmente 30 *frames*/segundo e com resolução em HD (1280x720 pixels) de 14 segundos. Como mostra a tabela 6, 219 *frames* foram processados no total resultando em uma taxa de processamento de 15,64 *frames*/segundo (50% do vídeo original). Destes 219 *frames* processados, 97 apresentaram erro de detecção e 29 apresentaram erro de cálculo do ângulo e da cobertura teórica resultando, portanto, em 93 *frames* válidos. Com isto, obtivemos uma taxa de acerto de 42,47% que *a priori* pode parecer um valor baixo. Entretanto, considerando um cenário com o tempo de monitoramento de análise igual a 60 segundos, a taxa de 42,47% é equivalente a 36 segundos consecutivos de defeito apresentado pelo jato. Ou seja, utilizando o método atuando em tempo real, é possível detectar que o jato apresenta algum tipo de defeito em até 36 segundos.

Outra análise importante a ser feita é em relação ao percentual de acertos do algoritmo para o cálculo do ângulo e da distância de pulverização sobre o total de *frames* sem erro de detecção. Dos 219 *frames* processados 97 apresentaram erro de detecção, portanto, 122 imagens foram de fato submetidas ao algoritmo para o cálculo. Destas, 93 foram calculadas representando assim em 76,23% de acerto em relação às imagens que de fato foram calculadas pelo algoritmo.

De forma análoga ao que foi descrito acima, a tabela 6 mostra os resultados da aplicação do método para as classes TC3040E e TC3040ENPRO.

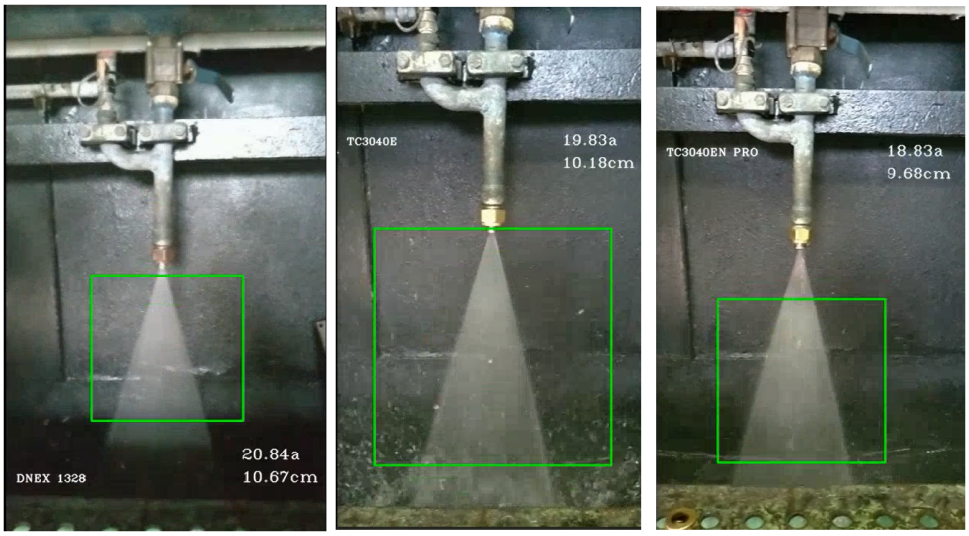
Tabela 6 – Percentual de erros e acertos do pré-processamento

	DNEX1328	**TC3040E**	**TC3040ENPRO**
Taxa de frames/segundo processados	15,64	12	15,62
Frames Processados	219	156	250
Erro Haarcascade	97	55	143
Erro cálculo parâmetros	29	12	24
Frames válidos	93	89	83
Percentual de acerto do método	42,47%	33,2%	57,05%
Percentual de acerto do algoritmo cálculo parâmetros	76,22%	77,57%	88,12%
Ângulo médio calculado	21,01º	19,25º	18,57º
Cobertura teórica média calculada	10,75cm	8,67cm	9,18cm

Fonte: Elaborada pela autora.

Uma observação importante é para o redimensionamento utilizado durante o processo. Como já explicado no desenvolvimento, as imagens de ROI foram processadas utilizando a dimensão 200x300 pixels por uma questão de custo computacional. O vídeo foi processado com esta mesma configuração para comprovar os resultados obtidos anteriormente com as imagens. Porém, utilizando a resolução original do vídeo foi possível processá-lo e obter resultados com valores maiores. Dessa forma, para um sistema real implantado com o método aqui desenvolvido, é possível processar as imagens utilizando a resolução original (em HD) das imagens capturadas. Entretanto, é necessário reajustar alguns parâmetros para a detecção do haarcascade e possivelmente o *threshold* da transformada de Hough. A imagem 58 mostra o resultado do processamento do *frame* DNEX 1328, o *frame* TC 3040E e o *frame* TC 3040EN PRO.

Figura 58 – Resultado da aplicação do método desenvolvido para os *frames* da classe DNEX 1328, TC3040E e TC3040EN PRO.



Fonte: Elaborado pela autora

7 Conclusão

“O prêmio maior de uma vitória é triunfar por meio de estratagemas, sem usar as tropas”.
Sun Tzu

Com base no referencial teórico e nas pesquisas realizadas para o desenvolvimento, este trabalho propôs um método para auxiliar o processo de avaliação e identificação de defeitos de bicos pulverizadores em bancada de teste. Sendo assim, esse trabalho descreve as técnicas a serem utilizadas para calibrar a acurácia da detecção e utilizando trigonometria, é capaz de medir parâmetros essenciais para avaliação do bico. Desse modo, o objetivo geral foi alcançado.

Os objetivos específicos também foram alcançados uma vez que o método desenvolvido foi capaz de detectar a região de interesse, delimitar segmentos de retas, representando as bordas do jato emitido, e calcular os parâmetros como o ângulo e a cobertura teórica para auxiliar o avaliador. A implementação foi capaz de reconhecer os jatos tanto em imagens quanto em sequências de imagens capturadas em tempo real.

O desenvolvimento desse projeto apresentou algumas dificuldades principalmente na etapa de aquisição de imagens reais para compor a base de dados. Impactando, assim, em uma análise minuciosa dos valores calculados em relação ao manual técnico disponibilizado pelas empresas fornecedoras.

Entretanto, a detecção da região de interesse e o processamento de forma simplificada da imagem demonstrou a validade em aplicar a visão computacional acoplada às técnicas de PDI para elaborar um processo de avaliação mais otimizado.

Como trabalhos futuros para a extensão desse projeto é possível citar:

- Implementar um sistema para interação com o usuário avaliador e a câmera para captura de imagens em tempo real (ou inserção de imagens digitalizadas ou arquivadas);
- Melhorar o classificador realizando treinamento com mais imagens e testar com a base de imagens diferenciadas;
- Utilização de RNA para classificação das diferentes variações dos jatos;
- Utilizar processamento em GPU.

Os experimentos realizados mostraram que é possível, utilizando visão computacional e técnicas de processamento digital de imagens, fazer análise dos jatos pulverizadores tendo em vista o acerto de 80,77% utilizando o *Haarcascade* para detecção da região de interesse e 92,06% de acertos utilizando filtragem espacial para suavização das imagens, operador de Canny para detecção de bordas, transformada de Hough para detecção de retas e operações

matemáticas para cálculo do ângulo e cobertura de pulverização. Para os testes realizados em vídeo, o método apresentou o percentual de 42,47% para o *frame* DNEX1328, 57,05% para o *frame* TC3040E e 33,2% para o *frame* TC3040ENPRO valores aceitáveis dado o cenário de processamento em tempo real. Quando avaliado apenas o algoritmo para o cálculo dos parâmetros do ângulo e cobertura teórica, a taxa de acerto apresentada foi de 76,22% para a classe DNEX1328, 77,57% para a classe TC3040E e 88,12% para a TC3040ENPRO.

Referências

ALMEIDA, R. H.; CORSO, D. A.; JUNIOR, A. Visão computacional–sistemas de visão aplicados à inspeção industrial. v. 6, n. 04, 2009. Citado na página 14.

ARAÚJO, S. A. d. *Casamento de padrões em imagens digitais livre de segmentação e invariante sob transformações de similaridade.* 2009. Tese (Doutorado) — Universidade de São Paulo, 2009. Citado na página 13.

AZEVEDO, E.; CONCI, A.; LETA, F. R. *Computação gráfica-volume 2: Teoria e prática*. [S.l.]: Elsevier Brasil, 2010. v. 2. Citado nas páginas 38 e 42.

BARELLI, F. *Introdução à Visão Computacional. Uma abordagem prática com Python e OpenCV*. 1ª edição. ed. Rua Vergueiro, 3185 - 8º andar. São Paulo.: Casa do Código, 2018. v. 1. ISBN 978-85-94188-57-1. Citado nas páginas 34, 35 e 36.

BASTOS-FILHO CARMELO JA E SILVA, W. A. e. L. L. R. Comparando meta-heurística para o treinamento adaboost aplicado à detecção de plaquetas. *IEEE Latin America Transactions*, IEEE, v. 12, n. 5, p. 942–950, 2014. Citado nas páginas 33 e 35.

BENTO, D. D. S. Técnicas de processamento de imagens para reconhecimento e análise da propagação de trincas em chapas de aço. 2016. Citado nas páginas 14 e 39.

FERNANDES, S. M. de C. et al. Desenvolvimento de um software de análise de imagens para caracterização microestrutural de materiais. *Exacta*, Universidade Nove de Julho, v. 10, n. 3, 2012. Citado na página 13.

FILHO, P. P. R. et al. New approach to evaluate a non-grain oriented electrical steel electromagnetic performance using photomicrographic analysis via digital image processing. *Journal of Materials Research and Technology*, Elsevier, 2017. Citado na página 14.

GOLNABI, H.; ASADPOUR, A. Design and application of industrial machine vision systems. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 23, n. 6, p. 630–637, 2007. Citado na página 13.

GONZALES, R. C.; WOODS, R. E. *Processamento Digital de Imagens*. 3ª edição. ed. Limão - São Paulo - SP: Casa do Código, 2010. Único. ISBN 978-85-7605-401-6. Citado nas páginas 18, 19, 21, 25, 26, 27, 31, 53, 54 e 55.

GRANATYR, J. *Reconhecimento facial com Python e OpenCV [online] Disponível em: https://www.udemy.com/reconhecimento-facial-com-python-e-opencv/learn/lecture/12839868overview*. 2018. Citado nas páginas 33 e 34.

GRIGIO, A. M. *Aplicação de sensoriamento remoto e sistema de informação geográfica na determinação da vulnerabilidade natural e ambiental do município de Guamaré (RN): simulação de risco às atividades da indústria petrolífera*. 2003. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2003. Citado na página 13.

GUEDES, M. A. et al. Caracterização física de grãos de soja utilizando-se processamento digital de imagens. *Revista Brasileira de Produtos Agroindustriais*, v. 13, n. 3, p. 279–294, 2011. Citado na página 13.

JAMUNDÁ, T. *Reconhecimento de Formas - A Transformada de Hough [online] Disponível em: <http://www.inf.ufsc.br/ visao/2000/Hough/>. [Acessado: 10-jun-2019]*. 2019. Citado na página 32.

JÚNIOR, F. A. d. O. *IDENTIFICAÇÃO E CLASSIFICAÇÃO DE SINALIZAÇÃO HORIZONTAL EM AUTOVIAS UTILIZANDO OPENCV*. 2016. Dissertação (Mestrado), 2016. Citado na página 40.

KHATCHATOURIAN, O.; PADILHA, F. R. Reconhecimento de variedades de soja por meio do processamento de imagens digitais usando redes neurais artificiais. *Eng Agric Jaboticabal*, v. 28, n. 4, p. 759–69, 2008. Citado na página 13.

KUROIWA, B. T.; CARRO, S. Detecção de intrusão com reconhecimento facial em imagens geradas por câmeras de segurança. In: *Colloquium Exactarum*. [S.l.: s.n.], 2015. v. 7, n. 2. Citado na página 13.

LULIO, L. C. *Técnicas de visão computacional aplicadas ao reconhecimento de cenas naturais e locomoção autônoma em robôs agrícolas móveis*. 2011. Tese (Doutorado) — Universidade de São Paulo, 2011. Citado na página 13.

MANUEL, L. do V. et al. Identificação e rastreamento de pessoas por meio de imagens digitais capturadas a partir de câmeras de video. In: *Colloquium Exactarum*. [S.l.: s.n.], 2016. v. 8, n. 1. Citado na página 13.

MARENGONI, M.; STRINGHINI, S. Tutorial: Introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, v. 16, n. 1, p. 125–160, 2009. Citado na página 13.

MARTINS, L. A. d. O. *Sistema de Inspeção Visual Automática Aplicado à Detecção de Defeitos em Aços Laminados*. 2010. Tese (Doutorado) — Centro Federal de Educação Tecnológica de Minas Gerais - CEFETMG, 2010. Citado nas páginas 14 e 38.

MENG, Y.; ZHUANG, H. Autonomous robot calibration using vision technology. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 23, n. 4, p. 436–446, 2007. Citado na página 14.

NEU, C. V. Desenvolvimento de um sistema de reconhecimento de sinais de trânsito utilizando processamento de imagens com opencv para um robô humanóide. 2014. Citado na página 13.

OLIVEIRA, R. B. et al. Classificação de assimetria em lesões de pele por meio de imagens usando máquina de vetor de suporte. In: *WVC 2012-VIII Workshop de Visão Computacional*. [S.l.: s.n.], 2012. Citado na página 13.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de Imagens Digitais*. São Paulo - SP: THOMSON, 2008. Único. ISBN 978-85-221-0595-3. Citado nas páginas 13, 18, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31 e 54.

PETERSON, L. et al. Análise cfd em chuveiros de descarepação. *ABM Week - Associação Brasileira de Metalurgia, Materiais e Mineração. 52º Congresso de Laminação*, ABM, p. 475–487, 2015. Citado na página 45.

PFEIFER, T.; WIEGERS, L. Reliable tool wear monitoring by optimized image and illumination control in machine vision. *Measurement*, Elsevier, v. 28, n. 3, p. 209–218, 2000. Citado na página 14.

QUEIROZ, L. C. L. *Classificação das amostras do ensaio de Baumann através do processamento digital de imagens.* 2015. Tese (Doutorado) — Universidade de São Paulo, 2015. Citado nas páginas 14, 20 e 38.

RUDEK, M.; COELHO, L. d. S.; JR, O. C. Visão computacional aplicada a sistemas produtivos: Fundamentos e estudo de caso. *XXI Encontro Nacional de Engenharia de Produção-2001, Salvador*, 2001. Citado na página 14.

SALIS, T. T.; PEREIRA, G. A. S. Contagem automática de tarugos de aço por meio de visão computacional (01). 2007. Citado nas páginas 14 e 39.

SCOMAZZON, M. Metodologia para identificação da posição de peças utilizando visão computacional. 2014. Citado na página 13.

SILVA, B. R. A. Sistema de contagem automática de objetos utilizando processamento digital de imagens em dispositivos móveis. *Master's thesis. Universidade do Estado do Rio Grande do Norte*, 2014. Citado nas páginas 13 e 39.

SOARES, H. B. Análise e classificação de imagens de lesões da pele por atributos de cor, forma e textura utilizando máquina de vetor de suporte. Universidade Federal do Rio Grande do Norte, 2008. Citado na página 13.

SPRAYINGSYSTEMS. *Por que a escolha certa do fabricante de bicos de pulverização é essencial na siderurgia? [online] Disponível em: https://www.spray.com.br/literature_pdfs/Portuguese/LI025-BR_Escolha_Certa_do_Fabricante_de_BicoS_de_Pulverizacao_Siderurgia.pdfVolume LI025BR. Acessado em 04/12/2018*. Citado na página 15.

SPRAYINGSYSTEMS. *Siderúrgica economiza 80 mil dólares e melhora sua operação com os novos bicos de descarepação DescaleJet Pro® [online] Disponível em: https://www.spray.com.br/literature_pdfs/Portuguese/CS159A-BR_Siderurgica_economiza_e_melhora_sua_operacao.pdf*. Citado na página 15.

YOON, H.-S.; CHUNG, S.-C. Machine vision inspection system of micro-drilling processes on the machine tool. *Transactions of the Korean Society of Mechanical Engineers A*, The Korean Society of Mechanical Engineers, v. 28, n. 6, p. 867–875, 2004. Citado na página 14.

ZHANG, L.; REN, Z.; LU, Q. Simulation of mesofracture process of asphalt mixture using digital image processing and extended finite-element method. *Journal of Testing and Evaluation*, ASTM International, v. 45, n. 1, p. 281–293, 2016. Citado na página 13.