FACULDADE DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE MATEMÁTICA

LEONOR ISABEL MORTÁGUA CASMARRINHA

Licenciatura em Matemática Aplicada à Gestão do Risco

DETEÇÃO DE PARTÍCULAS CONTAMINANTES PRESENTES EM AMOSTRAS DE ÓLEO LUBRIFICANTE

MESTRADO EM MATEMÁTICA E APLICAÇÕES

ESPECIALIDADE EM CIÊNCIAS DOS DADOS E DA DECISÃO

Universidade NOVA de Lisboa

Setembro, 2025

FACULDADE DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE MATEMÁTICA

DETEÇÃO DE PARTÍCULAS CONTAMINANTES PRESENTES EM AMOSTRAS DE ÓLEO LUBRIFICANTE

LEONOR ISABEL MORTÁGUA CASMARRINHA

Licenciatura em Matemática Aplicada à Gestão do Risco

Orientadora: Isabel Cristina Maciel Natário
Professora Associada, NOVA FCT

Coorientador: André Carmo e Silva
Especialista em Serviços Técnicos de Campo, GALP

Júri

Presidente: Doutora Ayana Maria Xavier Furtado Mateus
Professora Auxiliar, NOVA FCT

Arguente: Doutor Pedro José dos Santos Palhinhas Mota
Professor Associado, NOVA FCT

Orientadora: Isabel Cristina Maciel Natário
Professora Associada, NOVA FCT

MESTRADO EM MATEMÁTICA E APLICAÇÕES
ESPECIALIDADE EM CIÊNCIAS DOS DADOS E DA DECISÃO

Universidade NOVA de Lisboa
Setembro, 2025

Deteção de Partículas Contaminantes Presentes em Amostras de Óleo Lubrificante

Este documento foi gerado com o processador (pdf/Xe/Lua)LaTeX e o modelo NOVAthesis (v7.2.5) [1].

Agradecimentos

Começo por agradecer à minha orientadora, Professora Isabel Natário, pela disponibilidade em reunir comigo, pela paciência nas revisões feitas ao longo desta dissertação e pela forma como me guiou ao longo deste percurso.

Quero também agradecer à GALP pela oportunidade e pela integração na equipa. Em especial, aos meus orientadores, os engenheiros André Carmo e Silva e Paulo Miguel Ferreira, que mesmo quando não conseguiam ajudar diretamente, nunca hesitaram em abrir uma reunião no *Teams* para, em conjunto, olharmos para as dificuldades e pensarmos em soluções.

Agradeço aos meus pais por todo o investimento na minha educação, em especial à minha mãe, que partiu praticamente do nada e construiu tudo para que eu tivesse tantas oportunidades, pelo carinho especial e por todo o excelente trabalho que fez. Obrigada por estarem presentes nas vitórias, à minha irmã também, pela cumplicidade de sempre e por tornar certos momentos mais especiais.

Quero agradecer não à Praxe de LMAGR em si, mas sim às pessoas que fazem dela aquilo que é. Pessoas extraordinárias, que independentemente dos anos que passam ou da diferença de idades, mantêm sempre o espírito de companheirismo e boa disposição.

Em especial, agradeço de coração aos meus amigos. À Telma, que me acompanha desde que aprendemos a ler e que, mesmo estando noutra etapa de vida e num país diferente, é sempre sinónimo de tardes bem passadas, conversas longas e bons conselhos. Às minhas Catarina e Maria, que sei que levarei comigo para o resto da vida e que me enchem de alegria sempre que estou na presença delas. À Catarina mais velha, pela amizade sincera e genuína, sempre acompanhada de compreensão.

Agradeço ao José, que me apoia a 100% em qualquer plano, que me ajuda a descomplicar momentos de *stress* e a ser uma pessoa melhor. À Débora, uma das pessoas mais bondosas que conheço, sempre pronta a ouvir, a atender uma chamada ou a percorrer quilómetros. Ao Tomás, por estes 11 anos de amizade, partilha de casa e de turma; obrigada pela companhia nos momentos de tensão, nas noitadas de estudo e pelo apoio silencioso, mas sempre presente. E, por último, obrigada Duarte; por estares sempre lá, onde e quando for, por me ouvires, por apareceres sem que precise pedir, pelo apoio

incondicional e por me acompanhares nesta coisa chata e boa que é a vida.

Obrigada ao Jack, ao Snoopy e ao Nero pelos abraços que ampararam tantas lágrimas no vosso pelo, a forma de amor mais pura que existe.

Agradeço ainda aos meus tios Pisco, que tantas vezes me salvaram das tardes sozinha na escola. O meu tio ia buscar-me e, em casa, havia sempre os lanches que eu queria, espaço para partilhar o meu dia e a companhia da minha tia, que jogava comigo.

Por fim, obrigada à minha avó, que me dava tudo o que eu pedia, que brincava comigo e que até me deixava sempre escolher o canal da televisão. Todos os dias permanece nos meus pensamentos.

«Quero quem falha e pede desculpa, nunca o infalível; quero quem se equivoca, nunca o dono do mundo; quero quem se enganou muitas vezes no caminho, nunca o que quer ser o meu GPS; quero quem me conhece os defeitos e me ama ainda assim, nunca quem quer amar-me como uma personagem qualquer; quero a pessoa que envelhece porque caminha e anda e cresce e vive, nunca o quadro sempre impecável na parede; quero quem vive na terra, nunca quem vive no altar. Podem não ser as pessoas mais perfeitas do mundo, mas são as pessoas perfeitas para o meu mundo.»

— Pedro Chagas Freitas, A Raridade das Coisas Banais
(Escritor)

Resumo

Os lubrificantes industriais desempenham um papel essencial no bom funcionamento e longevidade das máquinas. No entanto, a sua eficácia pode ser gravemente comprometida pela contaminação com detritos metálicos resultantes do desgaste de componentes das máquinas. Os métodos atuais de deteção são demorados e dispendiosos, podendo atrasar decisões de manutenção e aumentar riscos operacionais.

Esta dissertação de mestrado, desenvolvida no âmbito do projeto MicroLubProbe da GALP, propõe uma metodologia automatizada para deteção e medição de partículas metálicas contaminantes presentes em imagens de microscopia de óleos e massas lubrificantes, recorrendo a técnicas de processamento digital de imagem. O algoritmo integra etapas de conversão de espaços de cor, aumento de contraste, redução de ruído e segmentação, implementados em Python com a biblioteca *Open Source Computer Vision Library* (OpenCV). Foram testadas várias técnicas de filtragem e realce avaliadas com mais do que uma métrica quantitativa.

O desenvolvimento inicial do algoritmo foi realizado com imagens obtidas em laboratório, uma vez que na fase inicial ainda não estavam disponíveis imagens recolhidas no terreno. Nessas condições controladas, e em imagens capturadas com foco adequado e iluminação homogénea, o algoritmo apresentou bom desempenho, identificando e quantificando partículas com precisão e consistência. No entanto, quando aplicado posteriormente às imagens de terreno, surgiram limitações significativas, o desempenho foi afetado por grandes diferenças no tipo de dados e nas condições de aquisição. Perante estes resultados, foi testada uma nova abordagem mas, dado que o conjunto de imagens disponível para esta fase foi extremamente reduzido (apenas duas imagens), não permitiu validar o método nem confirmar o seu potencial para integração em sistemas de monitorização em tempo real. São necessários mais dados de campo e ajustes de calibração para garantir a fiabilidade da solução fora do ambiente laboratorial.

Palavras-chave: Deteção de Partículas, Processamento de Imagens, Lubrificantes, Suavização, Operações Morfológicas, Diâmetro Maior

Abstract

Industrial lubricants play a key role in the proper functioning and longevity of machines. However, their effectiveness can be severely compromised by contamination with metallic debris resulting from the wear of machine components. Current detection methods are time-consuming and expensive, which may delay maintenance decisions and increase operational risks.

This Master's dissertation, developed within the scope of the MicroLubProbe project at GALP, proposes an automated methodology for the detection and measurement of metallic contaminant particles in microscopy images of oils and greases, using digital image processing techniques. The algorithm integrates steps of colour space conversion, contrast enhancement, noise reduction, and segmentation, implemented in Python with the OpenCV library. Several filtering and enhancement techniques were tested and evaluated using more than one quantitative metric.

The initial development of the algorithm was carried out using laboratory-acquired images, since at that stage field images were not yet available. Under these controlled conditions, and with images captured with proper focus and homogeneous lighting, the algorithm performed well, accurately and consistently identifying and quantifying particles. However, when later applied to field images, significant limitations emerged, as performance was affected by substantial differences in data type and acquisition conditions. In light of these results, a new approach was tested; however, the dataset available for this phase was extremely limited (only two images), preventing the validation of the method or confirmation of its potential for integration into real-time monitoring systems. Additional field data and calibration adjustments are required to ensure the reliability of the solution outside the laboratory environment.

Keywords: Particle Detection, Image Processing, Lubricants, Smoothing, Morphological Operations, Major Diameter

Índice

Anexos

Índice de Figuras

Siglas

CLAHE *Contrast Limit Adaptative Histogram Equalization (pp. x, xi, xiii, 5, 18–21, 55, 56, 61, 62)*

HSV *Hue, Saturation & Value (pp. x, xiv, 7, 16, 72, 73, 80–82, 85, 86)*

MSE *Mean Square Error (pp. xi, 5, 6, 33, 62)*

OpenCV *Open Source Computer Vision Library (pp. vii, ix, 9, 13–15, 23, 29, 36, 44, 46, 48, 54, 81, 82, 86)*

PIB Produto Interno Bruto *(p. 2)*

PSNR *Peak Signal to Noise Ratio (pp. xi, 5, 6, 33, 62)*

RGB *Red, Blue & Green (pp. 12–16, 54)*

RNC Redes Neuronais Convolucionais *(pp. 6, 7, 87)*

SR Sinal-Ruído *(p. 7)*

SSIM *Structural Similarity Index Method (pp. xi, 33, 34, 62)*

1

INTRODUÇÃO

1.1 Contextualização do Problema

A presente dissertação insere-se no projeto "MicroLubProbe", proposto pela empresa GALP. O projeto visa abordar problemas críticos relacionados com a análise da qualidade de lubrificantes no contexto industrial, promovendo a investigação de métodos de diagnóstico mais automatizados. Este trabalho tem como objetivo desenvolver uma metodologia automática para a deteção de partículas metálicas em amostras de óleo e massa lubrificante, com recurso a técnicas de análise de imagem digital. Através da aplicação de etapas sistemáticas, como o pré-processamento, que inclui a aplicação de filtros, o realce de contraste e a redução de ruído, segmentação de regiões de interesse e extração de características morfológicas, procura-se identificar e quantificar de forma precisa as partículas presentes. A informação quantitativa extraída será o maior diâmetro das partículas para a avaliação da contaminação. O algoritmo que se propõe é uma alternativa objetiva, eficiente e reprodutível em relação à análise manual, contribuindo para o controlo de qualidade e diagnóstico precoce de desgaste.

Lubrificação

A lubrificação é o uso de substâncias entre superfícies em movimento para reduzir o atrito, o desgaste e a temperatura [2]. A sua origem provém de agricultores que utilizavam gordura animal para lubrificar os eixos dos seus carros de bois, atualmente nas máquinas modernas são aplicados produtos quimicamente muito mais complexos [3]. Apesar dessas mudanças, o princípio básico de prevenir o contacto metal-metal por meio de uma camada intermediária de fluído, ou um material semelhante a este, permanece inalterado [4]. A substância que compõe esta cama intermediária é conhecida como lubrificante, e a sua função principal é reduzir o atrito e o desgaste entre superfícies móveis [4].

Lubrificantes

Os lubrificantes podem ser compostos por óleos minerais, sintéticos ou vegetais, podendo ainda ser muitas vezes combinados com aditivos [5]. Na indústria de máquinas, os lubrificantes são indispensáveis para todos os tipos de equipamentos, desde motores automóveis até sistemas de energia eólica [6]. Ainda neste contexto, desempenham um papel crucial no funcionamento das máquinas, nomeadamente evitam o superaquecimento e, como já mencionado, formam uma barreira que impede o contacto direto entre superfícies reduzindo o atrito [7], que por sua vez origina uma diminuição do desgaste das superfícies. Esta diminuição do atrito leva a que os lubrificantes desencadeiem processos mais fluídos aumentando a durabilidade das máquinas e protegendo os componentes críticos que as compõem, reduzem ainda o consumo de energia que também leva a um menor gasto económico. A combinação destas características ajuda a evitar as falhas associadas aos equipamentos, aumentando o tempo de vida útil destes [5, 8]. Financeiramente também apresenta benefícios, cerca de 30% do consumo de energia primária mundial está relacionado com o atrito, as perdas económicas causadas pela dissipação de energia devido ao atrito e pelo desgaste representam cerca de 2% a 7% do Produto Interno Bruto (PIB) de diferentes países todos os anos [9].

Além disso, estes também apresentam outras funções benéficas, tais como, reduzir vibrações e ruídos operacionais, tornando o ambiente industrial mais confortável e benéfico para os trabalhadores [8].

Contaminação de Lubrificantes

Embora os lubrificantes sejam indispensáveis para o bom funcionamento das máquinas, a sua eficácia pode ser comprometida quando se encontram contaminados. A contaminação de lubrificantes ocorre quando partículas sólidas, líquidos ou outros materiais indesejados entram no fluído lubrificante, interferindo com as suas propriedades [10]. As fontes de contaminação incluem, por exemplo, partículas metais resultantes do desgaste mecânico de componentes, poeira proveniente do ambiente envolvente, resíduos de processos industriais e água ou produtos químicos [11]. Os impactos desta contaminação são severos principalmente quando é devido à presença de partículas sólidas, até mesmo de dimensões muito pequenas como 10 μm, que podem atuar como abrasivos entre superfícies em contacto, danificando os componentes [12]. O tamanho e concentração em que os contaminantes se encontram são fatores que afetam muito negativamente esta interação entre superfícies [13]. Estes problemas frequentemente resultam em custos elevados consequentes da manutenção corretiva, interrupções inesperadas na produção e, em casos mais graves, falhas catastróficas [14].

Monitorizar e controlar a contaminação é essencial, se os lubrificantes se encontrarem em boas condições é possível manter a funcionalidade ideal dos equipamentos para que operem de modo contínuo e confiável. A deteção precoce de contaminantes evita não só

problemas maiores, como também aumenta a segurança dos trabalhadores, reduzindo os riscos de acidentes relacionados ao mau funcionamento das máquinas [15].

1.2 Objetivos

No contexto da empresa GALP, a análise da qualidade dos lubrificantes é essencial para garantir a longevidade e bom funcionamento dos equipamentos dos clientes compradores dos seus produtos, evitando o quanto possível falhas não programadas decorrentes do uso de lubrificantes que já não atendem às condições ideais de desempenho. Estudos como o realizado por Susan Benes [16] demonstram as limitações dos métodos tradicionais utilizados para deteção de partículas contaminantes, como a espectroscopia e a contagem ótica de partículas, que frequentemente assumem que partículas são perfeitamente arredondadas, levando a resultados potencialmente imprecisos.

O processo atualmente adotado pela empresa GALP para verificar o estado do óleo lubrificante em uso numa determinada máquina é demorado e dispendioso, envolve etapas como a coleta da amostra, envio para laboratório e posterior análise por um técnico especializado. Esta análise é realizada através de ferrografia, uma técnica que permite identificar detritos de desgaste e partículas contaminantes presentes no lubrificante, por meio da separação dessas partículas utilizando campos magnéticos e a sua observação ao microscópio [17]. Com base nos resultados obtidos, é elaborado um relatório técnico que é enviado à GALP, responsável por comunicar ao cliente a necessidade ou não de substituição do óleo, bem como a sua urgência. Este procedimento pode levar até duas semanas em casos nacionais, sendo ainda mais demorado para clientes internacionais, o que não só acarreta custos médios na ordem dos 200 euros por análise, como também gera um período de incerteza quanto ao real estado de funcionamento das máquinas, podendo, em casos críticos, resultar em danos evitáveis caso a substituição do óleo seja adiada.

Esta dissertação procura desenvolver um algoritmo automatizado baseado em técnicas de análise de imagens para identificar e medir partículas contaminantes diretamente a partir dessas imagens de microscópio, permitindo futuramente uma análise automática e em tempo real, reduzindo significativamente o tempo de processamento e os custos associados. Após o investimento inicial no microscópio, as análises subsequentes implicariam apenas o custo unitário das lamelas, considerando que na aquisição mais recente, 100 lamelas custaram 15 euros, o custo associado a cada observação ficaria reduzido a apenas 15 cêntimos por observação.

Inicialmente, o desenvolvimento do algoritmo será testado em imagens obtidas em laboratório, onde as condições são controladas e bem definidas, o que permitirá validar o modelo antes da sua implementação em campo. Após essa fase inicial, o algoritmo será testado diretamente em amostras coletadas no terreno, onde poderá ser avaliada a sua eficácia em condições reais de operação. O objetivo final é viabilizar a monitorização do óleo de forma prática e eficiente no local de operação, capturando uma imagem e

determinando diretamente a partir da análise dessa imagem o nível de contaminação do óleo.

2

Estado da arte

O processamento de imagens digitais tem passado por um crescimento e evolução significativos. Inicialmente, na década de 1970 era um campo com aplicações limitadas [18], mas expandiu-se rapidamente devido aos avanços no poder de computação e nos métodos de obtenção de dados [19]. O aumento no volume de dados, apresentou novos desafios e oportunidades no processamento de imagens [20]. Esse crescimento foi impulsionado pela versatilidade do campo em gerenciar dados complexos e as suas aplicações em diversos domínios, como segurança e saúde [21]. O surgimento das redes neuronais e o desenvolvimento de abordagens inovadoras de inteligência artificial levaram a avanços significativos no entendimento e processamento de imagens [20].

Esta dissertação baseia-se em metodologias de segmentação de imagens, deteção e identificação de partículas. Tendo em consideração as condições de baixa luz e elevada ampliação em que as fotografias microscópicas são obtidas, estas estão suscetíveis a vários tipos de ruído, o que pode degradar a qualidade da imagem [22]. O ruído em imagens é uma variação aleatória nas informações de brilho ou cor que pode degradar a qualidade das imagens digitais [23]. A remoção de ruído em imagens, um processo que elimina o ruído enquanto preserva as informações importantes, é crucial. Para atingir esse fim várias técnicas de remoção de ruído foram propostas, nomeadamente, filtros espaciais, como média, mediana, gaussiano e Wiener podem ser eficazes [24]. Devido também às condições em que as fotografias microscópicas são obtidas, aprimorar o contraste da imagem pode também ser essencial. Métricas de avaliação como *Peak Signal to Noise Ratio* (PSNR) e *Mean Square Error* (MSE) têm sido utilizadas para avaliar a eficácia dessas técnicas [25]. A equalização de histograma é uma técnica comum para o aumento de contraste, mas frequentemente resulta em contraste excessivo e destaque de ruído, para isso diversos métodos foram desenvolvidos. Esses métodos utilizam diferentes critérios para dividir o histograma de entrada, de modo a preservar o brilho e reduzir o excesso de contraste [26]. O *Contrast Limit Adaptative Histogram Equalization* (CLAHE) é uma técnica avançada que supera as limitações da equalização de histograma padrão ao aprimorar o contraste localmente. O CLAHE automático com correção gama dupla foi proposto para aprimorar o contraste enquanto preserva a naturalidade da imagem, sendo especialmente

eficaz para imagens com baixo contraste [27].

O trabalho de Ferraretti *et al.* [28], publicado em 2011, aborda a etapa de segmentação, que é crucial no processamento digital de imagens, destacando a sua importância e os desafios que pode apresentar, especialmente em imagens complexas. A segmentação consiste em dividir uma imagem em múltiplos segmentos para facilitar a sua análise e interpretação [29]. Os autores propõem e testam três algoritmos baseados em técnicas de limiarização global e local para deteção de manchas em imagens com ruído. A limiarização é uma técnica crucial de processamento de imagens usada para separar o primeiro plano do plano de fundo em imagens digitais [30], a limiarização global aplica um único valor de limiar para toda a imagem, enquanto que a limiarização local calcula diferentes valores de limiar para cada pixel com base nas informações dos píxeis vizinhos [31]. Para validação, esses três algoritmos são aplicados em imagens interpretadas por geólogos para identificar e quantificar zonas de profundidade que possam conter óleo e gás. Os geólogos observam diretamente os resultados das imagens escolhidas em diferentes profundidades do poço e selecionam o melhor algoritmo entre os propostos.

O trabalho de Shanthi *et al.* [32] baseia-se em técnicas de processamento de imagem para a análise do tamanho de partículas, técnica usada para determinar a variação no tamanho das partículas presentes numa amostra. Os autores utilizam uma abordagem baseada em limiarização e operações morfológicas, as principais operações morfológicas são a erosão e a dilatação [33]. A erosão é uma técnica que analisa uma imagem para identificar as áreas onde um elemento estruturante se pode encaixar completamente dentro dos objetos da imagem. Já a dilatação faz o oposto, expandindo os objetos da imagem de modo a preencher espaços ao redor destes [34]. Após a limiarização e aplicação de operações morfológicas, os autores utilizam algoritmos para deteção de bordas de estruturas presentes em imagens. A deteção de bordas é uma técnica fundamental no processamento de imagens que identifica mudanças abruptas na intensidade dos píxeis, marcando as fronteiras entre diferentes regiões constituintes de uma imagem [35]. Por último, determinam o diâmetro através da área e o diâmetro de Feret, que representa a maior distância entre dois pontos da borda de um objeto, comummente medida ao longo de uma direção específica ou em várias direções para descrever as dimensões do objeto [36]. O método apresentado é apontado como uma alternativa viável e eficaz para medições realizadas em tempo real diretamente por meio de sistemas computacionais. Em cenários industriais, como esteiras transportadoras, elimina a necessidade de métodos tradicionais aplicados em campo, como a peneiração. Em 2024, Karakuş [37] comparou o algoritmo de deteção de bordas desenvolvido por Canny com o desenvolvido por Sobel com a adição de operadores morfológicos, constatando que o algoritmo de Canny superou o algoritmo de Sobel com base nas métricas MSE e PSNR. O estudo também concluiu que a aplicação de operações morfológicas melhorou o desempenho de ambos os algoritmos. Esses resultados destacam o potencial das operações morfológicas no aprimoramento das técnicas de deteção de bordas.

O artigo de Mabaso *et al.* [38], em 2018, destaca o potencial que as Redes Neuronais

Convolucionais (RNC) têm no contexto da segmentação de imagens microscópicas. As RNC constituem uma arquitetura avançada de rede neuronal artificial amplamente utilizada em *deep learning* para tarefas de reconhecimento e classificação de imagens [39]. Os autores propõem o uso de uma arquitetura de RNC, à qual deram o nome de detectSpot, para detetar pontos em imagens de microscopia baseada na técnica *sliding-window*. A ideia central é ter uma "janela"de tamanho fixo, que se move através dos dados ou pacotes de informação à medida que são enviados, esta janela é usada para controlar que dados podem ser enviados ou recebidos a qualquer momento [40]. A metodologia apresentada consiste em treinar uma RNC supervisionada para identificar manchas em imagens de teste, com uma etapa subsequente de *overlap suppression*, uma técnica para eliminar dados ou regras redundantes de modo a melhorar a eficiência e a clareza das informações [41], garantindo que cada mancha seja detetada uma única vez. Os testes envolveram duas etapas: a utilização de dados sintéticos gerados com diferentes relações Sinal-Ruído (SR) e a comparação do detectSpot, o nome dado ao modelo desenvolvido pelos autores, com modelos pré-treinados, concluindo que o uso de modelos pré-treinados, adaptando-os ao tipo de dados com que se está a lidar, pode ser uma alternativa, eliminando a necessidade de treinar RNC do zero.

O trabalho de Tongur *et al.* [42], publicado em 2023, revisita o uso de técnicas de processamento de imagem para a medição do tamanho das partículas, destacando a importância do pré-processamento e da segmentação na precisão dos resultados. Os autores discutem o uso de métodos clássicos como a limiarização, filtros de suavização e realce de contraste para segmentar partículas em imagens microscópicas. Tongur *et al.* [42] também exploram como a combinação dessas técnicas pode ser aplicada em várias áreas industriais. Os autores realizam um estudo comparativo entre as técnicas clássicas e as abordagens mais modernas baseadas em *machine learning*, concluindo com esta pesquisa que, apesar do avanço das técnicas baseadas em *deep learning*, como o trabalho desenvolvido por Mabaso *et al.* [38], os métodos tradicionais continuam a ser eficazes e viáveis, especialmente quando as exigências computacionais ou o tamanho dos dados não são tão elevados. Têm ainda sido obtidos resultados utilizando um espaço de cores diferente, o espaço *Hue, Saturation & Value* (HSV), que explora o uso deste espaço de cores para detectar e classificar partículas metálicas em várias aplicações. Chun *et al.* [43] desenvolveram uma técnica de inspeção de partículas em *wafers*, lâminas finas de material semicondutor, utilizando modelos de transformação do espaço de cores HSV para detetar e classificar com precisão partículas em escala de micrômetros em *wafers*.

Rameshkumar *et al.* [44], em 2024, propuseram uma abordagem baseada em *machine learning* para prever o nível de contaminação por partículas sólidas em óleos lubrificantes, utilizando dados de sensores e técnicas de análise de vibração. O modelo desenvolvido demonstrou alta precisão na previsão da contaminação e eficácia no monitoramento em tempo real da condição do óleo. Para esta abordagem é combinada análise de imagem com sensores industriais e *deep learning*, sensores industriais medem diversos parâmetros como temperatura e pressão, além de métricas mais avançadas, como tamanho de

partículas [45]. Esta combinação permite uma manutenção preditiva menos dependente de intervenções manuais. Também um estudo desenvolvido no mesmo ano por Royer *et al.* [46] aborda avanços na segmentação e classificação de detritos presentes em imagens, neste caso micro plásticos, utilizando como base modelos de *deep learning* e obtendo 96% de precisão, o que significa que 96% das vezes que o modelo fez uma previsão, essa previsão estava correta, e apenas 4% das previsões feitas estavam erradas.

-

3

MÉTODOS

3.1 Processamento de Imagens

Neste trabalho, iremos apresentar uma série de técnicas de processamento de imagem com o objetivo de identificar e extrair estruturas relevantes a partir de imagens já digitais, ou seja, imagens que não necessitam de digitalização prévia. Essas técnicas serão aplicadas de forma sistemática, iniciando-se pelo pré-processamento, com o intuito de preparar a imagem para as etapas seguintes, para posteriormente ser possível realizar operações mais avançadas como a segmentação de regiões de interesse e a extração de características que auxiliem na identificação das estruturas desejadas.

O processamento de imagens é um método computacional que transforma imagens de entrada em imagens de saída e/ou que extrai destas informação útil [47], com o objetivo concreto de facilitar a interpretação, a análise ou a tomada de decisão baseada no conteúdo visual. Normalmente, envolve tratar imagens como sinais bidimensionais, ou seja, uma matriz de valores organizados em duas dimensões, a altura (linhas) e a largura (colunas) da imagem, sendo que cada valor nesta matriz representa a intensidade de cor ou brilho de um píxel específico. Esses valores podem ser manipulados por técnicas de processamento de imagem de modo a extrair características ou realizar outras operações [48]. Este campo abrange diversas técnicas e aplicações, envolve conceitos fundamentais como amostragem, quantificação e conectividade de píxeis, que diz respeito à forma como os píxeis se relacionam com os seus vizinhos dentro da imagem, assim como tópicos avançados como aprimoramento, restauração e segmentação de imagens [49]. A restauração de imagens consiste em melhorar a qualidade de imagens degradadas, corrigindo distorções, como por exemplo desfoque, causadas durante a captura da imagem [50].

Para o desenvolvimento dos algoritmos desta dissertação será utilizada a biblioteca *Open Source Computer Vision Library* (OpenCV), uma biblioteca de código aberto, amplamente utilizada para processamento e análise de imagens e compatível com Python, que oferece uma vasta gama de funções para tarefas como leitura e escrita de imagens, conversão de espaços de cor, deteção de bordas, filtragem, transformações geométricas, segmentação e análise de contornos. [51].

3.1.1 Pré-Processamento

O pré-processamento constitui a etapa inicial do processamento de imagens e tem como objetivo principal melhorar a qualidade da imagem antes de procedermos à sua análise detalhada. Esta fase visa reduzir o ruído, artefactos e elementos irrelevantes que possam comprometer a extração da informação que se pretende obter das imagens. Diversas técnicas são aplicadas para ajustar a imagem de modo a preservar a informação relevante e eliminar elementos que comprometam as etapas subsequentes, assegurando que a informação relevante seja preservada e realçada [52]. Entre as técnicas comummente utilizadas nesta fase, destacam-se:

- **Remoção de ruído**: O ruído, que se refere a distúrbios ou interferências na imagem, é um desafio frequente em imagens digitais. Filtros de suavização, como o filtro de média, o filtro de mediana, o filtro gaussiano e o filtro bilateral são frequentemente aplicados para atenuar essas distorções na tentativa de obter uma imagem mais limpa.
- **Correção de problemas de iluminação ou contraste**: A adequação da distribuição de luz na imagem é fundamental para garantir que todos os detalhes significativos sejam visíveis e bem definidos. Esta etapa envolve o ajuste da gama de intensidade da imagem, de modo a melhorar a visibilidade de áreas escuras ou claras demais, utilizando técnicas como a equalização de histograma, que distribui uniformemente os níveis de intensidade dos píxeis ao longo do intervalo de intensidades disponíveis, são amplamente utilizadas para aprimorar a visibilidade de certas regiões da imagem.

3.1.2 Segmentação

A segmentação é a etapa responsável por isolar os elementos de interesse presentes na imagem em relação ao fundo. Esta fase é fundamental, pois influencia diretamente o bom desempenho das etapas subsequentes. Caso os objetos de interesse não sejam bem destacados em relação ao fundo, poderá tornar-se difícil distingui-los nas etapas seguintes, comprometendo a eficácia das técnicas que dependem dessa separação. Diversas técnicas de segmentação são aplicadas para separar os objetos de interesse do fundo da imagem, destacando-se a limiarização e a segmentação baseada em *clusters*, como o algoritmo *k-means*, que agrupa os píxeis em diferentes classes com base em características semelhantes, permitindo distinguir objetos do fundo [53].

3.1.3 Extração de Características

Após a segmentação, a próxima etapa envolve a extração de características dos elementos previamente isolados na imagem. A extração de características visa converter a informação visual bruta em dados que possam ser utilizados para análise posterior [54]. Nesta fase, são extraídas informações quantitativas relevantes, que podem incluir:

- **Propriedades geométricas**: Medições de características geométricas dos objetos presentes na imagem, como área, perímetro e diâmetro.

- **Intensidade dos píxeis**: Refere-se aos valores de cor ou brilho dos píxeis, no caso de imagens em tons de cinza, que podem ser usados para identificar características específicas na imagem, como a intensidade de determinadas regiões.

- **Padrões**: A deteção de padrões mais complexos, como texturas ou formas geométricas reconhecíveis. Tais padrões podem ser usados para identificar ou classificar objetos presentes na imagem.

3.1.4 Análise

A análise é a etapa final no processamento de imagens, na qual os dados quantitativos extraídos são interpretados com o objetivo de responder às questões ou atingir os objetivos do estudo. Nessa fase, pode-se realizar a comparação entre diferentes amostras ou aplicar ferramentas estatísticas para identificar padrões ou tendências dentro dos dados.

Neste estudo, as imagens microscópicas capturadas são inicialmente avaliadas visualmente por um técnico, que identifica a presença das partículas contaminantes. O resultado dessa análise visual pode então ser comparado com os resultados obtidos pelo algoritmo desenvolvido, permitindo avaliar a precisão e a eficácia da solução automatizada em relação à análise humana.

O processamento de imagens é, portanto, uma sequência lógica, onde cada sub-processo prepara a imagem para o seguinte. O sucesso das etapas posteriores depende diretamente da precisão e eficácia das etapas iniciais.

3.2 Representação de Imagens Digitais

Tons de Cinza

Uma imagem digital em tons de cinza é representada como uma função bidimensional de intensidade $I(n, m)$, onde (n, m) são as coordenadas espaciais da imagem e $I(n, m)$ denota a intensidade da luz em cada píxel da imagem [48]. Neste modelo, cada píxel contém apenas um valor de intensidade, que é um único número representando a luminosidade do píxel que varia tipicamente de 0 (preto) a 255 (branco).

A imagem em tons de cinza pode ser representada de forma matricial. Cada píxel da imagem corresponde a um valor de intensidade e a imagem como um todo é representada por uma matriz cujos elementos são os valores de intensidade dos píxeis. A representação de uma imagem em tons de cinza pode ser escrita da seguinte forma:

$$I(n,m) = \begin{bmatrix} I(0,0) & I(0,1) & \cdots & I(0,M-1) \\ I(1,0) & I(1,1) & \cdots & I(1,M-1) \\ \vdots & \vdots & \ddots & \vdots \\ I(N-1,0) & I(N-1,1) & \cdots & I(N-1,M-1) \end{bmatrix}$$

onde N e M representam as dimensões espaciais da imagem, isto é, a altura e a largura, e cada $I(n,m)$ é um valor que representa a intensidade de luz do píxel na coordenada (n,m).

Coloridas

Por outro lado, no contexto deste estudo, as imagens fornecidas são coloridas. A representação das imagens coloridas é feita através de matrizes tridimensionais no formato $(N,M,3)$, onde N e M representam as dimensões espaciais da imagem (altura e largura), enquanto o valor 3 refere-se aos três canais de cor presentes no modelo de cor *Red, Blue & Green* (RGB). Cada píxel é descrito por três valores inteiros, um para cada canal de cor, variando de 0 a 255. Portanto, ao contrário da imagem em tons de cinza, onde há apenas um valor por píxel, uma imagem colorida possui três valores por píxel, que correspondem às intensidades de vermelho, verde e azul, respectivamente. Esses três valores combinados produzem a cor final de cada píxel.

O modelo de cor RGB é um dos mais comuns na representação de imagens coloridas em computação gráfica e fotografia digital. Ele é baseado na combinação de três cores primárias de luz, vermelho (*Red*), verde (*Green*) e azul (*Blue*). Cada uma dessas cores pode ter diferentes intensidades, e a combinação desses três canais cria uma gama vasta de cores visíveis.

Cada valor desses canais pode variar de 0 a 255, onde 0 significa que a cor correspondente está ausente e 255 significa que a cor está em sua intensidade máxima. Por exemplo $(255,0,0)$ representa a cor vermelha pura, pois o valor máximo está no canal vermelho enquanto que os outros 2 canais não têm influência pois são zero.

A junção desses três canais para formar uma imagem colorida pode ser representada de forma matricial. Em vez de uma única matriz, como no caso das imagens em tons de cinza, uma imagem colorida é representada por três matrizes, uma para cada canal de cor. Assim, cada píxel na imagem colorida é descrito por três valores: um para o canal vermelho, outro para o verde e outro para o azul.

As matrizes $R(n,m)$, $G(n,m)$ e $B(n,m)$ representam as intensidades de cada canal de cor para cada píxel da imagem, e a combinação dessas matrizes resulta na imagem final. A imagem colorida pode ser representada como uma matriz tridimensional $I(N,M,3)$, onde a terceira dimensão contém os valores dos três canais de cor para cada píxel:

$$I(n,m) = [R(n,m), G(n,m), B(n,m)].$$

Portanto, enquanto numa imagem em tons de cinza cada píxel é representado por um único valor $f(n, m)$, numa imagem colorida, cada píxel é representado por três valores $(R(n, m), G(n, m), B(n, m))$ que correspondem aos canais de cor vermelho, verde e azul.

3.3 Conversões de Espaços de Cor

3.3.1 Conversão para Escala de Cinza

Para reduzir a quantidade de informação presente nas imagens podemos converte-las a imagens em apenas tons de cinza usando uma transformação simples. Apesar de uma imagem em tons de cinza conter menos informação do que uma imagem colorida, a maior parte da informação relacionada com características importantes é preservada, como bordas, regiões e contornos [55]. Como neste projeto estas são as informações que queremos extrair, não sendo a cor das partículas relevante, as imagens são convertidas. A conversão para escala de cinza utiliza uma combinação ponderada dos canais RGB:

$$I_{\text{tons de cinza}}(n, m) = \alpha \cdot I_{\text{vermelho}}(n, m, r) + \beta \cdot I_{\text{verde}}(n, m, g) + \gamma \cdot I_{\text{azul}}(n, m, b) \tag{3.1}$$

sendo n, m as coordenadas do píxel na imagem, r, g, b as intensidades das componentes vermelho, verde e azul do píxel e α, β, γ coeficientes ponderados que determinam a influência de cada componente de cor na conversão. Estes valores refletem a sensibilidade do olho humano às diferentes cores pelo que os valores mais comuns para esta conversão são $\alpha \simeq 0.2989$, $\beta \simeq 0.5870$ e $\gamma \simeq 0.1140$. O coeficiente mais elevado é o correspondente à cor verde pois é a cor à qual o olho humano tem maior sensibilidade [49].

3.3.1.1 Implementação em Python

Função `cv2.imread()`

A função `cv2.imread()` lê a imagem no formato especificado e converte-a para uma matriz numérica. O OpenCV suporta vários formatos de imagem, os dados fornecidos para este estudo encontram-se nos formatos JPG, PNG e JFIF, todos suportados pelo OpenCV.

Função `cv2.cvtColor()`

A função `cv2.cvtColor()` é utilizada para converter uma imagem de um espaço de cores para outro. No código, essa função é utilizada para converter a imagem colorida, no formato RGB, para uma imagem em escala de cinza. Os parâmetros desta função são `imagem_colorida` e `cv2.COLOR_BGR2GRAY`. O parâmetro `imagem_colorida` é a imagem original que será convertida e `cv2.COLOR_BGR2GRAY` é um código de conversão que indica que a imagem deve ser convertida do espaço de cores RGB para escala de cinza por meio da fórmula ponderada 3.1 e utilizando os valores para α, β e γ os valores mais comuns indicados anteriormente. A função retorna a imagem convertida para escala de

cinza o que significa que a imagem resultante contém apenas intensidades de cinza, com valores que representam a luminosidade de cada píxel, sem informação de cor.

Exemplo de Implementação em Python

```
# Carregar a imagem
imagem_colorida = cv2.imread('imagem.jpg')

# Converter de RGB para escala de cinza
imagem_cinza = cv2.cvtColor(imagem_colorida, cv2.COLOR_BGR2GRAY)
```

3.3.2 Conversão para o Espaço de Cor *L*a*b**

A conversão para o espaço de cor *L*a*b**, também denominado *CIELAB*, também pode ser de particular interesse para a segmentação das imagens, uma vez que este espaço de cores foi desenvolvido para ser perceptualmente uniforme, aproximando-se da forma como o ser humano distingue as cores. Ao separar de forma explícita a componente de luminância, L^*, das componentes cromáticas a^*, a componente vermelho-verde, e b^*, a componente amarelo-azul, o modelo *L*a*b** permite uma análise independente das variações de iluminação. No contexto específico da deteção de partículas microscópicas em lubrificantes, esta conversão permite distinguir de forma mais clara regiões com características cromáticas semelhantes, mas que se sobrepõem em termos de intensidade no espaço RGB.

A conversão de uma imagem do espaço RGB para *L*a*b** é feita convertendo-se a imagem de RGB para *CIEXYZ*, e de seguida de *CIEXYZ* para *CIELAB*. O espaço de cor *CIEXYZ* foi um modelo matemático definido para representar cores com base na perceção humana, é um espaço de cor aditivo e triestímulo, ou seja, descreve qualquer cor visível como uma combinação ponderada de três estímulos primários: X, Y e Z. Esses primários são imaginários, ou seja não correspondem a cores reais que possamos ver ou gerar fisicamente, mas foram escolhidos de forma a garantir que todas as funções de correspondência de cor fossem não-negativas, o que facilita os cálculos e evita erros numéricos. A conversão de valores RGB para *CIEXYZ* é feita através de uma transformação linear baseada numa matriz de conversão. Esta matriz depende das características espetrais dos primários RGB utilizados, a transformação é expressa por [56]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

No caso do OpenCV, a conversão de RGB para *CIEXYZ* utiliza uma matriz de transformação linear baseada na recomendação Rec. 709 com ponto de branco D65 [57]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Para a transformação de *CIEXYZ* para *CIELAB* utiliza-se [57, 58]:

$$L^* = \begin{cases} 116 \cdot Y^{1/3} - 16 & \text{se } Y > \delta^3 \\ 903.3 \cdot Y & \text{se } Y \leq \delta^3 \end{cases} \quad \text{com } \delta = \frac{6}{29} \tag{3.2}$$

$$a^* = 500 \cdot \left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right) \tag{3.3}$$

$$b^* = 200 \cdot \left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right) \tag{3.4}$$

onde a função $f(t)$ é definida da seguinte forma:

$$f(t) = \begin{cases} t^{1/3}, & \text{se } t > \delta^3 \\ \frac{t}{3\delta^2} + \frac{4}{29}, & \text{caso contrário} \end{cases} \quad \text{com } \delta = \frac{6}{29} \tag{3.5}$$

Nesta equação, X, Y e Z representam as componentes da imagem no espaço *CIEXYZ*, enquanto X_n, Y_n e Z_n são os valores do ponto branco de referência, no caso do OpenCV o D65. Em sistemas de cores como *CIEXYZ* e *CIELAB*, o ponto branco é uma referência para o branco absoluto, define como um dispositivo deve interpretar o branco verdadeiro, representado pelo valor 1 e não pelo valor 255, como ocorre nas representações de imagem não normalizadas utilizadas em Python. O D65 corresponde a um espetro de luz com temperatura de cor de aproximadamente 6504 Kelvin, o que equivale a uma luz diurna média, não muito quente, ou seja não muito amarelada, nem muito fria, ou seja nem muito azulada, os valores do D65 no espaço *CIEXYZ* são $X_n = 0.950456$, $Y_n = 1.000$ e $Z_n = 1.088754$, [57].

3.3.2.1 Implementação em Python

Função `cv2.cvtColor()` para Espaço *L*a*b

A função `cv2.cvtColor()` também pode ser utilizada para converter imagens para o espaço de cores *L*a*b**. No código abaixo, a imagem original em RGB é convertida através de `cv2.COLOR_BGR2LAB`, que indica a transformação do espaço de cores RGB para o espaço *L*a*b**.

Exemplo de Implementação em Python

```
# Converter do espaço BGR para L*a*b*
imagem_lab = cv2.cvtColor(imagem_colorida, cv2.COLOR_BGR2LAB)
```

3.3.3 Conversão para o Espaço de Cor HSV

O espaço de cores HSV é amplamente utilizado em processamento de imagens e deteção de objetos devido às suas vantagens em relação ao espaço de cor RGB. O que o distingue deste último espaço é a separação entre informação de cor, da saturação e brilho, o que faz com que seja particularmente útil para deteção de elementos baseada em cor [59]. Tendo em consideração que os elementos de interesse neste trabalho são partículas metálicas, trabalhar com os canais da saturação e do brilho pode ser vantajoso.

Este espaço pode ser representado pela convolução de três matrizes, cada uma com propriedades distintas, ao contrário do espaço RGB, onde todas as matrizes correspondem exclusivamente à informação de cor. O matiz representado pela letra H determina as diferentes cores, como vermelho, laranja e verde, variando de 0° a 360°, sendo 0° e 360º a cor vermelha, devido à natureza cíclica do espaço HSV; dessa forma, o espetro organiza-se de maneira contínua, onde 60° corresponde ao amarelo, 120° ao verde, 180° ao ciano, 240° ao azul e 300° ao magenta. Por sua vez, a saturação, representada pela letra S, refere-se à intensidade da cor, por exemplo, o vermelho pode ser classificado em vermelho escuro ou vermelho claro, com valores que vão de 0% a 100%, sendo 100% totalmente saturado. Por fim, o valor, representado pela letra V, define a luminosidade da cor, variando de 0%, que corresponde a preto, a 100%, que corresponde a branco.

A transformação do espaço de cor RGB para o espaço de cor HSV é não linear, e as fórmulas de conversão são as seguintes [60]:

$$H = \begin{cases} \arccos\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right), & \text{se } B \leq G \\ 2\pi - \arccos\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right), & \text{se } B > G \end{cases}$$

$$S = \frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)}$$

$$V = \frac{\max(R,G,B)}{255}$$

3.3.4 Implementação em Python

Função `cv2.cvtColor()`

Tal como mencionado em 3.3.1.1, a função `cv2.cvtColor()` é utilizada para converter uma imagem de um espaço de cores para outro. No código, essa função é utilizada para converter a imagem colorida, no formato RGB para uma imagem no formato HSV. Os parâmetros desta função são `imagem_colorida` e `cv2.COLOR_BGR2HSV`, o que indica que a imagem deve ser convertida do espaço de cores RGB para o espaço de cores HSV retornando a imagem convertida para HSV.

Exemplo de Implementação em Python

```
# Converter para o espaço de cores HSV
imagem_hsv = cv2.cvtColor(imagem_colorida, cv2.COLOR_BGR2HSV)
```

3.4 Aumento do Contraste

O contraste em imagens refere-se à diferença de luminância ou cor que torna um objeto distinguível do fundo e de outros objetos, é a diferença entre as áreas mais claras e mais escuras de uma imagem, garantindo a visibilidade dos detalhes presentes nela [61]. Tendo esta definição em consideração, o primeiro passo do pré-processamento consistir em aumentar o contraste pode ser uma abordagem eficaz pois ajuda a realçar os detalhes na imagem, tornando as diferenças entre áreas claras e escuras mais pronunciadas, o que ajuda a realçar os elementos de interesse. Isso é particularmente útil em imagens microscópicas, onde pode ser necessário destacar partículas que de outra forma seriam difíceis de visualizar. Descreve-se, de seguida, algumas técnicas para o conseguir.

3.4.1 Equalização de Histograma

A equalização de histograma redistribui os L níveis ordenados de intensidade de uma imagem em escala de cinza $X_i, i = 0, ..., L-1$, de maneira uniforme, de modo a que o histograma, que corresponde à distribuição dos níveis de cinza, fique mais uniforme ou equalizado, cobrindo toda a faixa de níveis disponíveis.

Esta transformação $T(X_k)$, $k = 0, ..., L-1$ é definida como [62]:

$$T(X_k) = \lfloor (L-1) \cdot \mathrm{CDF}_X(X_k) \rfloor = \left\lfloor (L-1) \cdot \sum_{i=0}^{k} p_X(X_i) \right\rfloor,$$

onde $L = 256$ é o número de níveis de cinza, $p_X(X_i)$ é a função densidade de probabilidade empírica $p_X(X_i) = \frac{n_i}{W}$, sendo W o número total de píxeis na imagem. Por fim, $\lfloor \cdot \rfloor$ denota a parte inteira da transformação e CDF representa a função de distribuição cumulativa empírica.

3.4.1.1 Exemplo Numérico

Consideremos a seguinte imagem composta por 16 píxeis com os seguintes níveis de intensidade:

$$\begin{bmatrix} 120 & 50 & 70 & 110 \\ 50 & 110 & 60 & 80 \\ 90 & 80 & 90 & 50 \\ 90 & 100 & 60 & 90 \end{bmatrix}$$

A tabela abaixo resume os valores calculados:

Nível i	n_i	$p_X(X_i) = \frac{n_i}{16}$	$\text{CDF}_X(X_i)$	$T(i) = \lfloor 255 \cdot \text{CDF}_X(X_i) \rfloor$
50	3	0.1875	0.1875	47
60	2	0.125	0.3125	79
70	1	0.0625	0.375	95
80	2	0.125	0.50	127
90	4	0.25	0.75	190
100	1	0.0625	0.8125	206
110	2	0.125	0.9375	238
120	1	0.0625	1.00	254

Após aplicar a transformação, a nova imagem equalizada é:

$$\text{Imagem equalizada: } \begin{bmatrix} 254 & 47 & 95 & 238 \\ 47 & 238 & 79 & 127 \\ 190 & 127 & 190 & 47 \\ 190 & 206 & 79 & 190 \end{bmatrix}$$

A equalização distribui os níveis de cinza pela faixa completa $[0, 255]$, melhorando o contraste global.

3.4.1.2 Implementação em Python

Função `cv2.equalizeHist()`

A função `cv2.equalizeHist()` é utilizada para realizar equalização de histograma numa imagem em tons de cinza. Esta função redistribui os níveis de intensidade da imagem de forma a aproveitar toda a gama possível de valores e melhorar o contraste.

```
# Aplicar equalização de histograma
imagem_equalizada = cv2.equalizeHist(imagem_cinza)
```

Para ilustrar este exemplo de forma mais intuitiva temos em baixo representada a imagem original 3.1a e a imagem após a aplicação de equalização de histograma 3.1b:

3.4.2 CLAHE

O CLAHE é uma técnica que melhora o contraste de uma imagem ao aplicar a equalização de histograma mas de forma adaptativa. Diferentemente da equalização de histograma global, que redistribui de forma igual os níveis de cinza por todas as intensidades da imagem, o CLAHE ajusta o contraste localmente, dividindo a imagem numa grelha de regiões retangulares, os *tiles*, onde a equalização é realizada de forma independente [63]. Essa inovação apresenta uma melhoria significativa em relação à equalização de histograma global, onde frequentemente ocorre perda de detalhes importantes devido ao contraste

(a) Imagem original (b) Imagem Equalizada

Figura 3.1: Visualização do efeito da aplicação da equalização de histograma.

ser ajustado uniformemente em toda a imagem, o que significa que áreas muito claras, com níveis de cinza próximos ao máximo, ou muito escuras, com níveis próximos ao mínimo, podem ser esticadas de forma desproporcional. Por exemplo, em imagens com regiões distintas, como a presença de um objeto de cor clara na imagem, a aplicação da equalização de histograma pode aumentar o contraste do fundo, no entanto compromete os detalhes do objeto por ser claro. Isso acontece porque o histograma global não está limitado a uma região específica, distribuindo as intensidades de forma uniforme por toda a imagem [64].

3.4.2.1 *Clip-Limit*

O principal problema deste ajuste local é que, em regiões homogéneas, os píxeis podem ser remapeados para o valor máximo, aumentando ainda mais o ruído da imagem. Para evitar o problema, o algoritmo CLAHE limita o aumento do contraste utilizando o parâmetro *clip-limit*. Esse limite impede que a função de distribuição acumulada empírica exceda um valor certo valor, o *clip-limit*. Quando isso ocorre, os píxeis excedentes são redistribuídos igualmente entre os restantes níveis do histograma.

A fórmula para determinar o valor correspondente ao *clip-limit* é dada por [63]:

$$\beta = \frac{m \cdot n}{B \cdot \alpha}$$

onde $m \cdot n$ corresponde ao número de píxeis no respetivo *tile*, B o número de classes (*bins*) no histograma e α a percentagem máxima de píxeis permitida em cada classe (*bin*).

3.4.2.2 Interpolação Bilinear

Para cada *tile* é construído um histograma de intensidades de píxeis a partir do qual se calcula a função de distribuição acumulada definida como:

$$CDF(s) = \sum_{t=0}^{s} \frac{h(t)}{N}$$

onde $h(t)$ é a contagem de píxeis com intensidade t, N é o número total de píxeis no *tile* e s é o valor de intensidade atual. Esta função é usada para remapear os valores de intensidade dos píxeis. Posteriormente é utilizada interpolação bilinear para suavizar as transições entre blocos adjacentes. Para um píxel P com valor s e posição (i, j), o valor mapeado s' é dado por [63]:

$$s' = (1-y)((1-x)F_A(s) + x \cdot F_B(s)) + y((1-x) \cdot F_C(s) + x \cdot F_D(s)) \tag{3.6}$$

onde A, B, C e D são os pontos centrais dos blocos adjacentes superior esquerdo, superior direito, inferior esquerdo e inferior direito respetivamente e F_A, F_B, F_C e F_D são funções de mapeamento destas regiões calculadas baseado em 3.6. Essas funções relacionam os níveis de cinza originais dos píxeis dentro do bloco com os novos níveis de cinza equalizados, de acordo com o histograma ajustado e o *clip-limit*. Tal como podemos observar no esquema 3.2, x é a distância normalizada entre P e o segmento $\overline{AC}$ formado pelos centros blocos adjacentes superiores e y a distância normalizada entre P e o segmento $\overline{AB}$ formado pelo centro bloco adjacente superior esquerdo e inferior esquerdo [65].

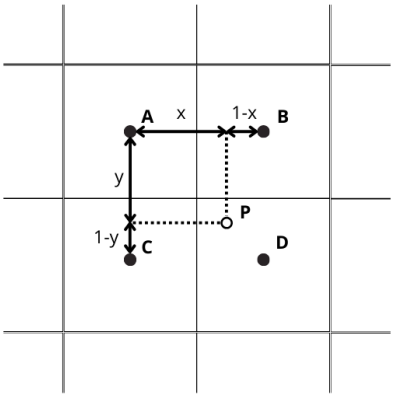


Figura 3.2: Esquema da interpolação bilinear.

3.4.2.3 Implementação em Python

Função `cv2.createCLAHE()`

A função `cv2.createCLAHE()` é utilizada para criar as condições para aplicar técnica de CLAHE a uma imagem. Os dois parâmetros principais desta função são `clipLimit` e `tileGridSize`, o parâmetro `clipLimit` define o limite máximo para a amplificação do

contraste (β), um valor maior permite mais contraste, mas pode amplificar o ruído. Por sua vez, o parâmetro `tileGridSize` especifica o tamanho dos *tiles* onde a equalização será aplicada.

O resultado da aplicação do CLAHE é uma imagem com o contraste aprimorado. A função `apply()` é utilizada posteriormente para aplicar o CLAHE à imagem em escala de cinza.

Exemplo de Implementação em Python

```
# Criar o objeto CLAHE
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

# Aplicar o objeto CLAHE à imagem em escala de cinza
imagem_contraste = clahe.apply(imagem_cinza)
```

Para ilustrar este exemplo de forma mais intuitiva temos em baixo representada a imagem original 3.3a e a imagem após a aplicação do algoritmo CLAHE 3.3b utilizando um *clip-limit* de 2 e *tiles* de tamanho 8×8:

(a) Imagem original (b) Imagem após aplicação de CLAHE

Figura 3.3: Visualização do efeito da aplicação de CLAHE.

3.5 Filtros de Suavização

O objetivo da suavização é reduzir o ruído presente na imagem sendo frequentemente realizada utilizando filtros que aplicam uma operação de convolução para atenuar as variações abruptas entre intensidades de píxeis. Sendo que a suavização acaba por desfocar a imagem, deve vir após o aumento de contraste, assim, haverá menor probabilidade de alguns detalhes serem perdidos neste processo.

3.5.1 Filtro de Média

O filtro de média pode ser interpretado como uma operação de convolução, em que o valor de cada píxel na imagem filtrada é determinado pela soma ponderada dos valores dos píxeis dentro de uma janela. Essa soma é ponderada com um *kernel* cujos valores são iguais. Para uma janela $l \times l$, o *kernel* utilizado é:

$$K(m,n) = \frac{1}{l \times l} \quad \text{, para todo } m, n.$$

Assim, o valor do píxel suavizado na posição (i, j) é calculado como:

$$I'(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} I(i+m, j+n) \cdot K(m,n),$$

com $k = \frac{l-1}{2}$, $l \times l$ o tamanho da janela e I a intensidade inicial do píxel.

3.5.1.1 Exemplo Numérico

Para uma janela 3×3, o *kernel* correspondente é:

$$\text{K=} \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}.$$

Considerando um píxel de coordenadas (i, j) com intensidade 22 cujas intensidades dos píxeis vizinhos são dadas pela janela seguinte:

$$\begin{bmatrix} 12 & 15 & 10 \\ 20 & 22 & 18 \\ 25 & 24 & 21 \end{bmatrix},$$

O cálculo ponto a ponto do píxel resultante é dado por::

$$\begin{bmatrix} 12 \cdot \frac{1}{9} & 15 \cdot \frac{1}{9} & 10 \cdot \frac{1}{9} \\ 20 \cdot \frac{1}{9} & 22 \cdot \frac{1}{9} & 18 \cdot \frac{1}{9} \\ 25 \cdot \frac{1}{9} & 24 \cdot \frac{1}{9} & 21 \cdot \frac{1}{9} \end{bmatrix} = \begin{bmatrix} 1.33 & 1.67 & 1.11 \\ 2.22 & 2.44 & 2.00 \\ 2.78 & 2.67 & 2.33 \end{bmatrix}.$$

Somando todos os valores:

$$\text{I'(i,j)=1.33 + 1.67 + 1.11 + 2.22 + 2.44 + 2.00 + 2.78 + 2.67 + 2.33 = 18.56.}$$

O píxel central p da janela será então substituído pelo valor 18.56, realizando-se o mesmo processo para os restantes píxeis. Para lidar com o problema de não existirem valores em redor das bordas e cantos para aplicar o *kernel* a solução padrão utilizada pela biblioteca OpenCV é a técnica de *border reflection*, na qual os valores ao longo das bordas são refletidos. Obtém-se a seguinte matriz:

$$\text{Matriz com border reflection: } \begin{bmatrix} 22 & 20 & 22 & 18 & 22 \\ 15 & 12 & 15 & 10 & 15 \\ 22 & 20 & 22 & 18 & 22 \\ 24 & 25 & 24 & 21 & 24 \\ 22 & 20 & 22 & 18 & 22 \end{bmatrix}$$

Após a realização do processo a todos píxeis obtém-se a seguinte matriz:

$$\text{Matriz suavizada: } \begin{bmatrix} 15.44 & 16.33 & 12.67 \\ 19.11 & 21.22 & 17.33 \\ 18.33 & 20.44 & 16.22 \end{bmatrix}$$

Após calcular o valor suavizado do píxel central, a janela é deslocada para a posição seguinte, repetindo-se o processo para todos os píxeis da imagem. Importa salientar que os novos valores são obtidos a partir da imagem original, garantindo que cada píxel da imagem resultante é calculado de forma independente.

3.5.1.2 Implementação em Python

Função `cv2.blur()`

A função `cv2.blur()` é utilizada para aplicar um filtro de média a uma imagem. Este filtro reduz o ruído da imagem, calculando a média dos píxeis numa janela retangular definida pelo utilizador. No código abaixo, o parâmetro `(l,l)` especifica o tamanho da janela do filtro, neste caso um quadrado de 9×9 píxeis.

A função retorna a imagem suavizada, que apresenta menos ruído em comparação com a imagem original, mas pode também sofrer ligeira perda de detalhes, dependendo do tamanho da janela utilizada.

Exemplo de Implementação em Python

```
# Aplicar o filtro de média
imagem_suavizada = cv2.blur(imagem_cinza, (9, 9))
```

Para ilustrar este exemplo de forma mais intuitiva temos em baixo representada a imagem original 3.4a e a imagem após a aplicação do filtro de média 3.4b *kernel* de tamanho 9×9:

(a) Imagem original

(b) Imagem com filtro de média

Figura 3.4: Visualização do efeito da aplicação do filtro de média.

3.5.2 Filtro da Mediana

O filtro da mediana é um filtro não linear utilizado para suavizar imagens, que funciona processando a imagem de forma local, cada janela é centrada num píxel, e a substituição do valor ocorre apenas dentro da região da janela e para o píxel à qual a janela foi aplicada. Após esse cálculo, a janela é movida para o próximo píxel, repetindo o mesmo processo para toda a imagem. Ao contrário do filtro de média, que utiliza uma média ponderada dos valores dos píxeis na vizinhança, o filtro da mediana seleciona o valor mediano dos píxeis dentro de uma janela de tamanho $l \times l$.

A fórmula para calcular o valor do píxel suavizado na posição (i, j) é dada por:

$$I'(i,j) = \text{Mediana}\left\{I(i+m, j+n) : -k \leq m, n \leq k\right\},$$

onde $I'(i,j)$ é o valor do píxel suavizado na posição (i,j), $I(i+m, j+n)$ são os valores dos píxeis na janela de tamanho $l \times l$, e $k = \frac{l-1}{2}$ define a posição central da janela.

3.5.2.1 Exemplo Numérico

Considerando a escolha de uma janela 3×3, posicionada sobre um píxel de intensidade 22, com a seguinte matriz de valores de intensidade:

$$\text{Janela: } \begin{bmatrix} 12 & 15 & 10 \\ 20 & 22 & 18 \\ 25 & 24 & 21 \end{bmatrix}.$$

Para calcular o valor da mediana, todos os valores da janela são ordenados:

$$\{10, 12, 15, 18, 20, 21, 22, 24, 25\}.$$

A mediana é o valor central da lista ordenada que neste caso é 20. Assim, o valor do píxel central da janela, originalmente 22, será substituído pelo valor 20. Após essa substituição, a janela é deslocada para o píxel seguinte, e o mesmo processo é repetido, utilizando os valores da matriz de intensidades original.

3.5.2.2 Implementação em Python

Função `cv2.medianBlur()`
A função `cv2.medianBlur()` é utilizada para aplicar um filtro da mediana a uma imagem. No exemplo de código fornecido abaixo, o parâmetro `imagem_cinza` é a imagem de entrada, já em escala de cinza, enquanto o parâmetro 9 especifica o tamanho da janela como $l \times l$, neste caso 9×9. O filtro da mediana funciona substituindo o valor de cada píxel pelo valor mediano dos píxeis dentro da janela, ordenados em termos de intensidade. A função retorna a imagem processada com o filtro da mediana aplicado.

Exemplo de Implementação em Python

```
# Aplicar o filtro de mediana a uma imagem
imagem_suavizada = cv2.medianBlur(imagem_cinza, 9)
```

Para ilustrar este exemplo de forma mais intuitiva temos em baixo representada a imagem original 3.5a e a imagem após a aplicação do filtro de média 3.5b utilizando um *kernel* de tamanho 9×9:

3.5.3 Filtro Gaussiano

Entre os filtros mais utilizados, destaca-se o filtro Gaussiano que é utilizado para suavizar imagens, reduzindo ruídos enquanto preserva variações suaves. A fórmula matemática do filtro Gaussiano bidimensional é dada por:

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

onde σ é o desvio padrão que controla o grau de suavização. Este filtro é isotrópico, ou seja, aplica suavização uniforme em todas as direções.

A função Gaussiana é separável [66], o que significa que a convolução da imagem com o *kernel* Gaussiano pode ser decomposta em duas convoluções unidimensionais, uma em cada direção. Ou seja:

$$G(x, y) = G_x(x) \cdot G_y(y),$$

onde:

$$G_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right), \quad G_y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{y^2}{2\sigma^2}\right).$$

(a) Imagem original

(b) Imagem com filtro de mediana

Figura 3.5: Visualização do efeito da aplicação do filtro de mediana.

Essa separabilidade reduz a complexidade computacional de $O(n^2)$ para $O(2n)$. O que significa que o número de operações deixa de crescer de forma quadrática em relação ao tamanho da entrada, número de píxeis de uma imagem, e passa a crescer de forma linear

Para garantir que o filtro Gaussiano preserve a intensidade total da imagem, o integral de $G(x, y)$ ao longo de todo o espaço deve ser igual a 1.

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x,y)\,dx\,dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)dx\,dy = 1.$$

Usando a separabilidade da função Gaussiana, podemos dividir o integral em dois termos unidimensionais:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x,y)\,dx\,dy = \left(\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right)dx\right)^2.$$

O valor do integral unidimensional é conhecido, por isso podemos tirar o seguinte resultado:

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right)dx = 1 \Rightarrow \left(\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right)dx\right)^2 = 1$$

o que valida que o integral da função Gaussiana bidimensional é 1.

Este resultado confirma que o filtro Gaussiano preserva a intensidade global da imagem, como a soma dos pesos do filtro é igual a 1, o valor total de todos os píxeis na imagem, ou seja, a intensidade global permanece inalterada. A função Gaussiana redistribui a intensidade localmente, mas sem aumentar ou reduzir intensidade.

3.5.3.1 Cálculo do *kernel* Gaussiano

Na prática, este filtro é implementado em imagens digitais como uma matriz discreta. A fórmula para o *kernel* é obtida determinando $G(x, y)$ em coordenadas discretas e, em seguida, normalizando os valores. A fórmula para o *kernel* Gaussiano discretizado é dada por:

$$K(m, n) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{m^2 + n^2}{2\sigma^2}\right),$$

onde $m, n \in \{-k, ..., 0, ..., k\}$ e são as coordenadas relativas ao centro do *kernel*. Por exemplo, para uma matriz 3×3, os valores de (m, n) relativos ao centro do *kernel* são:

$$\begin{bmatrix} (-1,-1) & (0,-1) & (1,-1) \\ (-1,0) & (0,0) & (1,0) \\ (-1,1) & (0,1) & (1,1) \end{bmatrix}.$$

Além disso, o intervalo de valores para os quais os pesos são calculados é comummente definido como $[-3\sigma, 3\sigma]$. Isso cobre aproximadamente 99% da distribuição Gaussiana, sendo os restantes valores tão pequenos e próximos de zero que a maior parte da função é considerada no cálculo, sem a necessidade de um *kernel* excessivamente grande [66]. Após o cálculo do *kernel*, é importante normaliza-lo:

$$\sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m, n) = 1.$$

O *kernel* Gaussiano é normalizado para garantir que, após a aplicação do filtro em todos os píxeis, a intensidade média da imagem resultante deve ser igual à intensidade média da imagem original, preservando assim a intensidade global desta última, garantido que não fica nem mais clara nem mais escura [48].

3.5.3.2 Convolução com o *kernel* Gaussiano

A aplicação do filtro envolve a convolução da imagem com um *kernel* Gaussiano discretizado. O valor do píxel suavizado na posição (i, j) é dado por:

$$I'(i, j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m, n) \cdot I(i + m, j + n),$$

onde $K(m, n)$ é o *kernel* Gaussiano e $I(i + m, j + n)$ são os píxeis vizinhos.

Influência de σ

Como mencionado, o parâmetro σ controla o grau de suavização aplicado à imagem. Quando σ é pequeno, o filtro concentra a maior parte do seu peso em torno do ponto central, resultando numa suavização leve, afetando apenas os píxeis muito próximos do centro do *kernel*. Por outro lado, quando σ é maior o filtro considera contribuições de

píxeis mais distantes, isto resulta numa maior suavização e consequente redução de ruído. Contudo, uma suavização excessiva pode levar à perda de detalhes importantes, como bordas de partículas.

3.5.3.3 Exemplo Numérico

Para $\sigma = 1.0$ e um *kernel* 3×3:

$$K = \begin{bmatrix} 0.0585 & 0.0965 & 0.0585 \\ 0.0965 & 0.1592 & 0.0965 \\ 0.0585 & 0.0965 & 0.0585 \end{bmatrix}$$

A soma dos valores deste *kernel* é dada por:

$$0.0585 + 0.0965 + 0.0585 + 0.0965 + 0.1592 + 0.0965 + 0.0585 + 0.0965 + 0.0585 = 0.7792$$

Para normalizar o *kernel*, dividimos cada elemento pelo valor 0.7792. O *kernel* normalizado será:

$$K = \begin{bmatrix} 0.0751 & 0.1239 & 0.0751 \\ 0.1239 & 0.2043 & 0.1239 \\ 0.0751 & 0.1239 & 0.0751 \end{bmatrix}$$

Suponhamos agora a seguinte janela 3×3 centrada num píxel de intensidade 22:

$$\text{Janela original: } \begin{bmatrix} 12 & 15 & 10 \\ 20 & 22 & 18 \\ 25 & 24 & 21 \end{bmatrix}.$$

O valor suavizado do píxel central é obtido pela convolução da janela de intensidades com o *kernel* normalizado. O que resulta em:

$$\begin{bmatrix} 12 \cdot 0.0751 & 15 \cdot 0.1239 & 10 \cdot 0.0751 \\ 20 \cdot 0.1239 & 22 \cdot 0.2043 & 18 \cdot 0.1239 \\ 25 \cdot 0.0751 & 24 \cdot 0.1239 & 21 \cdot 0.0751 \end{bmatrix} = \begin{bmatrix} 0.901 & 1.859 & 0.751 \\ 2.478 & 4.475 & 2.230 \\ 1.878 & 2.974 & 1.577 \end{bmatrix}.$$

Somando todos os valores ponderados:

$$I' = 0.901 + 1.859 + 0.751 + 2.478 + 4.475 + 2.230 + 1.878 + 2.974 + 1.577 = 19.12.$$

Assim, o valor suavizado do píxel central passa de 22 para aproximadamente 19.12. Este processo é repetido para todos os píxeis da imagem, utilizando sempre os valores de intensidade originais.

3.5.3.4 Exemplo de Implementação em Python

Função `cv2.GaussianBlur()`

A função `cv2.GaussianBlur()` é utilizada para aplicar um filtro Gaussiano a uma imagem. No código abaixo, o parâmetro `(1, 1)` especifica o tamanho do *kernel* Gaussiano utilizado e o último parâmetro σ representa o desvio padrão que vai ser utilizado na função Gaussiana.

No exemplo, é definido $\sigma = 0$, o que significa que o OpenCV calculará automaticamente o valor ideal de σ com base no tamanho do *kernel*, escolhido de forma empírica. O σ é calculado através da expressão [67]:

$$\sigma = 0.3 \cdot ((ksize - 1) \cdot 0.5 - 1) + 0.8$$

onde *ksize* é referente ao tamanho do *kernel*.

A função retorna a imagem suavizada. O filtro Gaussiano, ao contrário do filtro de média, aplica pesos diferentes aos píxeis vizinhos, com base na distribuição Gaussiana, permitindo uma suavização mais natural.

Exemplo de Implementação em Python

```
# Aplicar o filtro Gaussiano
imagem_suavizada = cv2.GaussianBlur(imagem_cinza, (9, 9), 0)
```

Para ilustrar este exemplo de forma mais intuitiva temos em baixo representada a imagem original 3.6a e a imagem após a aplicação do filtro gaussiano 3.6b utilizando um *kernel* de tamanho 9×9 e σ calculado automaticamente:

3.5.4 Filtro Bilateral

Apesar do filtro Gaussiano preservar variações suaves, as bordas das partículas podem apresentar grandes variações de intensidade em relação ao fundo da imagem. O filtro Bilateral, além de suavizar a imagem para reduzir o ruído, também preserva as bordas pois tem em consideração tanto a proximidade espacial dos píxeis quanto a similaridade de intensidade entre eles [48]. O filtro Bilateral é composto pela combinação de duas funções Gaussianas, uma no domínio espacial e outra no domínio da intensidade.

A distância euclidiana entre dois pontos $P_1, P_2 \in \mathbb{R}^n$ definidos por $P_1 = (x_1, x_2, \ldots, x_n)$ e $P_2 = (y_1, y_2, \ldots, y_n)$ é dada por [68]:

$$d(P_1, P_2) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2} \tag{3.7}$$

No caso de um plano bidimensional, como é o caso das imagens em tons de cinza, em que os pontos se reduzem a $P_1 = (x_1, x_2)$ e $P_2 = (y_1, y_2)$ pode-se resumir a:

(a) Imagem original

(b) Imagem com filtro Gaussiano

Figura 3.6: Visualização do efeito da aplicação do filtro Gaussiano.

$$d(P_1, P_2) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2} \tag{3.8}$$

O valor transformado da intensidade do píxel, após a aplicação do filtro bilateral, é dado por:

$$I'(i,j) = \frac{1}{W(i,j)} \sum_{(m,n)\in S} \left(G_{\sigma_s} \sqrt{(i-m)^2 + (j-n)^2} \right) \cdot G_{\sigma_r}(|I(i,j) - I(m,n)|) \cdot I(m,n)$$

onde:

- (i, j) são as coordenadas do píxel central tendo em conta o diâmetro da janela escolhida.
- (m, n) são as coordenadas dos píxeis vizinhos dentro de uma janela espacial S,
- $G_{\sigma_s}\left(\sqrt{(i-m)^2 + (j-n)^2}\right) = \exp\left(-\frac{(i-m)^2+(j-n)^2}{2\sigma_s^2}\right)$ é a função de proximidade espacial baseada na distância euclidiana definida em 3.8,
- $G_{\sigma_r}(|I(i,j) - I(m,n)|) = \exp\left(-\frac{|I(i,j)-I(m,n)|^2}{2\sigma_r^2}\right)$ é a função de similaridade de intensidade (baseada na diferença de intensidade entre $I(i,j)$ e $I(m,n)$),
- $W(i,j)$ é a constante de normalização dada por:

$$W(i,j) = \sum_{(m,n)\in S} G_{\sigma_s}(\sqrt{(i-m)^2 + (j-n)^2}) \cdot G_{\sigma_r}(|I(i,j) - I(m,n)|).$$

3.5.4.1 Exemplo Numérico

Consideremos uma pequena janela 3×3 em torno do píxel central de coordenadas arbitrárias (i, j) com intensidade igual a 22, com os seguintes valores de intensidade $I(m, n)$ para os píxeis em redor deste:

$$\begin{bmatrix} 12 & 15 & 10 \\ 20 & 22 & 18 \\ 25 & 24 & 21 \end{bmatrix}.$$

Aqui assumimos que o píxel central (i, j) tem uma intensidade $I(i, j) = 98$. Vamos considerar para este exemplo os valores $\sigma_s = 1$ e $\sigma_r = 2$. Com estas constantes vamos calcular a função de proximidade espacial G_{σ_s} e a função de similaridade de intensidade G_{σ_r} para o píxel $(i-1, j-1)$:

$$G_{\sigma_s}(\sqrt{(i-(i-1))^2 + (j-(j-1))^2}) = \exp\left(-\frac{(i-(i-1))^2 + (j-(j-1))^2}{2\sigma_s^2}\right) = \exp\left(-\frac{1^2+1^2}{2(1)^2}\right) =$$

$$= \exp\left(-\frac{2}{2}\right) = \exp(-1) \approx 0.3679.$$

$$G_{\sigma_r}(|I(i,j) - I(i-1,j-1)|) = G_{\sigma_r}(|22-12|) = G_{\sigma_r}(10) = \exp\left(-\frac{10^2}{2\sigma_r^2}\right) = \exp\left(-\frac{100}{2(2)^2}\right) =$$

$$= \exp\left(-\frac{100}{8}\right) = \exp(-0.5) \approx 3.72 \cdot 10^{-6}.$$

O peso, tendo em conta apenas o píxel $(i-1, j-1)$, é calculado como:

$$W(i,j) = G_{\sigma_s}(\sqrt{(i-(i-1))^2 + (j-(j-1))^2}) \cdot G_{\sigma_r}(|I(i,j) - I(i-1,j-1)|) \approx 0.3679 \cdot 3.72 \cdot 10^{-6} \approx 1.37 \cdot 10^{-6}.$$

Agora, para obtermos o peso tendo também em consideração os restantes píxeis dentro da janela, somamos os produtos ponderados de cada píxel e normalizamos dividindo pelo peso total.

Janela de píxeis com o respetivo peso

$$\begin{bmatrix} (12, 0.3679 \cdot 3.72 \cdot 10^{-6}) & (15, 0.6065 \cdot 0.002) & (10, 0.3679 \cdot 1.52 \cdot 10^{-8}) \\ (20, 0.6065 \cdot 0.607) & (22, 1.0 \cdot 1.0) & (18, 0.6065 \cdot 0.1353) \\ (25, 0.3679 \cdot 0.3247) & (24, 0.6065 \cdot 0.607) & (21, 0.3679 \cdot 0.8825) \end{bmatrix}$$

Peso total e Intensidade ajustada

$$W = \sum G_{\sigma_s} \cdot G_{\sigma_r} \approx 2.262,$$

$$I'(i,j) = \frac{1}{W} \sum_{(m,n) \in S} G_{\sigma_s}(\sqrt{(i-m)^2 + (j-n)^2}) \cdot G_{\sigma_r}(|I(i,j) - I(m,n)|) \cdot I(m,n).$$

Substituindo os valores:

$$I'(i,j) \approx \frac{1}{2.262}(12 \cdot 1.37e{-}6 + 15 \cdot 0.00121 + 10 \cdot 5.59e{-}9 + 20 \cdot 0.368 + 22 \cdot 1.0 + 18 \cdot 0.082 + 25 \cdot 0.119 + 24 \cdot$$

Neste exemplo, o novo valor de intensidade no píxel central (i, j), ou seja, no píxel de intensidade 22, é substituído por um valor aproximado a 21.44.

3.5.4.2 Implementação em Python

Função `cv2.bilateralFilter()`

A função `cv2.bilateralFilter()` é utilizada para aplicar um filtro bilateral a uma imagem.

No código fornecido abaxo, a variável `imagem_cinza` é a imagem de entrada, e o segundo parâmetro, $d = 9$, especifica o diâmetro da janela que será usada para filtrar os píxeis ao redor de cada píxel central.

Os parâmetros *sigmaColor* e *sigmaSpace* controlam, respectivamente, o grau de suavização nas dimensões de brilho e no espaço, distância entre os píxels. Quanto maiores esses valores, mais forte será o efeito de suavização.

A função retorna a imagem suavizada, que apresenta menos ruído.

Exemplo de Implementação em Python

```
# Aplicar o filtro bilateral
imagem_suavizada = cv2.bilateralFilter(imagem_cinza, d=9, sigmaColor, sigmaSpace)
```

No exemplo visual abaixo conseguimos ver o efeito de diferentes valores para *sigmaColor* e *sigmaSpace* que neste caso é representado por σ_d:

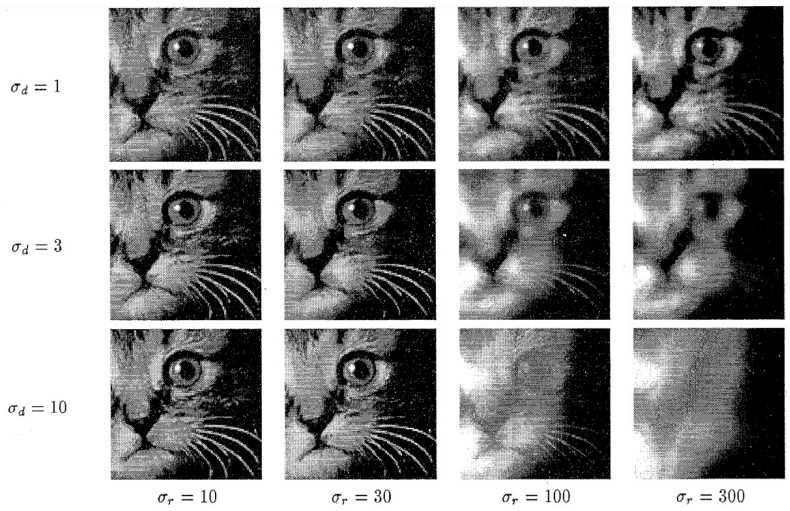


Figura 3.7: Diferentes valores para os parâmetros de domínio espacial e de intensidade [69].

3.6 Avaliação Quantitativa dos Filtros

A utilização de métricas quantitativas é fundamental, pois garante uma avaliação objetiva, reprodutível e comparável entre diferentes métodos, ultrapassando as limitações da simples inspeção visual. Além disso, permite assegurar que o processamento não introduz degradação excessiva, evitando perda de informação relevante ou o inverso, a criação de artefactos indesejados que inicialmente não estavam presentes na imagem. No capítulo 2 é brevemente referido que nos artigos [25, 37] são utilizadas 2 métricas para comparar técnicas, o MSE e PSNR.

3.6.1 MSE

A métrica MSE permite quantificar a diferença ao quadrado, tomada em média, entre a intensidade dos píxeis da imagem original e os da imagem depois de pré-processada. Sendo I a imagem original e K a imagem filtrada, ambas com dimensões $m \times n$, o MSE é definido como [70]:

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[I(i,j) - K(i,j) \right]^2 \tag{3.9}$$

Quanto menor for o valor de MSE, maior será a semelhança entre as duas imagens.

3.6.2 PSNR

A métrica PSNR mede a qualidade da imagem processada em relação à original, sendo inversamente proporcional ao MSE. Assume-se que os valores de intensidade variam entre 0 e MAX_I, o valor máximo de intensidade na imagem. A PSNR é definida como [70]:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right) \tag{3.10}$$

Valores mais elevados de PSNR indicam que há menos alterações na imagem filtrada em relação à original. Para imagens em tons de cinza, onde o valor de MAX_I é 255, o PSNR deve estar entre 30 e 50 dB[1] para evitar uma degradação percetível da qualidade visual [70]. No caso do PSNR, indica a razão entre a potência máxima da imagem e a potência do erro de reconstrução.

3.6.3 *Structural Similarity Index Method* (SSIM)

Além das métricas 3.6.1 e 3.6.2 o artigo [70] também faz referência à métrica SSIM. Esta também avalia a semelhança estrutural entre duas imagens, no entanto reflete melhor a perceção visual humana, pois em vez de comparar os píxeis diretamente como o MSE,

[1] *dB* significa *decibel*, uma unidade logarítmica que expressa a razão entre dois valores de potência ou intensidade.

considera as componentes luminância, contraste e estrutura. A expressão para calcular esta métrica é dada por [70]:

$$\text{SSIM}(I, K) = \frac{(2\mu_I\mu_K + C_1)(2\sigma_{IK} + C_2)}{(\mu_I^2 + \mu_K^2 + C_1)(\sigma_I^2 + \sigma_K^2 + C_2)} \tag{3.11}$$

onde μ_I e μ_K são as intensidades médias locais das imagens I e K, ou seja, da imagem original e da imagem após o pré processamento; σ_I^2 e σ_K^2 são as variâncias locais e σ_{IK} é a covariância local entre I e K. Por sua vez, $C_1 = (0.01 \cdot \text{MAX}_I)^2$ e $C_2 = (0.03 \cdot \text{MAX}_I)^2$, C_1 e C_2 são pequenas constantes utilizadas para garantir a estabilidade numérica, evitando divisões por zero [71].

O valor da SSIM varia entre 0 e 1, sendo que um SSIM mais próximo de 1 significa que as imagens, mesmo após ser realizado o pré processamento, ainda são idênticas não havendo uma degradação significativa da imagem original, enquanto que valores abaixo de 0.9 indicam perdas visíveis na estrutura da imagem [71].

3.7 Segmentação

A segmentação tem como objetivo identificar e isolar objetos de interesse, sendo que a deteção de bordas faz parte dessa etapa, fornecendo informações estruturais essenciais para a posterior extração das delimitações exteriores das estruturas de interesse para este projeto, as partículas. Após o pré-processamento, avalia-se visualmente a capacidade do algoritmo em identificar corretamente os contornos, garantindo que nenhuma interferência do ruído de fundo seja interpretada como uma borda. Para as imagens convertidas para o espaço de cores em tons de cinza, adotou-se uma abordagem preliminar baseada na deteção de bordas, sendo o algoritmo de Canny o método tradicional mais eficiente para esse propósito [72]. O método de Canny destaca áreas de transição abrupta na intensidade da imagem, identificando como bordas as áreas que correspondem às variações mais acentuadas, sendo estas definidas como todas aquelas cujo nível de intensidade de brilho é superior a um determinado valor. Este processo combina diversas técnicas, o fluxograma 3.8 descreve o processo geral do método de Canny [73], aplicado a este projeto.

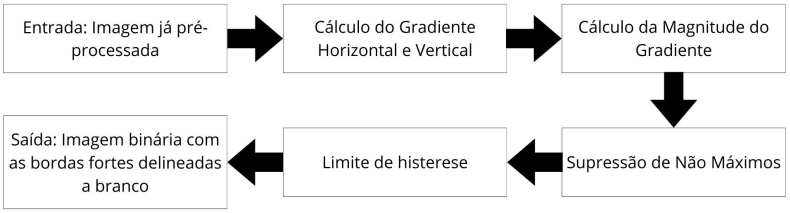


Figura 3.8: Fluxograma do método de Canny.

Por outro lado, quando a imagem é convertida para o espaço de cores *L*a*b**, a segmentação requer uma abordagem diferente. Tal como descrito em 3.3.2 este espaço foi desenvolvido para se aproximar da forma como o olho humano percebe a cor, separando a informação de luminosidade da informação cromática, isso permite uma distinção mais clara entre elementos de interesse e fundo, mesmo quando as bordas não são bem definidas em termos de intensidade.

Assim, em vez de recorrer à deteção de bordas utilizando Canny, visto que esta ferramenta apenas é aplicada em imagens em tons de cinza e depende das transições abruptas de intensidade, opta-se por aplicar um algoritmo de *clustering*, o *K-means*, que agrupa os píxeis com base nas suas características nos três canais do espaço *L*a*b**. Este método permite identificar regiões com aparência visual semelhante, agrupando píxeis com base na sua semelhança visual, neste caso partículas e fundo, de forma coerente com a perceção humana.

3.7.1 Deteção de Bordas utilizando Canny

3.7.1.1 Cálculo do Gradiente

O cálculo do gradiente é o primeiro passo no método de Canny. Como o gradiente de uma imagem indica a taxa de variação da intensidade em diferentes direções, este passo tem como objetivo identificar as áreas de transição abrupta na intensidade da imagem, que correspondem às bordas. Para calcular o gradiente, utilizam-se as derivadas parciais da imagem nas direções x e y, representadas por G_x e G_y:

$$G_x = \frac{\partial I}{\partial x}, \quad G_y = \frac{\partial I}{\partial y}$$

No entanto, como as imagens digitais são representadas como matrizes, ou seja, funções discretas e não contínuas, não é possível calcular derivadas no sentido clássico, sendo necessário recorrer-se a aproximações numéricas chamadas derivadas parciais discretas, que substituem a variação infinitesimal por diferenças finitas entre valores de intensidade

de píxeis vizinhos. Para o cálculo dessas aproximações, os operadores de Sobel constituem uma abordagem utilizada, o OpenCV é uma das bibliotecas de processamento de imagem que os implementa para estimar gradientes direcionais. Estes operadores aplicam um *kernel* convolucional sobre imagem para estimar as taxas de variação, na direção x e y, da intensidade, além disso, incorporam uma componente de suavização porque os valores do *kernel* são ponderados, quanto mais próximos os píxeis estiverem do centro mais influência têm no cálculo do valor final de cada píxel. Abaixo temos os *kernels* de tamanho 3×3, onde podemos observar que um *kernel* é igual ao outro com uma rotação de 90°, além disso ambos apresentam uma linha central de zeros, o que reforça a sensibilidade direcional, o *kernel* do gradiente horizontal possui zeros na coluna central, enquanto o vertical possui zeros na linha central. [74]:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Para cada píxel (i, j) da imagem I, aplica-se convolução com os *kernels* de Sobel:

$$G_x(i,j) = \sum_{m=0}^{2}\sum_{n=0}^{2} G_x(m,n)\cdot I(i+m-1, j+n-1), \quad G_y(i,j) = \sum_{m=0}^{2}\sum_{n=0}^{2} G_y(m,n)\cdot I(i+m-1, j+n-1),$$

A magnitude do gradiente em cada píxel (i, j), $G(i, j)$, que representa a intensidade da variação de brilho no píxel, é calculada combinando as derivadas nas direções x e y naquele ponto:

$$G(i,j) = \sqrt{(G_x(i,j))^2 + (G_y(i,j))^2}.$$

3.7.1.2 Supressão de Não Máximos:

Neste próximo passo do processo o objetivo é refinar a deteção de bordas previamente realizada, garantindo que apenas os píxeis mais significativos ao longo das bordas sejam preservados, com o intuito de ter linhas finas e bem definidas. Na etapa anterior, verificou-se que o gradiente indica a variação de intensidade da imagem, destacando as transições abruptas entre áreas claras e escuras, além disso, cada píxel possui uma direção associada ao gradiente, que indica para onde ocorre a maior variação de intensidade. Nesta etapa, compara-se a magnitude do gradiente de cada píxel com a dos seus vizinhos ao longo da direção do gradiente. Por exemplo, se um píxel possui um gradiente direcionado para a direita, isso indica que a maior variação de brilho ocorre nessa direção. A direção do gradiente θ também é determinada pelas derivadas parciais G_x e G_y:

$$\theta(i,j) = \arctan\left(\frac{G_y(i,j)}{G_x(i,j)}\right),$$

Com base na direção do gradiente, toma-se a seguinte decisão:

1. Se a magnitude do gradiente do píxel for superior à magnitude do gradiente dos píxeis adjacentes, na direção do gradiente, o píxel mantém a sua magnitude.

2. Caso contrário, a magnitude do gradiente do píxel é definida como 0.

Assim, geralmente são examinados três píxeis dentro de uma vizinhança 3×3 ao redor do píxel (i, j), por exemplo, se $\theta = 0°$, então os píxeis $(i + 1, j)$, (i, j) e $(i - 1, yj$ são comparados e posteriormente é tomada a decisão mencionada acima.

3.7.1.3 Limiarização com Histerese:

Após a supressão de não máximos, as bordas obtidas podem conter tanto respostas fortes como respostas fracas, ou seja, píxeis que representam bordas bem definidas e outros que são mais incertos, no entanto algumas das bordas fracas podem corresponder a ruído. Para diferenciá-las de bordas reais, aplica-se um processo denominado limiarização com histerese, que utiliza dois limiares, um inferior representado por L_{inf} e um superior, representado por L_{sup}. Consideremos agora, para simplificação, $G(i, j) = G$. A classificação dos píxeis como borda ou fundo é baseada nas seguintes condições:

- Píxeis com $G > L_{\text{sup}}$ são considerados bordas fortes.

- Píxeis com $L_{\text{inf}} < G < L_{\text{sup}}$ são preservados se forem adjacentes a bordas fortes.

- Píxeis com $G < L_{\text{inf}}$ são descartados, mais uma vez, sendo definidos como 0.

Para determinar os valores para L_{inf} e L_{sup} existem várias abordagens como veremos a seguir.

Método de Hossain et al.

De acordo com Hossain et al. [74] para escolher os limiares podemos utilizar:

$$L_{inf} = \max\left(0, \frac{\mu - \sigma}{a}\right)$$

$$L_{sup} = \min\left(A, \frac{\mu + \sigma}{b}\right)$$

onde μ e σ são o valor médio e desvio padrão da função densidade de probabilidade empírica, construída a partir do histograma da imagem, A é o valor máximo de intensidade da imagem, neste caso 255, a é um parâmetro empírico entre 5 e 7, preferencialmente 6 e b é um parâmetro empírico entre 2 e 4, preferencialmente 3 [74].

Método de Otsu

Um dos critérios para definir estes limiares é a utilização do método de Otsu [75], uma técnica para seleção de limiar baseada na análise do histograma de níveis de intensidade da imagem, podendo também ser aplicado para definir automaticamente os limiares do algoritmo de Canny, como proposto por Fang et al. [76]. A ideia central do método é dividir os píxeis da imagem em duas classes e encontrar o valor de limiar que maximiza a separação entre essas classes, isso é feito por meio da maximização da variância entre as classes.

Seja $G = \{0, ..., L-1\}$ o conjunto dos níveis de cinza da imagem $I(x, y)$, e P_i a probabilidade de ocorrência do nível de cinza i. Quando se escolhe um limiar T, a imagem é dividida em duas classes, a classe $C_0 = \{0, \ldots, T\}$ que representa a região de fundo onde se encontram os píxeis com intensidade igual ou inferior a T e $C_1 = \{T+1, \ldots, L-1\}$ que representa a região que contém a borda, onde se inserem os píxeis com intensidade igual ou superior a T.

As probabilidades das duas classes são dadas por:

$$\alpha_0 = \sum_{i=0}^{T} P_i, \quad \alpha_1 = 1 - \alpha_0$$

Assim, as médias das intensidades em cada classe são dadas por:

$$\mu_0 = \frac{1}{\alpha_0} \sum_{i=0}^{T} iP_i, \quad \mu_1 = \frac{1}{\alpha_1} \sum_{i=T+1}^{L-1} iP_i$$

Como resultado, a média global da imagem é:

$$\mu = \sum_{i=0}^{L-1} iP_i$$

A função critério de Otsu, baseada na variância entre as duas classes, é expressa por:

$$\eta^2(T) = \alpha_0(\mu_0 - \mu)^2 + \alpha_1(\mu_1 - \mu)^2$$

ou, equivalentemente:

$$\eta^2(T) = \alpha_0\alpha_1(\mu_0 - \mu_1)^2$$

O valor ótimo do limiar T^* é aquele que maximiza $\eta^2(T)$ e é utilizado como o limiar superior no algoritmo de Canny, $L_{sup} = T^*$ e o limiar inferior é definido proporcionalmente como:

$$L_{inf} = \alpha \cdot L_{sup}, \quad \alpha \in [0.3, 0.6].$$

Método de Zhao et al.

Outra abordagem para definir automaticamente os limiares do algoritmo de Canny é a proposta por Zhao et al. [77], baseada numa abordagem Bayesiana e entropia cruzada, que é uma medida da diferença de informação entre duas distribuições de probabilidade definida como:

$$D(P,Q) = \sum_{i=1}^{N} p_i \ln\left(\frac{p_i}{q_i}\right)$$

onde $P = \{p_1, p_2, ..., p_N\}$ e $Q = \{q_1, q_2, ..., q_N\}$ são as duas distribuições empíricas de probabilidade. Embora P e Q sejam distribuições empíricas de probabilidade, à semelhança de P_i utilizado no método de Otsu, neste caso representam distribuições distintas associadas a diferentes regiões da imagem, a região do fundo, correspondente ao óleo lubrificante sem partículas metálicas, e a região do objeto presente, que neste caso são as partículas metálicas suspensas no óleo. Primeiramente queremos separar os píxeis da imagem em duas classes, a classe C_0 e C_1, tal como aplicado no método de Otsu descrito anteriormente. Suponhamos que p_i corresponde à distribuição da classe do fundo, então é matematicamente definida da seguinte forma:

$$p_i = \begin{cases} \dfrac{P_i}{\sum_{j=0}^{T} P_j}, & \text{se } 0 \leq i \leq T \\ 0, & \text{caso contrário} \end{cases}$$

onde T corresponde ao limiar utilizado. Então, p_i é a probabilidade de um píxel ter intensidade i, dado que pertence à classe do fundo. Analogamente podemos definir q_i:

$$q_i = \begin{cases} \dfrac{Q_i}{\sum_{j=T+1}^{L-1} Q_j}, & \text{se } T+1 \leq i \leq L-1 \\ 0, & \text{caso contrário} \end{cases}$$

Neste procedimento também será utilizada a entropia cruzada simétrica, que é definida como:

$$D(P:Q) = \sum_{i=1}^{N} p_i \ln\frac{p_i}{q_i} + \sum_{i=1}^{N} q_i \ln\frac{q_i}{p_i} \tag{3.12}$$

Assume-se que os valores de intensidade seguem distribuições Gaussianas em cada classe, com média μ_i e variância σ_i^2. Assim, a probabilidade condicional de um valor g, um valor de cinza, dado a classe $i \in \{C_0, C_1\}$ é:

$$p(g \mid i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(g-\mu_i)^2}{2\sigma_i^2}\right).$$

As probabilidades *a priori* das classes do fundo e do objeto são obtidas através do histograma da imagem $h(g)$:

$$P_{C_0} = \sum_{g=0}^{T} h(g), \quad P_{C_1} = \sum_{g=T+1}^{L-1} h(g),$$

e as variâncias correspondentes a cada uma dessas classes podem ser determinadas como:

$$\sigma^2_{C_0}(T) = \frac{1}{P_{C_0}} \sum_{g=0}^{T} h(g) \left[g - \mu_{C_0}(T)\right]^2$$

$$\sigma^2_{C_1}(T) = \frac{1}{P_{C_1}} \sum_{g=T+1}^{L-1} h(g) \left[g - \mu_{C_1}(T)\right]^2$$

Com a teorema de Bayes obtemos então as probabilidades *a posteriori*:

$$p(i \mid g) = \frac{P_i p(g \mid i)}{P_{C_0} p(g \mid C_0) + P_{C_1} p(g \mid C_1)}$$

representando P_i probabilidade a priori da classe i, neste caso $i \epsilon \{C_0, C_1\}$. Para avaliar o quão bem um valor g separa as duas classes, utilizamos a entropia cruzada simétrica definida em 3.12:

$$D(C_0 : C_1; g) = \frac{1}{3(1 + p(C_0 \mid g))} \ln\left(\frac{1 + p(C_0 \mid g)}{1 + p(C_1 \mid g)}\right) + \frac{1}{3(1 + p(C_1 \mid g))} \ln\left(\frac{1 + p(C_1 \mid g)}{1 + p(C_0 \mid g)}\right).$$

Esta fórmula mede o grau de separação entre as distribuições de fundo e borda para cada valor g, ou seja, se $p(C_0 \mid g)$ for próximo de $p(C_1 \mid g)$, a separação é fraca, caso contrário, se uma das probabilidades for muito maior, a separação é forte. Para cada valor possível de limiar T, calcula-se a entropia cruzada total entre as duas classes:

$$D(C_0 : C_1; T) = \sum_{g=0}^{T} \frac{h(g)}{P_{C_0}} D(C_0 : C_1; g) + \sum_{g=T+1}^{L-1} \frac{h(g)}{P_{C_1}} D(C_0 : C_1; g).$$

O limiar ótimo T^* é obtido de maneira a maximizar a entropia cruzada entre as classes:

$$D(C_0 : C_1; T^*) = \max D(C_0 : C_1; T)$$

onde T^* é o valor ótimo do limiar encontrado por busca iterativa. T^* será utilizado para definir o limiar superior L_{sup} e o limiar inferior L_{inf}:

$$L_{inf} = \alpha_i \cdot T^* \quad L_{sup} = \alpha_s \cdot T^*$$

onde α_i e α_s são fatores multiplicadores que definem os limiares inferior e superior, $\alpha_i < \alpha_s$.

3.7.1.4 Implementação em Python

Função `cv2.Canny()`

A função `cv2.Canny()` é utilizada para aplicar o método de deteção de bordas de Canny a uma imagem.

No código abaixo, a variável `imagem_pp` representa a imagem de entrada, que deve ser a imagem já previamente pré-processada. Os parâmetros `threshold1` e `threshold2` definem os limiares inferior e superior para a etapa de limiarização com histerese.

A função retorna uma imagem binária, onde os píxeis pertencentes às bordas são representados pelo valor 255 (branco), enquanto os restantes píxeis assumem o valor 0 (preto).

Exemplo de Implementação em Python

O código abaixo demonstra a aplicação do algoritmo de Canny a uma imagem.

```
# Aplicar a função que deteta as bordas de Canny
bordas_detetadas = cv2.Canny(imagem_pp, threshold1=50, threshold2=100)
```

Para ilustrar este exemplo de forma mais intuitiva temos em baixo representada a imagem original 3.9a e a imagem binária 3.9b resultante da aplicação do algoritmo de Canny, sem ter sido realizado pré-processamento, ou seja, aplicando diretamente a função na imagem cinza, utilizando um L_{inf} de 50 e um L_{sup} de 100:

(a) Imagem original (b) Imagem após a aplicação de Canny

Figura 3.9: Visualização das bordas detetadas utilizando o algoritmo de Canny.

3.7.2 *Clustering* como Método de Segmentação

O *clustering* permite identificar regiões com propriedades homogéneas com base em medidas de distância ou similaridade, visa identificar subconjuntos (*clusters*) cujos elementos apresentam menor variabilidade interna e maior separação entre grupos.

3.7.2.1 *K-means*

O *K-means* é um dos algoritmos de *clustering* mais utilizados pela sua simplicidade, eficiência computacional e aplicabilidade em diversos contextos. A sua finalidade é particionar um conjunto de dados $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ em k grupos distintos, de forma a minimizar a variância intra-*cluster* [78].

A função objetivo do *K-means* é dada por:

$$J = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in S_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \tag{3.13}$$

aqui S_j representa o conjunto de pontos pertencentes ao *cluster* j, e $\boldsymbol{\mu}_j$ é o centróide correspondente, definido por [78]:

$$\boldsymbol{\mu}_j = \frac{1}{|S_j|} \sum_{\mathbf{x}_i \in S_j} \mathbf{x}_i \tag{3.14}$$

A norma ao quadrado representa a distância euclidiana, ao quadrado, entre o vetor $\mathbf{x}_i$ e o centróide $\boldsymbol{\mu}_j$:

$$\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 = \sum_{d=1}^{D} \left(x_{i,d} - \mu_{j,d}\right)^2 \tag{3.15}$$

onde D é a dimensão dos vetores $\mathbf{x}_i$ e $\boldsymbol{\mu}_j$.

Para inicialização são escolhidos aleatoriamente k centróides, seguidamente cada ponto é atribuído ao *cluster* cujo centróide está mais próximo, com base na distância euclidiana 3.15. Por fim, os centróides são recalculados como a média dos pontos em cada *cluster* 3.14 e o processo repete-se até que os centróides deixem de variar significativamente, ou seja, o momento em que a função objetivo 3.13 é estabilizada, já não melhora mais. Nesta abordagem, cada píxel da imagem foi representado por um vetor tridimensional no espaço *L*a*b**, a aplicação do algoritmo *K-Means* com $k = 2$ permite agrupar os píxeis em dois *clusters*, ou seja, um correspondente às partículas e outro ao fundo.

3.7.2.2 Implementação em Pyhton

A classe `KMeans()` da biblioteca `scikit-learn` é utilizada para realizar a segmentação da imagem através de *clustering*.

No contexto deste trabalho, a imagem deve ser previamente convertida para o espaço de cores *L*a*b**, sendo depois redimensionada para um vetor bidimensional onde cada linha representa um píxel, e cada coluna representa um dos canais (L, a, b).

O parâmetro `n_clusters` define o número de grupos a serem identificados — neste caso, $k = 2$, representando partículas e fundo. O parâmetro `random_state` garante a reprodutibilidade dos resultados, e `n_init` define o número de reinicializações com diferentes centróides para melhorar o modelo.

A função `fit_predict()` ajusta o modelo aos dados e retorna os rótulos correspondentes a cada píxel que posteriormente são reformatados para as dimensões originais da imagem para visualização da segmentação.

Exemplo de Implementação em Python

```
# Aplicação do algoritmo K-means
kmeans = KMeans(n_clusters=2, random_state=0, n_init=10)
labels = kmeans.fit_predict(píxel_values)
```

3.8 Operações Morfológicas

A morfologia matemática é uma abordagem utilizada para a extração de informações estruturais em imagens através da aplicação de operações não lineares. Entre estas operações, destacam-se a dilatação e a erosão [79], sendo que a dilatação desempenha um papel fundamental na melhoria da deteção de bordas, ao permitir a conexão de segmentos descontínuos e a unificação da delimitação da estrutura [80]. Deste modo, após a deteção das bordas das partículas, aplicou-se a dilatação para assegurar a continuidade dos contornos. Como alguns píxeis da delimitação apresentavam níveis de intensidade mais elevados, pode surgir descontinuidades nos contornos, comprometendo a segmentação, a utilização da dilatação permite preencher essas lacunas, garantindo uma representação mais precisa e completa das estruturas a analisar.

3.8.1 Dilatação

Uma imagem binária pode ser definida como:

$$I(m,n) = \begin{cases} 1, & \text{se } (m,n) \text{ pertence ao objeto} \\ 0, & \text{se } (m,n) \text{ pertence ao fundo} \end{cases} \tag{3.16}$$

Numa imagem binária, que é o caso da imagem resultante da aplicação do método de Canny 3.7.1.4, a dilatação expande as regiões brancas, obtendo o efeito de preencher lacunas, através da conexão de bordas fragmentadas e ampliação das estruturas finas na imagem segmentada, o que resulta em que essa operação seja particularmente útil após a

aplicação do operador de Canny. A operação de dilatação morfológica pode ser definida matematicamente como [81]:

$$A \oplus B = \{z \in \mathbb{Z}^2 \mid (\hat{B})_z \cap A \neq \emptyset\},$$

em que A representa a imagem binária, B é um elemento estrutural, no caso tanto da dilatação como da erosão uma matriz binária, $\hat{B}$ é a reflexão de B em relação ao seu centro e z é uma posição no espaço bidimensional $\mathbb{Z}^2$, o conjunto dos pares de inteiros. Esta definição indica que o píxel z será incluído no resultado da dilatação se, ao posicionar o elemento estrutural refletido $\hat{B}$ na posição z, existir pelo menos uma sobreposição entre $\hat{B}$ e a imagem A que contenha um valor 1, ou seja, um píxel que corresponda à delimitação de uma estrutura, se houver alguma interseção não vazia entre o *kernel* e uma região branca da imagem, os píxeis sobrepostos pelo *kernel*, o píxel z é definido como 1, Isto provoca uma expansão das regiões brancas da imagem, alargando os contornos e preenchendo pequenas falhas .

No entanto, quando esta operação é aplicada no contexto de imagens digitais, como em implementações com a biblioteca OpenCV, as posições dos píxeis pertencem ao conjunto dos números naturais, já que os índices das imagens não assumem valores negativos. Assim, podemos adaptar a definição para:

$$A \oplus B = \{(m, n) \in \mathbb{N}^2 \mid (\hat{B})_{(m,n)} \cap A \neq \emptyset\},$$

em que (m, n) são coordenadas bidimensionais naturais, que representam posições reais na imagem digital. Além disso, quando o elemento estrutural B é uma matriz composta exclusivamente por 1s, como aquele considerado pelo OpenCV exceto indicação contrária, a reflexão $\hat{B}$ não altera a matriz, pois todos os seus elementos são idênticos, o que torna a operação de reflexão irrelevante neste caso específico. Ficamos então com:

$$A \oplus B = \{(m, n) \in \mathbb{N}^2 \mid B_{(m,n)} \cap A \neq \emptyset\},$$

3.8.1.1 Implementação em Python

Função `cv2.dilate()`

A função `cv2.dilate()` é utilizada para aplicar a operação morfológica de dilatação a uma imagem binária.

No código abaixo, a variável `imagem_canny` representa a imagem de entrada, que deve ser a imagem resultante da aplicação de Canny

O parâmetro `kernel` que é uma matriz que define o elemento estrutural utilizado para a dilatação, define a forma e o tamanho da expansão das bordas. Neste exemplo, utiliza-se

`np.ones((5,5), np.uint8)`, que cria um kernel de 5x5 preenchido com valores 1, promovendo uma expansão uniforme das bordas. O último parâmetro, `iterations`, representa o número de vezes que a operação será aplicada, influenciando a intensidade da dilatação.

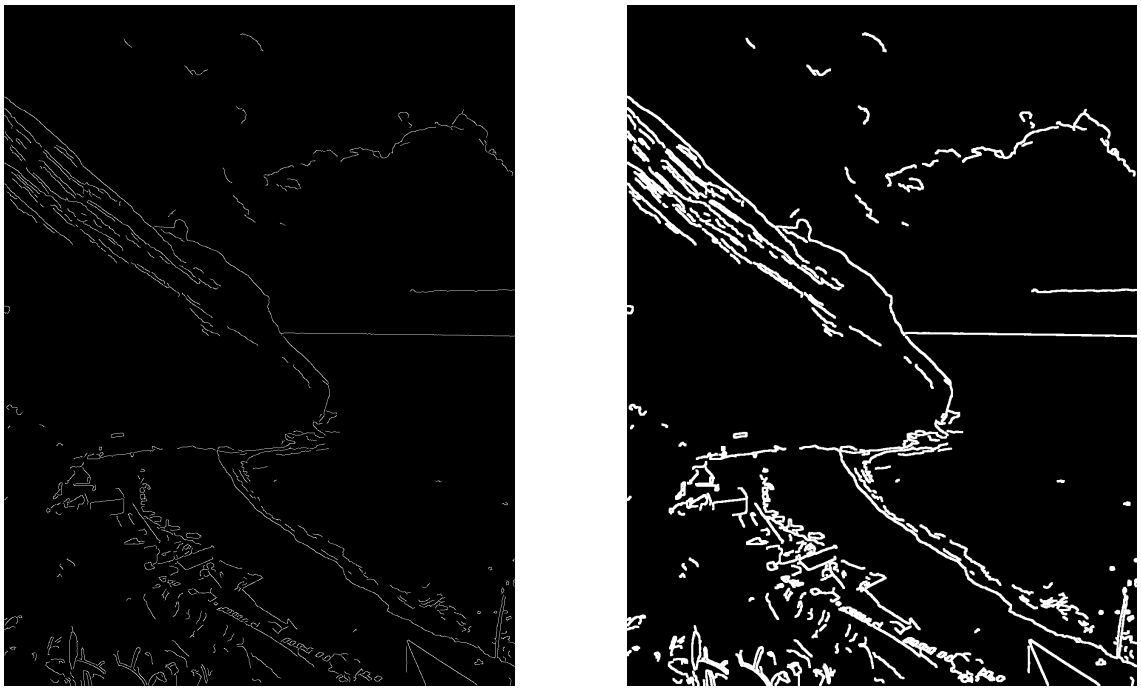
A função retorna uma imagem binária onde as bordas detetadas são expandidas.

Exemplo de Implementação em Python

```
# Criar um elemento estrutural, kernel, de tamanho 5x5
kernel = np.ones((5,5), np.uint8)

# Aplicar a dilatação na imagem de bordas detetadas
imagem_dilatada = cv2.dilate(imagem_canny, kernel, iterations=1)
```

Para ilustrar este exemplo de forma mais intuitiva temos em 3.10a representada a imagem binária resultante da aplicação algoritmo de Canny e em 3.10b a imagem, também binária, resultante da aplicação de dilatação à imagem resultante do algoritmo de Canny, utilizando um *kernel* de tamanho 5×5 e realizando 1 iteração:



(a) Imagem após a aplicação de Canny (b) Imagem após aplicar dilatação

Figura 3.10: Visualização das bordas detetadas após ser aplicada dilatação.

A erosão desempenha um papel essencial na remoção de ruídos indesejados em imagens. Assim, após a deteção das bordas das partículas, a erosão pode ser utilizada para eliminar pequenas estruturas, que na verdade são ruído e não elementos de interesse.

3.8.2 Erosão

Numa imagem binária a erosão atua contraindo as regiões brancas, o inverso do que acontece na dilatação. Esta operação reduz os contornos dos objetos, ao remover alguns píxeis de borda e pequenas estruturas detetadas que são ruído. A operação de erosão morfológica pode ser definida matematicamente como [81]:

$$A \ominus B = \{z \in \mathbb{Z}^2 \mid B_z \subseteq A\},$$

em que A representa a imagem binária, B é um elemento estrutural e z é uma posição no espaço bidimensional $\mathbb{Z}^2$. Esta definição indica que o píxel z será incluído no resultado da erosão apenas se, ao posicionar o elemento estrutural B na posição z, todos os píxeis cobertos por B estiverem contidos na região branca da imagem A. Isto significa que a erosão apenas mantém os píxeis centrais suficientemente rodeados por outros píxeis brancos, removendo os que estiverem parcialmente rodeados por fundo, substituindo o seu valor de intensidade para 0.

No contexto digital, como em implementações com a biblioteca OpenCV, os píxeis pertencem ao conjunto dos números naturais, pois os índices da imagem não assumem valores negativos. Assim, podemos adaptar a definição para:

$$A \ominus B = \{(m, n) \in \mathbb{N}^2 \mid B_{(m,n)} \subseteq A\},$$

em que (m, n) são coordenadas naturais que representam as posições do píxel em questão na imagem digital. Quando o elemento estrutural B é uma matriz composta apenas por 1s, como ocorre por omissão no OpenCV, a operação de erosão depende inteiramente da presença de todos os elementos do *kernel* sobre píxeis com valor 1. Caso contrário, o píxel central será erodido ou seja definido como 0.

3.8.2.1 Implementação em Python

Função `cv2.erode()`

A função `cv2.erode()` é utilizada para aplicar a operação morfológica de erosão a uma imagem binária.

No código abaixo, o parâmetro `kernel` define o elemento estrutural que determina o como será aplicada a erosão. Neste exemplo, utiliza-se `np.ones((5,5), np.uint8)`, que cria um *kernel* de 5×5 com valores 1, permitindo uma erosão uniforme das regiões brancas, tal como descrito acima. O parâmetro `iterations` controla o número de vezes que a operação será aplicada, o que afeta o grau de erosão das estruturas.

A função retorna uma imagem binária em que as regiões brancas estão mais contraídas do que na imagem de entrada.

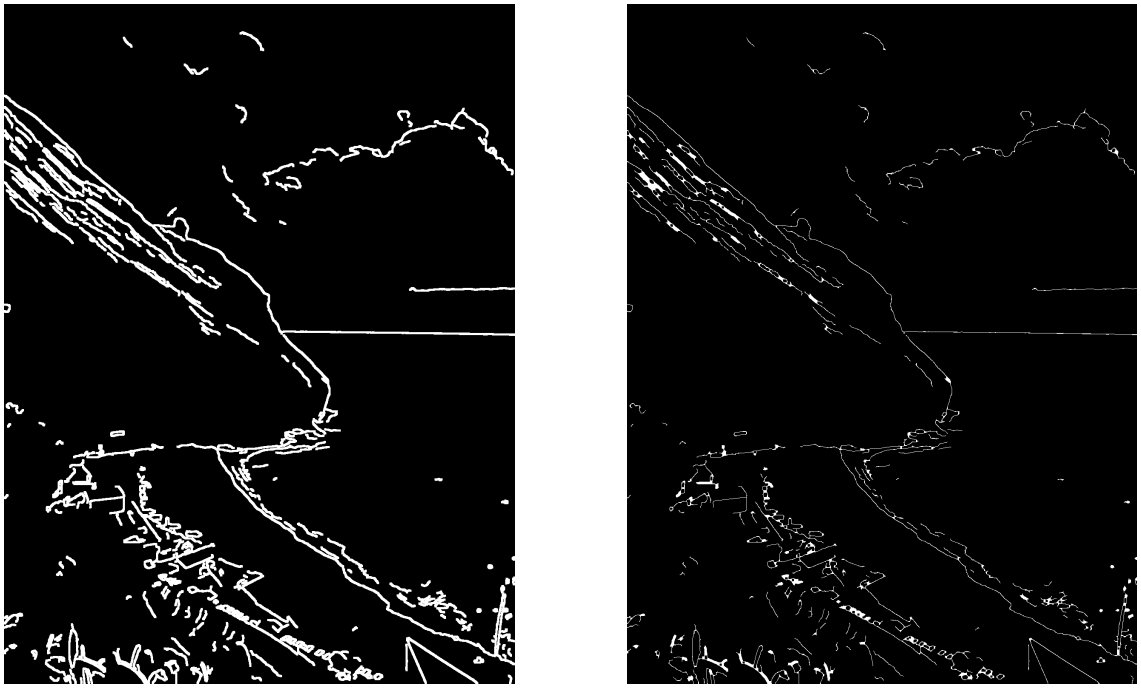
Exemplo de Implementação em Python

```
# Criar um elemento estrutural, kernel, de tamanho 5x5
```

```
kernel = np.ones((5,5), np.uint8)

# Aplicar a erosão na imagem de bordas detetadas
imagem_erosionada = cv2.erode(imagem, kernel, iterations=1)
```

Para ilustrar este exemplo de forma mais clara, temos em 3.11a a imagem binária resultante da aplicação de dilatação e em 3.11b a imagem, também binária, resultante da aplicação da erosão à imagem já dilatada, com um *kernel* de tamanho 5×5 e 1 iteração:



(a) Imagem após a aplicação de dilatação (b) Imagem após aplicar erosão

Figura 3.11: Visualização das bordas detetadas após ser aplicada erosão.

3.9 Encontrar os Contornos das Partículas

A deteção de contornos é o processo de identificação das fronteiras entre objetos e fundo numa imagem binária, onde os objetos são representados por píxeis com valor 1 e o fundo por píxeis com valor 0. O objetivo é obter o conjunto de pontos que delimitam a forma das partículas presentes na imagem, após a aplicação da dilatação, a deteção de contornos torna-se mais eficaz, uma vez que a dilatação ajuda a conectar segmentos descontínuos e a suavizar descontinuidades nos objetos segmentados [82]. Tendo isto em consideração, um contorno pode ser definido como o conjunto de píxeis pertencentes ao objeto, ou seja com valor 1, que estão adjacentes a pelo menos um píxel de fundo, ou seja com valor 0 sendo expresso como:

$$C = \left\{(x, y) \mid I(x, y) = 1 \wedge \exists (x', y') \in V(x, y) : I(x', y') = 0\right\} \tag{3.17}$$

onde $V(x, y)$ representa a vizinhança de 8-conectividade do píxel (x, y), abaixo definida.

A deteção de contornos no OpenCV baseia-se no algoritmo de Suzuki e Abe [82], que segue um método que rastreia fronteiras para extrair contornos de forma hierárquica. O algoritmo consiste nas seguintes etapas:

1. Identificação do primeiro píxel do objeto: o processo de deteção começa com a procura do primeiro píxel pertencente ao objeto na imagem binária, ou seja, o primeiro píxel com valor 1. Esta pesquisa ocorre de forma sistemática, da esquerda para a direita e de cima para baixo, garantindo a deteção do píxel mais a norte e a oeste do objeto. O píxel inicial, designado por P_0, é dado por:

$$P_0 = min\{(x, y) \mid I(x, y) = 1\}$$

   Este será o ponto de partida para encontrar os restantes píxeis pertencentes ao contorno.

2. Seguimento do contorno usando vizinhança de 8-conectividade: uma vez identificado o píxel inicial, o algoritmo segue o contorno do objeto, deslocando-se através de uma vizinhança de 8-conectividade, o que significa que a procura pelo próximo píxel do contorno pode ser feita nas 8 direções adjacentes ao píxel atual. O conjunto de direções discretas d_i é representado por:

$$d_i \epsilon \{0, 1, 2, 3, 4, 5, 6, 7\}$$

   Cada valor de d_i corresponde a uma direção específica, por exemplo d_0 corresponde ao píxel acima e à esquerda.

3. Representação por código de cadeia (*Chain Code*): após a extração do contorno, este pode ser armazenado como uma sequência de direções de deslocamento conhecida como código de cadeia (*Chain Code*):

$$C = \{d_1, d_2, ..., d_n\}$$

   Cada d_i indica a direção do deslocamento do píxel atual para o próximo ao longo do contorno, este é armazenado como uma sequência de direções de deslocamento. Como por exemplo, se obtivermos $C = \{1, 2, 5\}$, significa que do P_0 iremos seguir para o píxel diretamente acima deste, por sua vez seguimos para o píxel acima e à direita do píxel anterior e por fim para o píxel diretamente à direita do anterior.

4. Construção da hierarquia de contornos: por fim, os contornos detetados são organizados numa estrutura hierárquica, distinguindo-se os contornos externos, que delimitam objetos principais, e os contornos internos, que definem regiões vazias dentro desses objetos. Esta organização possibilita a segmentação de estruturas aglomeradas. Os contornos são organizados numa estrutura hierárquica, onde contornos externos e internos são diferenciados.

A hierarquia criada organiza os contornos em diferentes níveis de profundidade, distinguindo os contornos externos dos contornos internos. Os contornos externos correspondem ao nível hierárquico 0 e são aqueles que não estão contidos em nenhum outro contorno, representando, na prática, os limites exteriores dos objetos. Por sua vez, os contornos internos são aqueles que se encontram dentro de um contorno externo e são associados ao contorno que os envolve, denominado contorno pai, caso existam subníveis (por exemplo, buracos dentro de buracos), o método atribui níveis hierárquicos sucessivamente crescentes, resultando numa estrutura em árvore na qual cada nó representa um contorno e os seus nós filhos correspondem aos contornos internos desse objeto. Esta distinção é feita durante o processo de deteção de contornos na qual é determinada a natureza de cada contorno através da análise da vizinhança dos píxeis, quando um novo píxel de objeto é encontrado, verifica-se a sua posição em relação aos píxeis adjacentes, se o píxel à sua esquerda pertencer ao fundo, isto indica o início de um contorno externo. Adicionalmente, mantém uma variável de controlo que regista o rótulo do contorno previamente detetado na mesma linha, caso o novo contorno surja numa região já rodeada por um contorno previamente rotulado, conclui-se que este se encontra contido nesse contorno, classificando-o como interno. Por outro lado, se não houver evidência de uma delimitação pré-existente na vizinhança imediata, o contorno é considerado externo.

No contexto do presente trabalho, o objetivo principal da deteção de contornos é identificar os limites exteriores das partículas, então apenas os contornos externos são relevantes. A seleção desses contornos é feita automaticamente durante a implementação, através da escolha de um modo específico de recuperação da hierarquia, como será detalhado adiante.

3.9.1 Implementação em Python

Função `cv2.findContours()`

A função `cv2.findContours()` é utilizada para detetar os contornos numa imagem binária processada.

No código abaixo, a variável `imagem_dilatada` representa a imagem de entrada após a dilatação, que melhora a continuidade dos contornos.

O parâmetro `cv2.RETR_EXTERNAL` especifica o modo como os contornos serão recuperados da hierarquia criada pelo algoritmo, tal como mencionado acima, apenas é de interesse selecionar os contornos externos das partículas, quando este modo é selecionado, a função `cv2.findContours()` retorna apenas os contornos externos, correspondentes ao nível hierárquico 0. Por sua vez, o parâmetro `cv2.CHAIN_APPROX_SIMPLE` reduz o número de pontos armazenados, simplificando os contornos.

A função `cv2.findContours()` devolve como resultado uma lista de contornos, onde cada contorno é representado por um vetor de pontos que descrevem a sua forma ao longo da imagem. Cada um desses contornos é constituído por uma sequência de coordenadas (x, y) que pertencem à linha de fronteira de uma região conectada na imagem binária.

Exemplo de Implementação em Python

```
# Aplicar a deteção de contornos na imagem dilatada

contornos, _ = cv2.findContours(imagem_dilatada, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

3.10 Encontrar o tamanho das partículas

Universalmente, as partículas são frequentemente comparadas com base no seu diâmetro, que é a distância entre duas linhas paralelas tangentes aos lados opostos de uma partícula, sendo este conhecido como diâmetro de Feret [83].

3.10.1 Diâmetro de Feret

Encontrados os contornos vamos agora percorrer cada um e encontrar os dois pontos mais distantes, percorrendo todos os pares de pontos dentro deste e calculando a distância entre eles baseada na distância euclidiana definida em 3.8. Para encontrar esta distância cria-se uma função que inicialmente considere a distância máxima como zero, seguidamente percorre cada par de píxeis e sempre que a distância entre um novo par de píxeis for superior à distância máxima atual, esta é atualizada com o novo valor. No final do processo, a maior distância registada será então o diâmetro de Feret do objeto, em píxeis. Matematicamente, dado um conjunto de pontos $S = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^2$, correspondente ao contorno de um objeto, o diâmetro de Feret, D_F, é definido como:

$$D_F = \max_{p_i, p_j \in S} \{\|p_i - p_j\|_2\} = \max_{p_i, p_j \in S} \left\{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right\} \tag{3.18}$$

onde $p_i = (x_i, y_i)$ e $p_j = (x_j, y_j)$ são coordenadas de dois pontos distintos pertencentes ao contorno da partícula.

3.10.2 Conversão de Escala

Em imagens digitais obtidas por microscopia, ao serem analisadas em ambiente computacional, as dimensões das estruturas de interesse são, inevitavelmente, expressas em píxeis. Para que essas medidas possam ser interpretadas em unidades físicas reais é necessário realizar uma conversão com base numa barra de escala, presente na imagem, que indica a correspondência entre uma determinada distância na imagem e um valor real conhecido.

Como a barra de escala é um elemento gráfico e não uma entidade geométrica ideal, o primeiro passo consiste em extrair uma representação linear aproximada que reflita sua posição e orientação. A partir dessa representação, mede-se o seu comprimento em

píxeis, o que, combinado com a informação de escala fornecida, permite determinar um fator de conversão entre píxeis e a unidade física desejada.

3.10.2.1 Transformada de Hough

Para encontrar o segmento de linha que replique a barra de escala é utilizado o recurso à Transformada de Hough que, aplicada sobre uma imagem binarizada, identifica segmentos lineares presentes na imagem. Assim, após a deteção das linhas, selecionar-se-á aquela que apresenta características compatíveis com a barra de escala, ou seja, comprimento, posição e orientação.

A Transformada de Hough é uma técnica utilizada na identificação de linhas em imagens, baseada na ideia de transformar os pontos da imagem para um espaço de parâmetros, onde cada ponto "vota"em todas as retas que o podem conter [84].

Inicialmente podemos considerar que, ao procurar retas numa imagem, uma abordagem mais intuitiva seria considerar a equação explícita de uma reta:

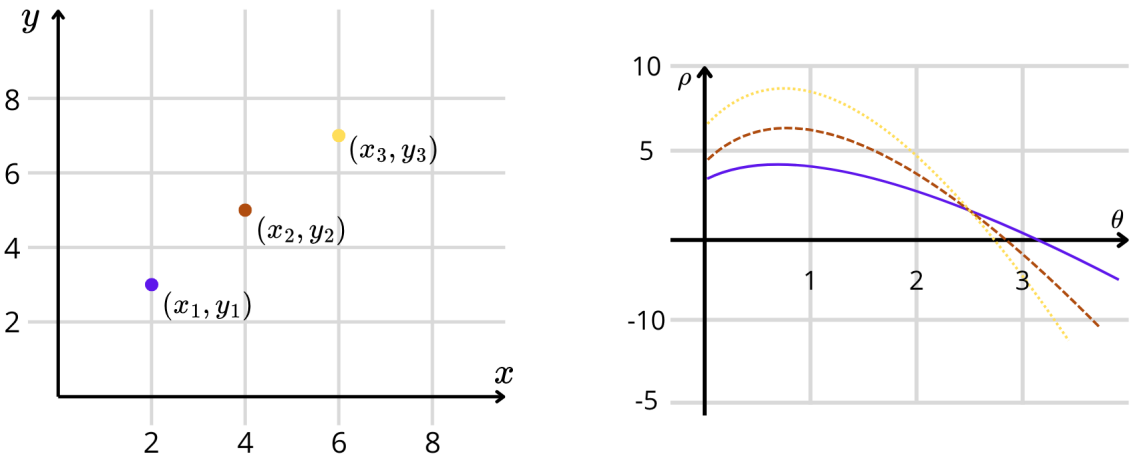
$$x_2 = kx_1 + q \tag{3.19}$$

onde onde k representa a inclinação da reta e q a ordenada na origem. No entanto, esta representação apresenta limitações, para retas verticais, onde o declive tende para infinito [85].

Para superar esta dificuldade, utiliza-se a forma polar da equação da reta [86]:

$$\rho = x_i \cos\theta + y_i \sin\theta, \quad \forall\theta \in [0, \pi] \tag{3.20}$$

nesta parametrização, ρ representa a distância da reta à origem e θ o ângulo entre o eixo x e a perpendicular à reta. Cada ponto (x_i, y_i) com valor 255 na imagem binarizada 3.12b contribui, então, com uma curva sinoidal no espaço de Hough 3.12b, representando todas as retas possíveis que o atravessam. Quando vários pontos pertencem a uma mesma reta, as suas curvas intersectam-se num ponto comum (ρ, θ), que define essa reta.



(a) Pontos na imagem I

(b) Curvas sinusoidais no Espaço de Hough

Figura 3.12: A transformada de Hough mapeia pontos da imagem (à esquerda) para curvas sinoidais no espaço de Hough (à direita). Baseado em [86].

O processo de deteção consiste em identificar os máximos no espaço de Hough, ou seja, os pares (ρ, θ) com o maior número de interseções, ou seja com maior número de "votos", como referido anteriormente, e que, por isso, correspondem às retas mais prováveis de estarem presentes na imagem.

Implementação em Python

Função `cv2.HoughLines()`

A função `cv2.HoughLines()` permite aplicar a Transformada de Hough a uma imagem binária. Os parâmetros incluem `rho`, que corresponde à resolução da distância em píxeis, `theta` a resolução do ângulo θ em radianos e `threshold` o número mínimo de interseções para que uma linha seja considerada.

A função retorna um *array* contendo os pares (ρ, θ) das linhas detetadas.

Exemplo de Implementação em Python

```
# Aplicar a transformada de Hough à imagem binarizada
linhas = cv2.HoughLines(imagem_binaria, rho = 1, theta = np.pi/180, threshold = 100)
```

Após identificar a linha correta, o seu comprimento em píxeis pode ser comparado com o valor real da escala em micrómetros, permitindo assim calcular a proporção `píxel/unidade` para converter medições realizadas diretamente na imagem para valores reais.

4

Resultados e Discussão

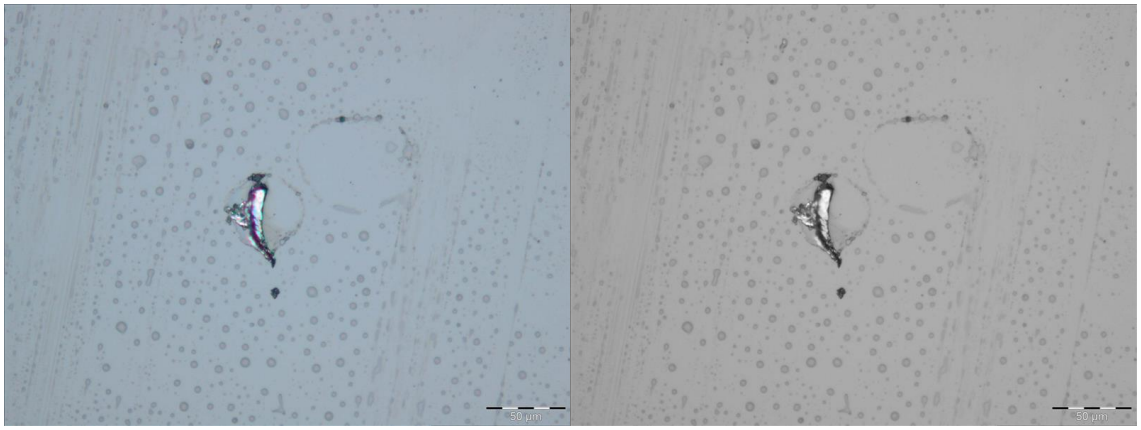
Neste capítulo são apresentados e discutidos os resultados obtidos a partir do conjunto de imagens disponíveis, constituído por dezasseis imagens de laboratório, recolhidas em condições controladas, e duas imagens de terreno, obtidas em ambiente real. Ambos os conjuntos de imagem se encontram em anexo (I).

O objetivo desta fase não foi aplicar diretamente o algoritmo já finalizado, mas sim avaliar e comparar diferentes técnicas de pré-processamento, aumento de contraste, suavização e segmentação, de modo a selecionar as mais adequadas para integrar posteriormente a construção do algoritmo final.

Assim, o capítulo inicia-se com a análise das imagens de laboratório, que permitiram testar os vários métodos em condições controladas e identificar aqueles que conduziram a melhores resultados. De seguida, apresentam-se os testes realizados sobre as imagens de terreno, onde se destacam as limitações encontradas relativamente ao desempenho observado em laboratório. Por fim, é descrita uma nova abordagem proposta para lidar com estas limitações, cuja validação, contudo, foi comprometida pelo número reduzido de imagens e pela ausência de informação complementar.

4.1 Conversão para Escala de Cinza

Abaixo encontram-se os resultados da aplicação do método de transformação de imagens RGB em escala de cinza implementando a função do OpenCV, apresentando em 3.3.1, a uma das imagens disponibilizadas pela empresa GALP, obtida em laboratório:



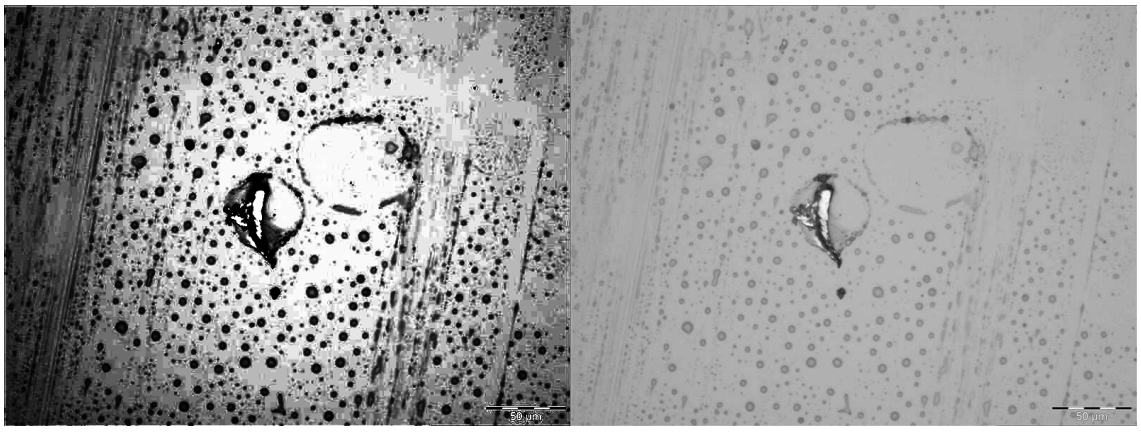
(a) Imagem RGB (b) Imagem Cinza

Figura 4.1: Visualização do efeito da conversão do espaço de cores RGB para escala de cinza.

4.2 Aumento do Contraste

4.2.1 Equalização de Histograma e CLAHE

Abaixo encontram-se os resultados visuais da aplicação dos dois diferentes processos referidos em 3.4.1 e 3.4.2, respetivamente a equalização de histograma e o CLAHE:



(a) Equalização de Histograma (b) CLAHE

Figura 4.2: Visualização do resultado da aplicação de contraste utilizando os dois processos diferentes.

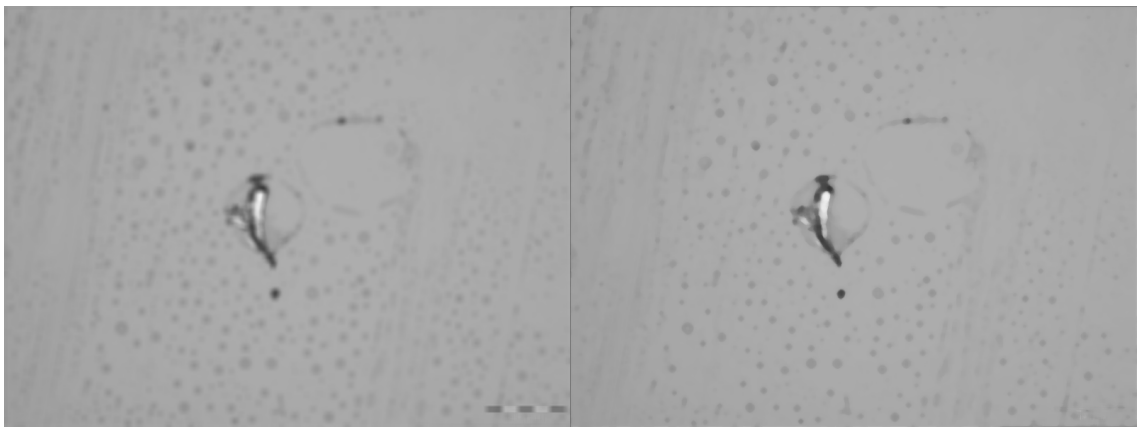
Na análise visual das imagens resultantes dos dois métodos, observa-se que em 4.2a o contraste aumentou significativamente. Porém, essa intensificação vem acompanhada de uma amplificação considerável do ruído presente na imagem, o que resulta em que algumas regiões se tornem excessivamente destacadas, dificultando a interpretação dos detalhes relevantes, pois acabam por se confundir com saliências que correspondem a ruído. Em contrapartida, por observação de 4.2b, embora também seja notado um certo aumento de ruído, tal é feito de maneira muito mais controlada, devido ao contraste ser

aumentado regionalmente ao invés de globalmente, tal como descrito em 3.4, resultando numa imagem com melhor equilíbrio entre contraste e preservação da qualidade visual. Diante dessas diferenças, optou-se pelo uso do CLAHE, no entanto, é importante destacar que, apesar de o CLAHE proporcionar um aumento de contraste mais equilibrado em comparação com a equalização de histograma, ainda se verifica uma amplificação residual do ruído presente na imagem. Embora essa intensificação seja visivelmente menos agressiva, não deixa de comprometer, em certa medida, a qualidade do resultado final. Tendo isso em consideração, optou-se por explorar uma abordagem alternativa, consistindo na aplicação prévia de uma técnica de suavização, com o objetivo de atenuar o ruído antes da execução do CLAHE.

4.3 Suavização

4.3.1 Comparação filtros de Média e Mediana

Abaixo encontram-se os resultados visuais da aplicação dos dois diferentes processos referidos em 3.5.1 e 3.5.2, respetivamente os filtros de suavização de média e mediana:



(a) Filtro de Média (b) Filtro de Mediana

Figura 4.3: Visualização do resultado da aplicação de suavização utilizando os filtros de média e mediana.

Na tentativa de remover ao máximo o ruído envolvente da partícula observada na imagem apresentada em 4.1b, sem comprometer excessivamente os detalhes relevantes, foi testado um *kernel* 9×9, uma pequena matriz de números aplicada sobre a imagem, combinando os valores dos píxeis vizinhos. Na comparação entre os dois métodos de suavização aplicados, observa-se que o filtro da média (4.3a) resulta numa imagem com redução global do ruído, mas com um efeito de suavização que compromete a definição das bordas e dos detalhes mais finos, como as fronteiras das cavidades e da fratura central. Por outro lado, o filtro da mediana (4.3b) demonstra uma capacidade mais eficaz de eliminar ruídos pontuais, preservando simultaneamente os contornos e estruturas relevantes da imagem. Considerando que, no âmbito deste trabalho, a preservação das

bordas e detalhes estruturais é essencial para a análise visual e morfológica das amostras, conclui-se que a aplicação do filtro da média poderá não ser a abordagem mais adequada, sendo a mediana uma alternativa mais apropriada entre estas duas.

4.3.2 Filtro Gaussiano e Variação do Parâmetro σ

A seguir apresenta-se o resultado da aplicação de um filtro gaussiano utilizando um *kernel* 15×15. Este valor foi aumentado em relação àquele definido anteriormente, pois não se verificaram diferenças visíveis quando aplicado um *kernel* de 9×9, sobre a imagem em escala de cinza referida em 4.1b. Ao contrário dos filtros da média e da mediana, em que o aumento do tamanho do *kernel* conduz diretamente a uma maior suavização da imagem, no filtro gaussiano o efeito não depende apenas dessa dimensão. Isto acontece pois neste filtro os pesos atribuídos aos píxeis vizinhos seguem uma distribuição gaussiana, concentrando a maior influência nos píxeis próximos do centro. Assim, pode afirmar-se que o filtro gaussiano é menos sensível ao aumento do tamanho do *kernel*, sendo o grau de suavização determinado sobretudo pelo valor do desvio padrão.

Mostrando em cada região o efeito de valores crescentes de σ, no canto superior esquerdo observamos o efeito de um $\sigma = 1$, no canto superior direito de um $\sigma = 5$, no canto inferior esquerdo, $\sigma = 10$ e no canto inferior direito $\sigma = 15$:

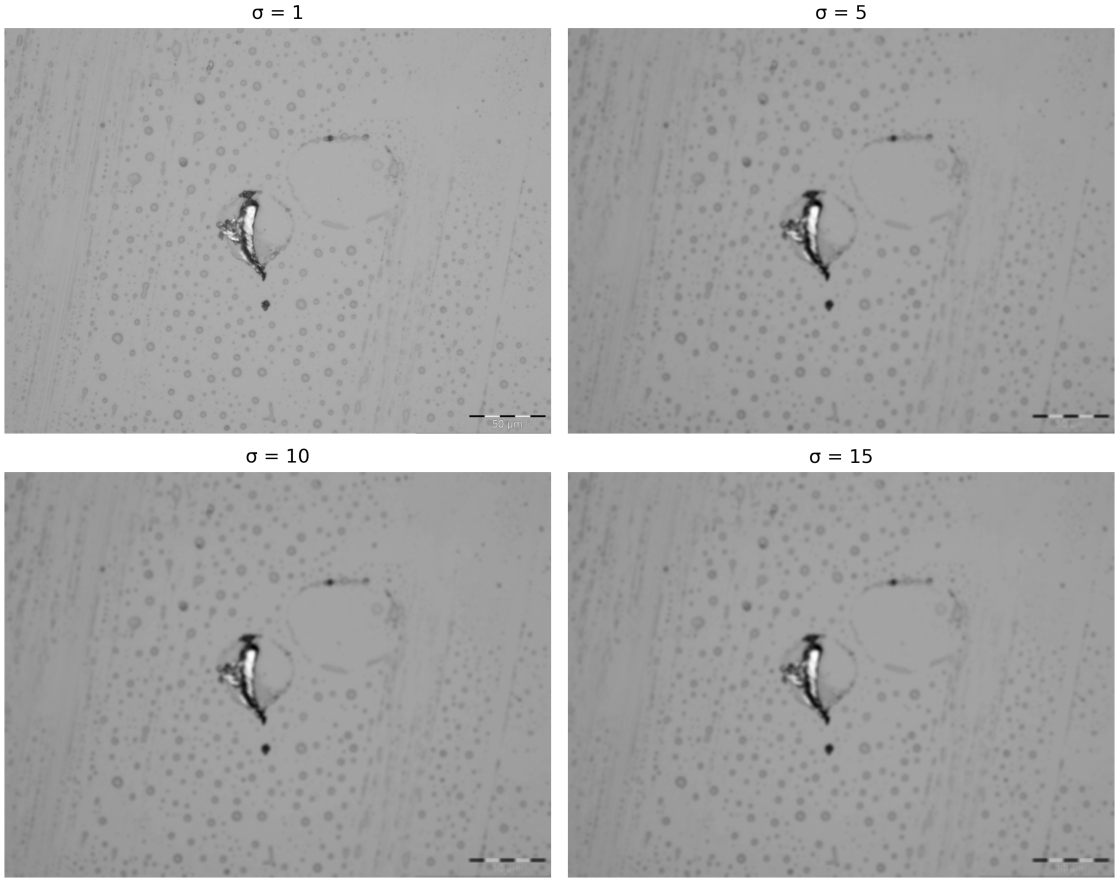


Figura 4.4: Efeito visual da variação do parâmetro σ.

Observa-se que:

- Para $\sigma = 1$, um σ relativamente pequeno, não é percetível uma redução do ruído.
- Com $\sigma = 5$ e $\sigma = 10$, há atenuação do ruído, mas começa a surgir um desfoque nas delimitações das partículas de interesse.
- Para $\sigma = 15$ o desfoque torna-se dominante e ainda se vê bastante ruído, além disso, as estruturas relevantes perdem nitidez e os contornos ficam excessivamente suavizados.

Em comparação com todos os filtros testados até ao momento, o filtro de mediana apresenta os resultados mais desejados, dado que, embora a desfoque, pelo menos a partícula mantém-se em destaque. Dessa forma, considerando que a preservação das partículas é o mais importante, conclui-se que o filtro de mediana proporciona os melhores resultados entre todas as opções avaliadas até este ponto.

4.3.3 Filtro Bilateral e Variação dos Parâmetros σ_r e σ_s

A seguir apresenta-se o resultado da aplicação do filtro bilateral sobre a imagem em escala de cinza referida em 4.1b, utilizando diferentes combinações dos parâmetros σ_r e σ_s, com um *kernel* fixo de 15×15. Na figura 4.5, cada quadrante ilustra o efeito visual da suavização para as diferentes intensidades dos parâmetros.

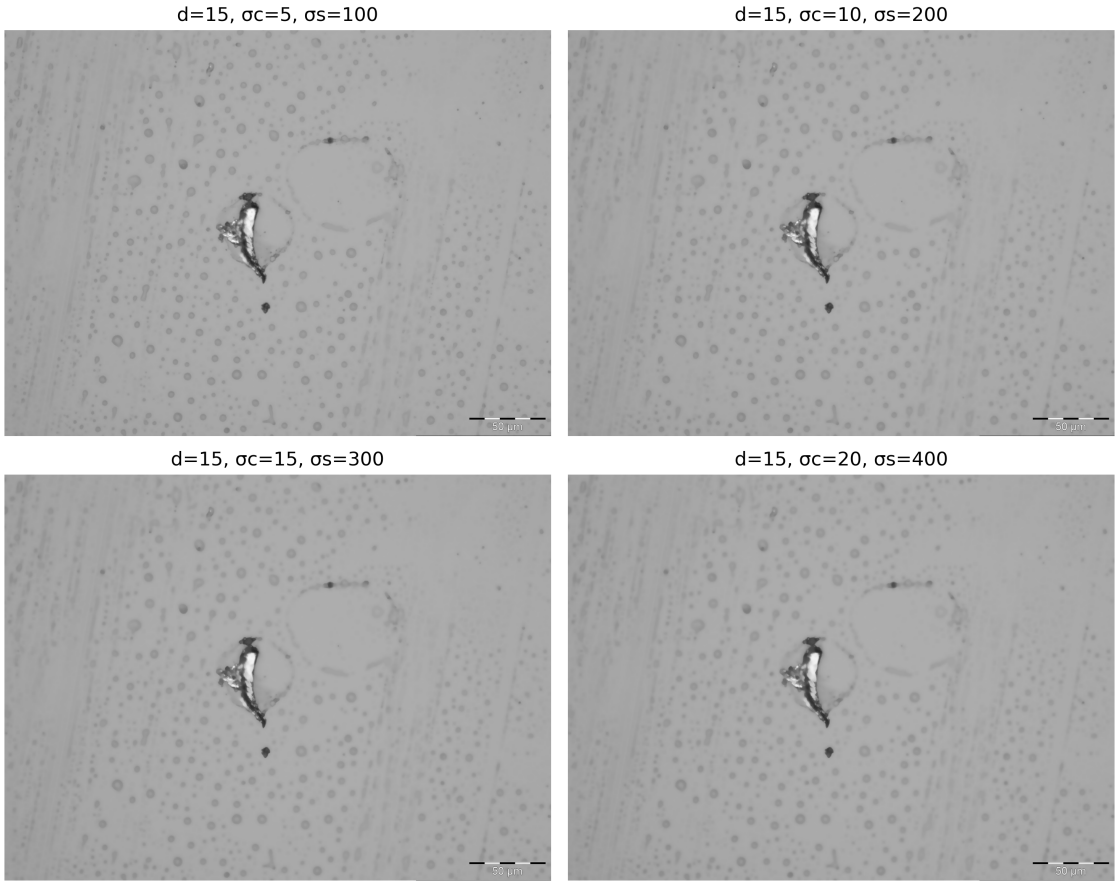


Figura 4.5: Efeito visual da variação dos parâmetros σ_r, na imagem representado como σ_c, e σ_s do filtro bilateral.

Observa-se que:

- Para os valores de $\sigma_r = 5$ e $\sigma_s = 100$ a suavização é ligeira e o ruído permanece visível, embora os contornos da partícula se mantenham nítidos.
- Com $\sigma_r = 10$ e $\sigma_s = 200$, observa-se uma redução do ruído, no entanto também a partícula fica desfocada.
- Para $\sigma_r = 15$ e $\sigma_s = 300$ também há uma redução de ruído, no entanto ainda não tanto como é necessário, e a partícula mantém-se com qualidade visual.
- Para valores mais elevados $\sigma_r = 20$, $\sigma_s = 400$, o ruído é diminuído, mas a textura da partícula é suavizada em excesso, levando a perda de detalhes importantes para análise morfológica.

Por análise visual do comportamento do filtro bilateral, observou-se que a variação do parâmetro σ_r teve um impacto significativamente mais acentuado na suavização da imagem do que a variação de σ_s; para um valor mais elevado de σ_r nota-se um desfoque mais pronunciado, afetando a nitidez das partículas, enquanto que a variação σ_s, que controla

a influência espacial dos píxeis vizinhos, mostrou-se visualmente menos agressiva, não comprometendo tanto os detalhes estruturais. Esta diferença sugere que, para aplicações em que a preservação dos contornos é essencial, como neste trabalho, pode ser vantajoso manter σ_r em valores moderados podendo-se trabalhar com valores mais altos de σ_s, de modo a alcançar um bom equilíbrio entre suavização e preservação das estruturas. Adicionalmente, ao contrário do comportamento observado com os filtros de média, mediana e gaussiano, no filtro bilateral a variação da dimensão do *kernel* teve um efeito relevante e benéfico, ao experimentar um *kernel* maior, de 27×27, 3 vezes o tamanho do *kernel* utilizado até agora, foi possível obter uma suavização mais eficaz e homogénea, sem perda excessiva de contorno. Esta melhoria justifica a apresentação, em seguida, de uma imagem resultante da aplicação do filtro bilateral onde além da variação dos parâmetros σ_r e σ_s, também varia o tamanho do *kernel*, representado por d, mantendo as combinações utilizadas em 4.5. Abaixo encontra-se esta comparação:

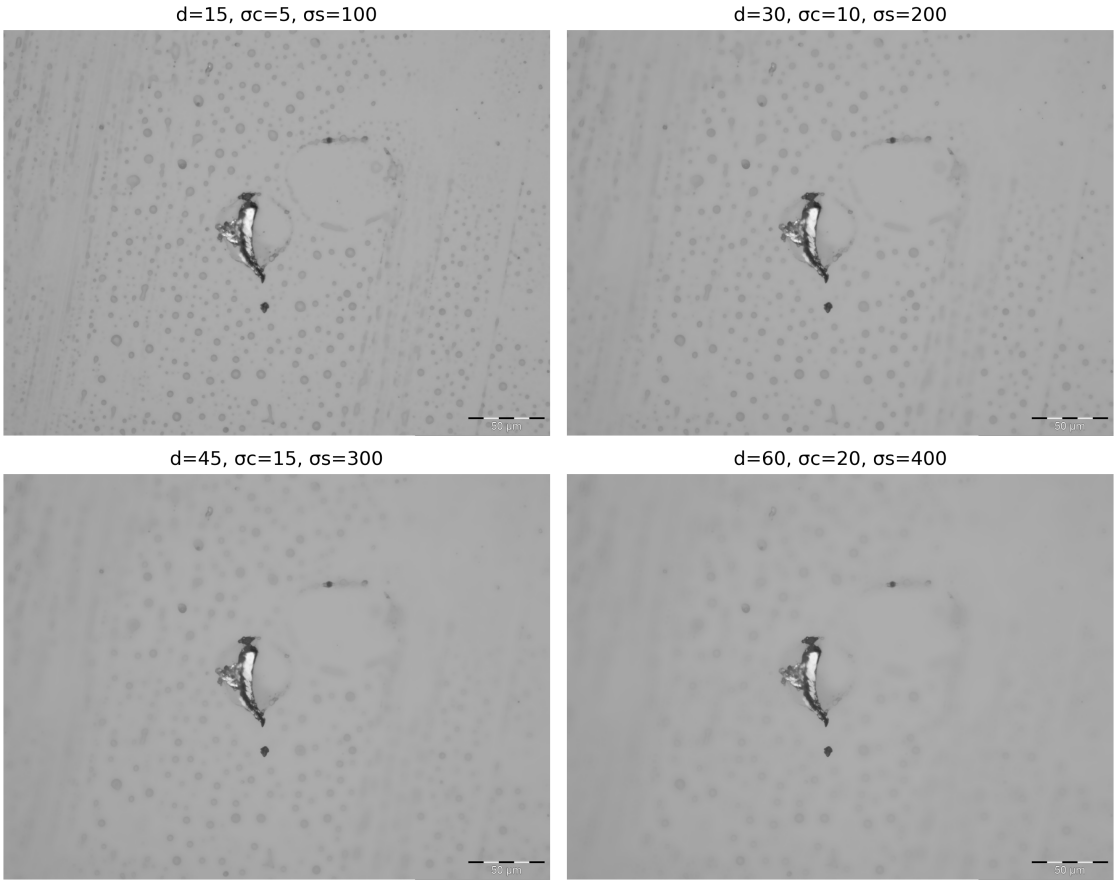


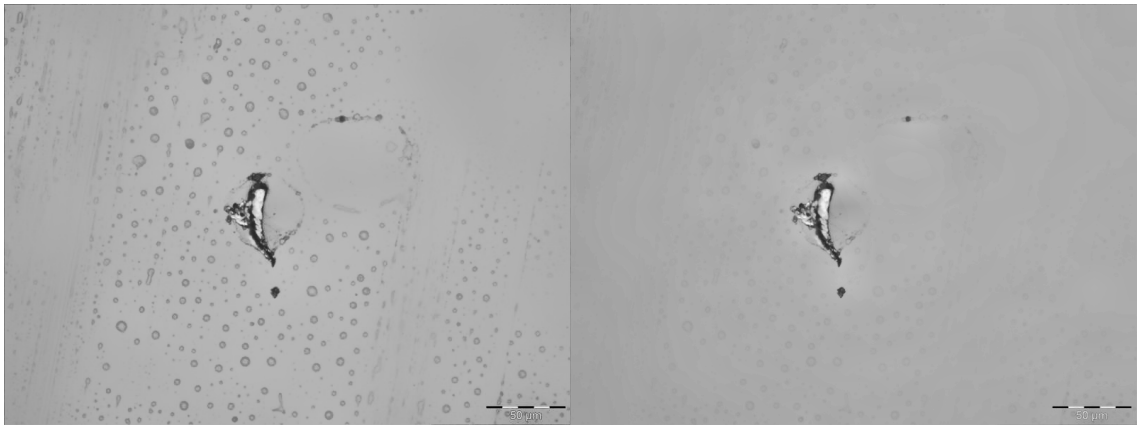
Figura 4.6: Efeito visual da variação do tamanho do *kernel* no filtro bilateral.

Por observação desta última figura, parece existir uma boa preservação da estrutura morfológica da partícula no lado esquerdo, onde o valor de σ_r é mais moderado, enquanto

que no lado direito já se observa algum desfoque da partícula, tanto na sua estrutura interna como na sua delimitação. Este aumento do *kernel* levou a que o ruído fosse substancialmente reduzido em todas as imagens havendo um bom equilíbrio entre redução de ruído e preservação estrutural, para um σ_r próximo de 10, o que levou a concluir que o filtro bilateral é, entre todos os avaliados, o que oferece o melhor resultado, sendo assim a abordagem mais adequada para os objetivos deste trabalho. Por experimentação os parâmetros escolhidos foram um *kernel* de 100, um σ_r de 20 e um σ_s de 500.

4.4 Suavização e Aumento de contraste

Após a seleção do filtro bilateral como a técnica de suavização mais eficaz, foi explorada a sequência ideal de processamento entre suavização e aumento de contraste. Inicialmente, testou-se a aplicação do aumento de contraste através do método CLAHE antes da suavização, conforme sugerido em algumas abordagens da literatura. No entanto, esta estratégia revelou-se subótima para as imagens em análise dado que, ao aplicar CLAHE diretamente sobre a imagem original, verificou-se um aumento excessivo do contraste em regiões com ruído como podemos observar em 4.7a. Face a este resultado, inverteu-se a ordem das operações, primeiro foi aplicada a suavização com o filtro bilateral com os parâmetros indicados em 4.5. Esta sequência mostrou-se mais eficaz, produzindo imagens com contornos preservados, ruído atenuado e contraste local melhorado como observado em 4.7b.



(a) Aumento do Contraste seguido de Suavização(b) Suavização seguida de Aumento do Contraste

Figura 4.7: Visualização do resultado da ordem de aplicação de aumento do contraste e suavização.

Esta análise permitiu concluir que a aplicação do filtro bilateral antes do aumento de contraste não só melhora a nitidez da imagem ao reduzir o ruído, como também contribui para uma segmentação e análise morfológica mais fiável, sendo esta a estratégia adotada para o restante processamento no presente trabalho.

4.5 Avaliação da Imagem Pré-Processada

Para assegurar que o processo de pré-processamento não resultou numa perda excessiva de informação relevante nas imagens, foram calculadas diversas métricas de qualidade, o MSE, o PSNR e o SSIM. Estas métricas foram avaliadas para todas as imagens do conjunto de dados, comparando as versões originais com as versões que já sofreram pré-processamento. A tabela 4.1 apresenta os valores obtidos das métricas de qualidade para o conjunto de dados anexado em I fornecido pela empresa GALP, após o processo de pré-processamento, que envolveu suavização com filtro bilateral seguida de aumento de contraste com CLAHE utilizando um *clip-limit* de 2 e *tiles* de tamanho 16×16).

Tabela 4.1: Métricas de qualidade para cada imagem pré-processada

Imagem	MSE	PSNR (dB)	SSIM
Imagem 1	40.30	32.08	0.9770
Imagem 2	20.50	35.01	0.9780
Imagem 3	5.70	40.57	0.9904
Imagem 4	7.34	39.47	0.9649
Imagem 5	13.14	36.94	0.9780
Imagem 6	23.71	34.38	0.9478
Imagem 7	17.16	35.79	0.9517
Imagem 8	9.05	38.56	0.9451
Imagem 9	23.98	34.33	0.9572
Imagem 10	7.24	39.53	0.9846
Imagem 11	23.66	34.39	0.9696
Imagem 12	9.77	38.23	0.9878
Imagem 13	18.01	35.58	0.9583
Imagem 14	14.78	36.43	0.9851
Imagem 15	8.73	38.72	0.9840
Imagem 16	5.99	40.36	0.9767

Relativamente à métrica MSE podemos observar que, de forma geral, o pré-processamento não introduziu alterações substanciais nas imagens, preservando a maioria dos detalhes visuais. A maioria das imagens apresenta valores de MSE abaixo de 25, indicando alta similaridade com as originais. Tal interpretação é justificada pelo facto de que um MSE de 25 corresponde a uma diferença média por pixel de $\sqrt{25} = 5$ níveis de intensidade, o que é relativamente pequeno comparado a 255.

Todas as imagens apresentam valores de PSNR superiores a 30 dB, indicando preservação da qualidade da imagem, no caso das imagens 3 e 16 temos valores superiores a 40 dB, significando que as diferenças são praticamente imperceptíveis ao olho humano.

Todos os valores de SSIM estão muito próximos de 1, o que significa que a estrutura, textura e contraste locais foram extremamente bem preservados, também a baixa variabilidade, dado que o desvio padrão da métrica SSIM é de somente 0.0149, mostra que a qualidade estrutural foi consistentemente mantida em todas as imagens.

Os resultados obtidos confirmam a eficácia da combinação das técnicas de pré-processamento

adotadas, uma vez que permitiu a redução de ruído potencialmente prejudicial à deteção das partículas, sem introduzir alterações substanciais nos elementos de interesse da imagem.

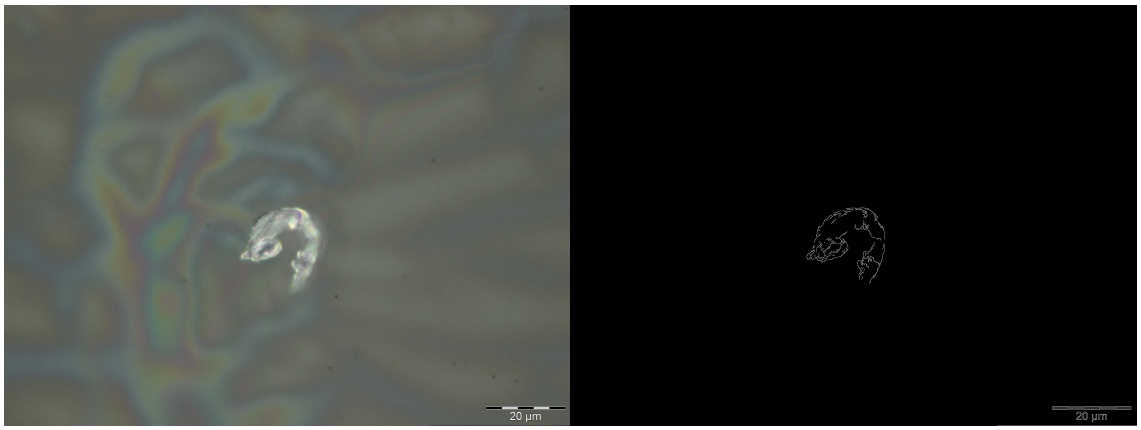
4.6 Segmentação

4.6.1 Deteção das Bordas das Partículas Utilizando Canny

Concluído o pré-processamento e validada a sua eficácia, as partículas encontram-se já destacadas e isoladas, permitindo avançar para a etapa seguinte, a deteção das suas bordas.

Primeiramente foi testado, para a imagem 4.8a, a primeira imagem do conjunto de dados, os métodos descritos em 3.7.1.3. Aplicando em primeiro lugar o método de limiarização de Otsu, seguidamente o método de Hossain e por último o de Zhao.

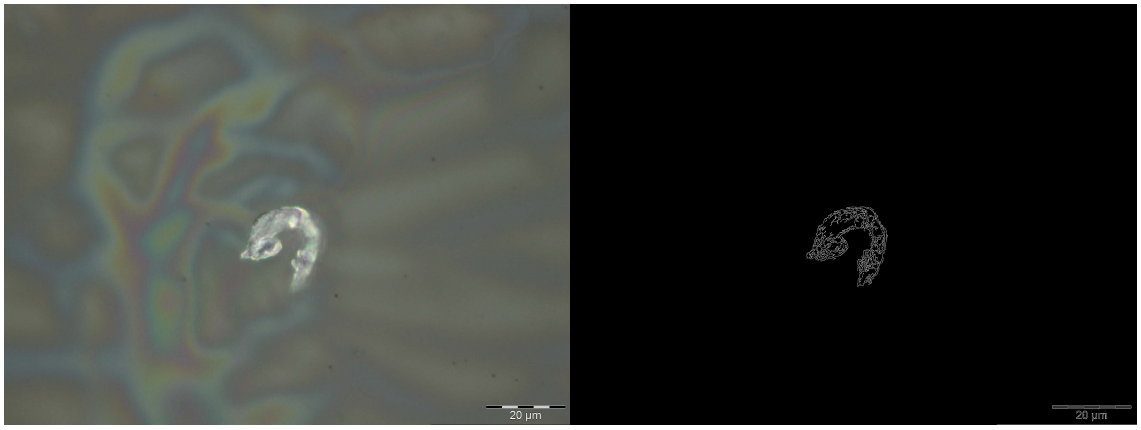
Para o primeiro método de limiarização obteve-se como limiar inferior o valor 53 e como limite superior o valor 106, podemos observar este resultado na imagem 4.8b encontrada na figura abaixo:



(a) Imagem Original (b) Bordas detetadas por Otsu

Figura 4.8: Visualização das bordas detetadas pelo método de Canny utilizando o critério de Otsu para definir os limiares.

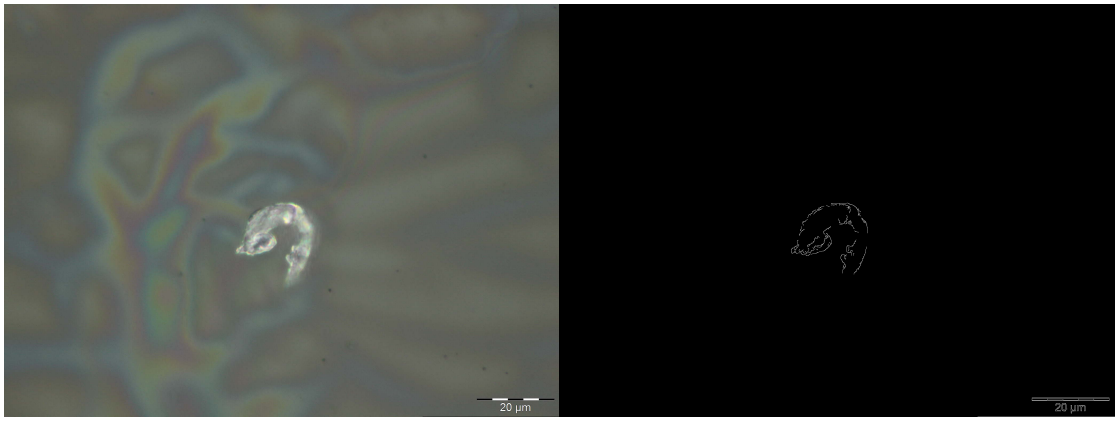
Para o método de limiarização de Hossain obteve-se como limiar inferior o valor 38 e como limite superior o valor 70, podemos observar este resultado na imagem 4.9b encontrada na figura abaixo:



(a) Imagem Original (b) Bordas detetadas por Hossain

Figura 4.9: Visualização das bordas detetadas pelo método de Canny utilizando o critério de Hossain para definir os limiares.

Para o método de limiarização de Zhao obteve-se como limiar inferior o valor 67 e como limite superior o valor 136, podemos observar este resultado na imagem 4.10b encontrada na figura abaixo:



(a) Imagem Original (b) Bordas detetadas por Zhao

Figura 4.10: Visualização das bordas detetadas pelo método de Canny utilizando o critério de Zhao para definir os limiares.

A imagem 4.10b, embora apresente uma estrutura simples, apresenta claramente descontinuidade no contorno externo; é o resultado menos contínuo e mais fragmentado dos 3 métodos. Deste modo, a deteção acaba por não ser suficientemente detalhada, sendo descartada nesta análise.

Embora a imagem 4.8b possa parecer visualmente mais limpa à primeira vista, devido a apresentar menos ruído interno e ter contornos relativamente definidos, essa limpeza resulta, na verdade, de uma simplificação excessiva da estrutura. O método de Otsu, por ser baseado na limiarização global com otimização do histograma 3.7.1.3, tende a eliminar variações mais subtis no contorno exterior. Como consequência detalhes importantes da

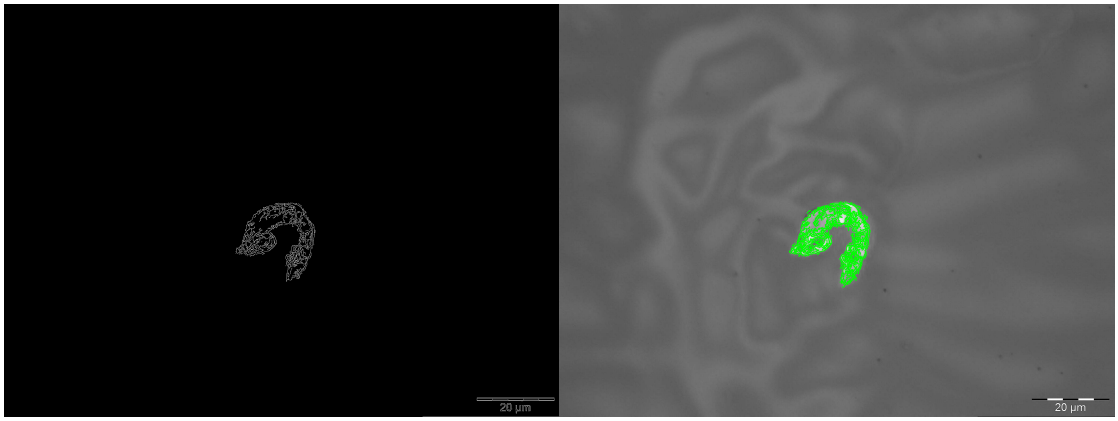
geometria real da estrutura, como variações muito reduzidas na curvatura ou pequenas ramificações, acabam por ser descartados.

Por outro lado, a imagem 4.9b mantém não só os contornos internos, mas também representa com maior detalhe e continuidade o contorno externo da estrutura. Apesar de apresentar mais ruído interno, apenas estamos interessados na delimitação dos contornos externos, por isso, o ruído interno torna-se irrelevante. O mais importante para conseguirmos determinar os dois pontos mais distantes é que o contorno verdadeiro da estrutura não seja simplificado, permitindo uma análise morfológica mais fiel à geometria real observada na imagem original. Assim, como o foco da análise está em capturar a forma autêntica e detalhada da borda externa, a imagem 4.9b revela-se a escolha mais adequada. Por observação atenta da imagem 4.9a, é possível verificar que certas demarcações da partícula se encontram desfocadas. Estas faltas de nitidez comprometem a transição de intensidade nessas regiões, o que pode resultar em que algumas demarcações não tenham sido identificadas corretamente pelo método de Canny. Como resultado, analisada a figura 4.9, se comparada a imagem original com imagem correspondente às bordas detetadas, verifica-se que quase todas as delimitações da partícula são visíveis, embora possamos reparar em algumas pequenas falhas de continuidade.

Para as restantes imagens testadas, o critério de Hossain, para determinar os limiares, também foi o que apresentou melhores resultados.

4.6.1.1 Deteção de Contornos

Apesar de inicialmente ter sido realizada a deteção dos contornos externos das partículas logo após a aplicação da deteção de bordas, os resultados obtidos revelaram-se insatisfatórios pois a maior parte das partículas apresenta elevado nível de detalhe interno. Tal limitação deve-se ao facto de a deteção de bordas, como já evidenciado anteriormente, não ser um processo perfeito devido a estar sujeita a diversos fatores, como as condições de iluminação, a qualidade da imagem capturada e o ruído presente. Consequentemente, muitas partículas não apresentam as suas bordas completamente fechadas após a deteção, o que compromete a eficácia da extração direta dos seus contornos, no caso da imagem 4.9a deveu-se ao desfoque mencionado. Esta fragilidade conduz à identificação de múltiplos contornos internos, resultando frequentemente na deteção de diversas "partículas"que, na realidade, correspondem apenas a uma única estrutura mal segmentada. Abaixo temos um exemplo desta ocorrência:



(a) Bordas detetadas (b) Contornos externos detetados

Figura 4.11: Visualização dos contornos detetados sem a aplicação de dilatação.

Face a este problema é necessário aplicar operações morfológicas, como é o caso da dilatação 3.8.1 e da erosão 3.8.2. Considerando que o objetivo é promover a conexão de fragmentos de borda dispersos ao redor das partículas, a aplicação da dilatação revela-se a abordagem mais adequada. Esta operação expande as regiões brancas da imagem binária, preenchendo pequenas lacunas e unindo segmentos próximos, favorecendo a formação de contornos fechados. Por outro lado, a utilização da erosão teria um efeito oposto, eliminando pormenores e estreitando as regiões existentes, o que poderia resultar na supressão adicional de contornos já incompletos, agravando o problema de fragmentação das partículas.

4.6.1.2 Dilatação

Vários tamanhos de *kernel* foram testados, assim como 1 e 2 interações, isto é, o número de vezes que a operação de dilatação é aplicada consecutivamente à imagem, aumentando progressivamente a expansão das delimitações brancas. Posteriormente à aplicação destas variações, procedeu-se à deteção de contornos visando avaliar o impacto de cada combinação na qualidade da segmentação. Esta análise permitiu identificar os parâmetros que, visualmente, melhor promovem a continuidade dos contornos. O impacto das diferentes variações nos parâmetros morfológicos pode ser visualizado na figura abaixo, onde se comparam os resultados obtidos para cada configuração testada:

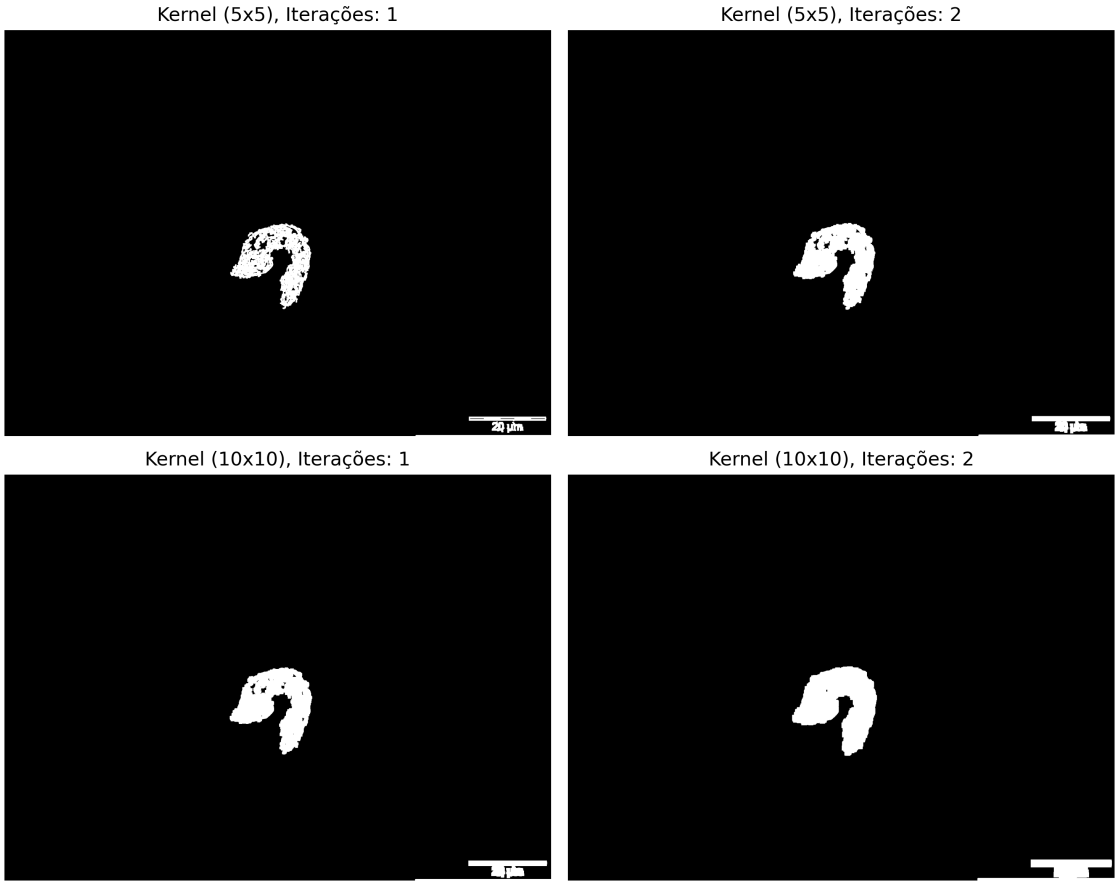
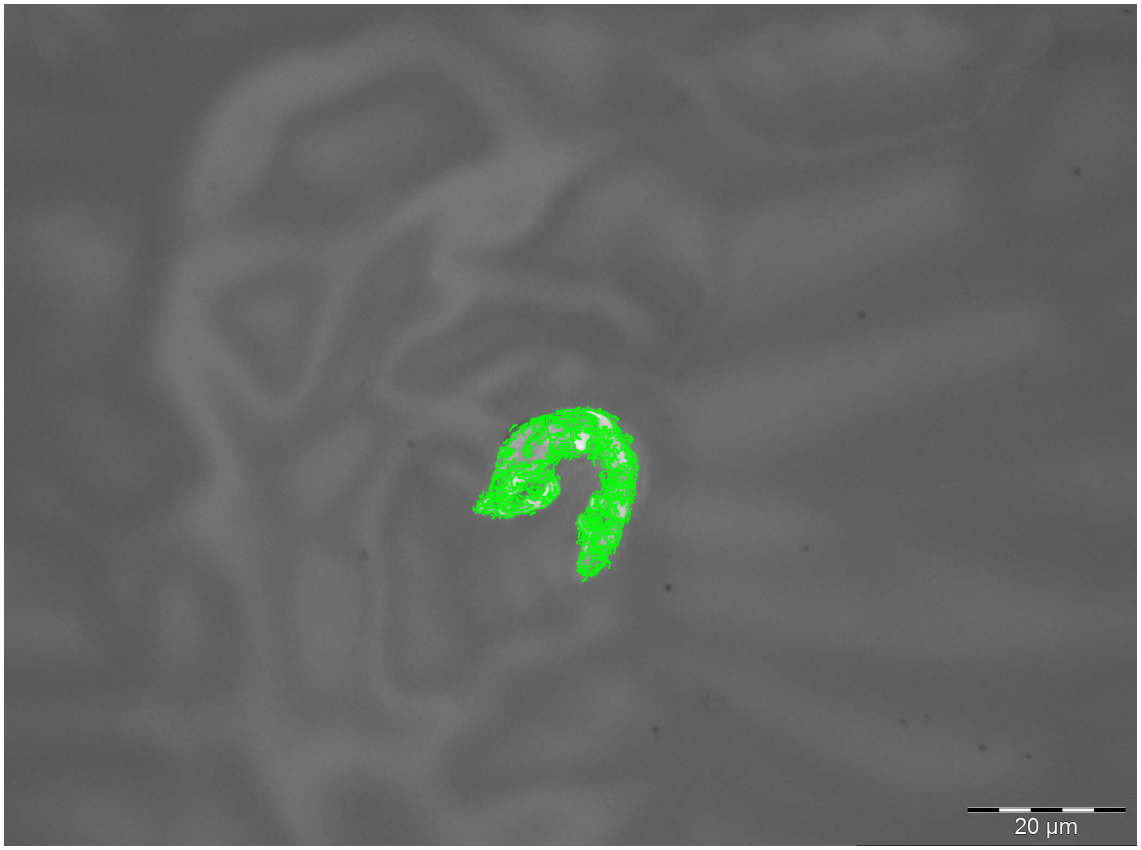


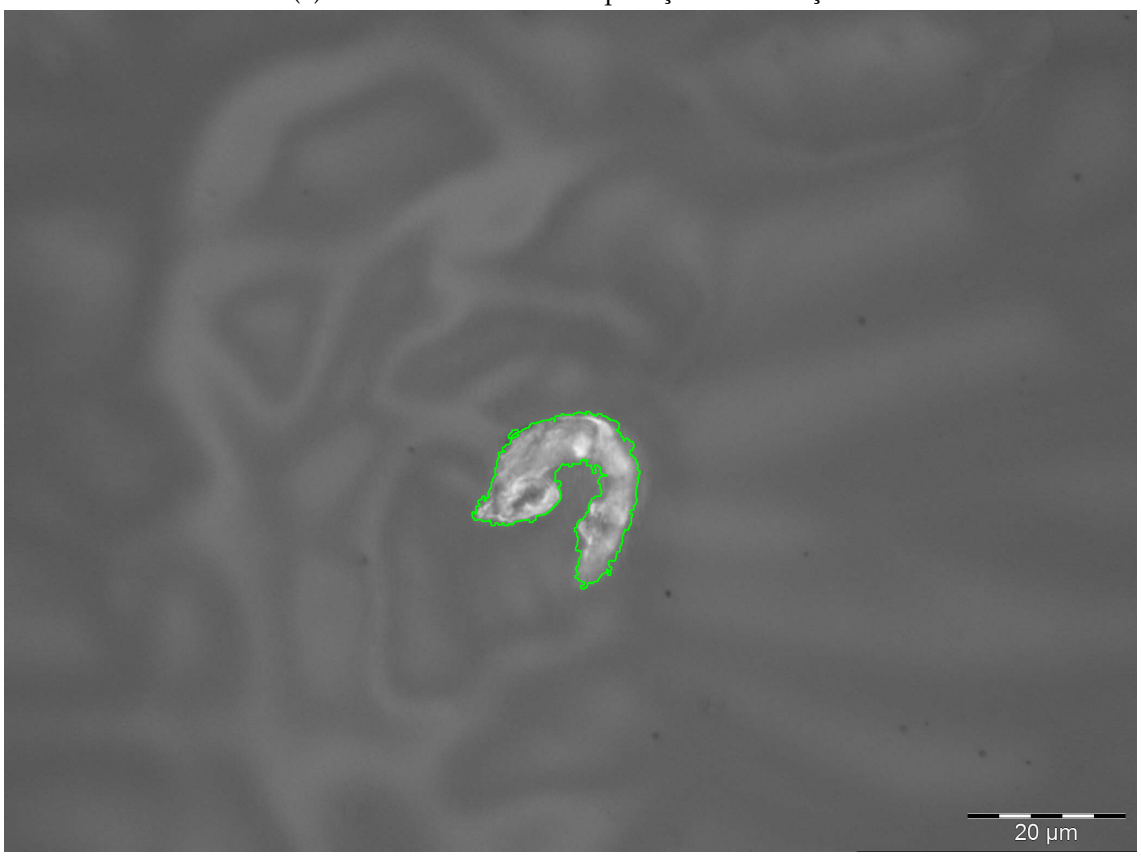
Figura 4.12: Efeito visual da variação do tamanho do *kernel* e do número de iterações.

A Figura 4.12 apresenta uma comparação visual dos resultados da operação de dilatação morfológica, variando o tamanho do *kernel* e o número de iterações. Observa-se que, à medida que se aumentam ambos os parâmetros, a partícula se torna progressivamente mais preenchida, com as descontinuidades ao longo do contorno a serem gradualmente colmatadas. Esta característica é desejável em casos onde a segmentação inicial apresenta falhas ou ruído nas fronteiras das partículas.

Com base nos resultados apresentados, conclui-se que qualquer uma das quatro combinações testadas é potencialmente adequada para esta imagem em particular. A título ilustrativo, é apresentado abaixo o resultado da aplicação da configuração mais simples, um *kernel* de 5×5 com uma iteração, a qual já se revela eficaz na recuperação da forma da partícula, produzindo um resultado visualmente satisfatório:



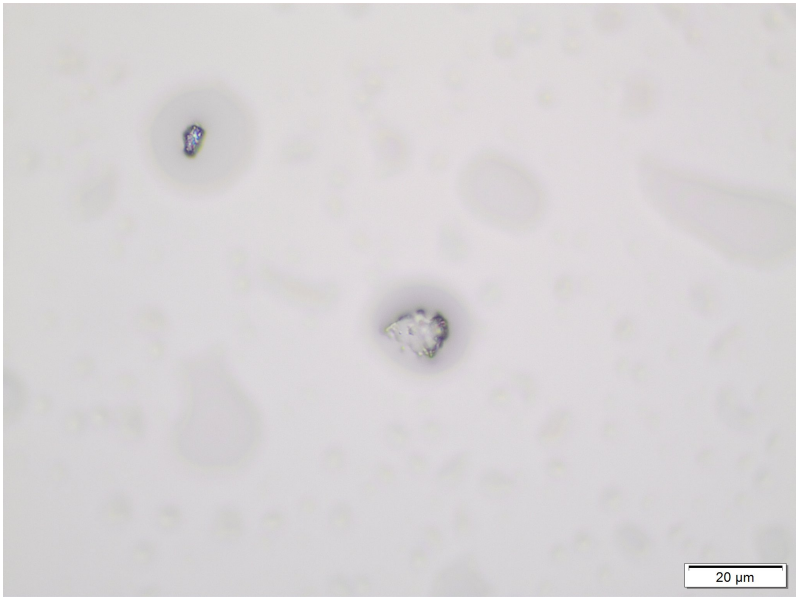
(a) Bordas detetadas sem aplicação de dilatação



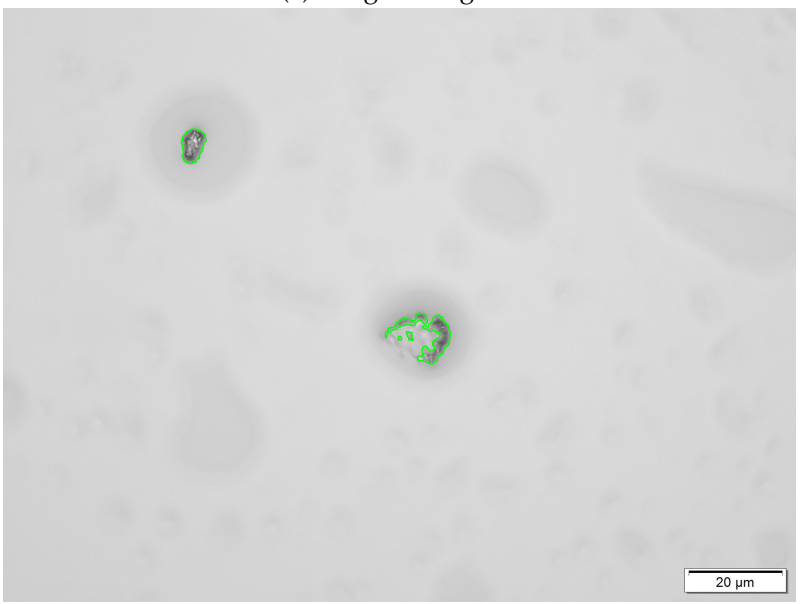
(b) Bordas detetadas com aplicação de dilatação

Figura 4.13: Visualização dos contornos detetados sem e com a aplicação de dilatação.

No entanto, quando estas mesmas configurações são aplicadas a imagens com maior complexidade morfológica, nomeadamente partículas com contornos fragmentados ou regiões próximas entre si, a aplicação de uma dilatação menos agressiva pode levar à deteção incorreta de múltiplas partículas onde existe, na realidade, apenas uma. Abaixo apresenta-se um exemplo (imagem 5 do anexo I) no qual a utilização de um *kernel* de 5×5 combinado com uma única iteração não permite unir adequadamente os fragmentos da partícula, comprometendo a segmentação:



(a) Imagem Original

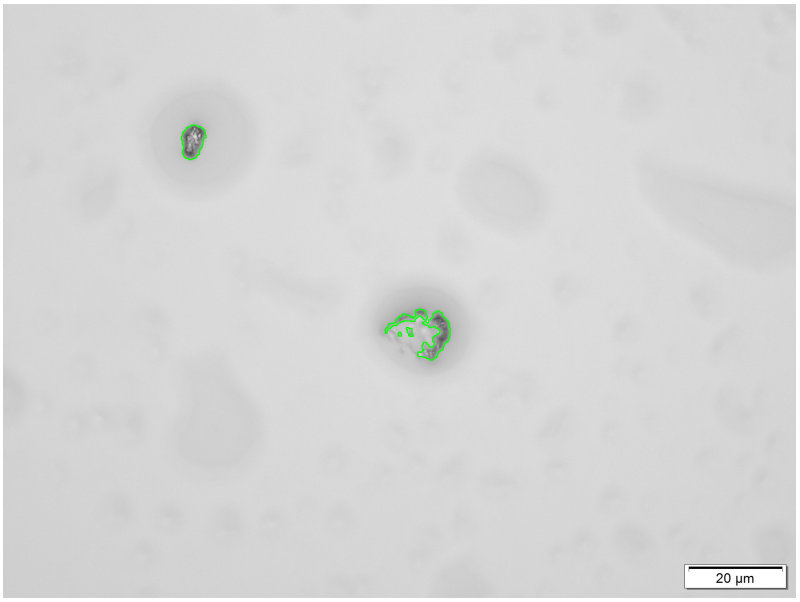


(b) Bordas detetadas com dilatação

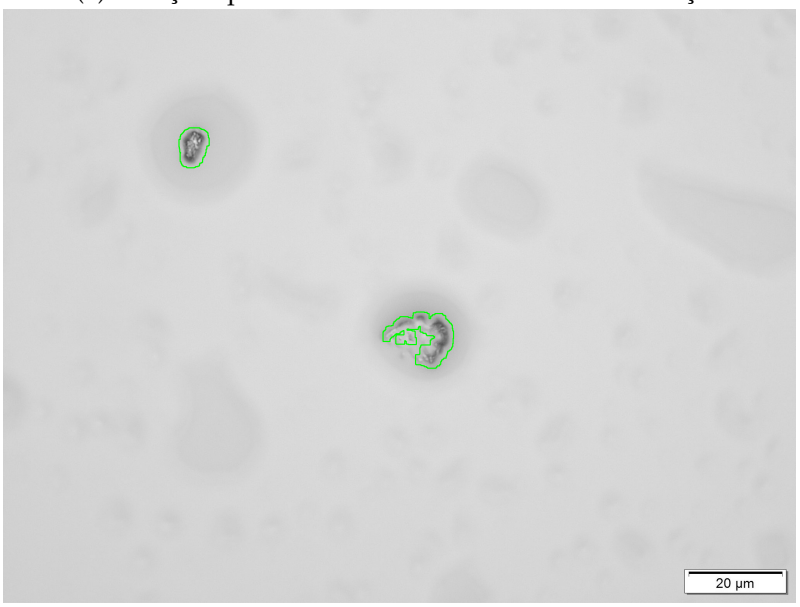
Figura 4.14: Visualização dos contornos detetados aplicando dilatação com um *kernel* de 5×5 e uma iteração.

Embora a configuração com *kernel* de 5×5 e uma iteração se tenha revelado adequada

na imagem anterior (4.13b), onde a partícula apresentava contornos bem definidos, a sua aplicação à imagem 4.14a não produz resultados satisfatórios. A partícula central de maiores dimensões, apesar de ser visualmente identificável como uma estrutura única e coesa, encontra-se parcialmente envolvida por uma região difusa, uma espécie de névoa, que se acredita ser óleo lubrificante. Esta névoa resulta numa transição pouco abrupta entre a partícula e o fundo, comprometendo a eficácia da dilatação com parâmetros reduzidos. Como consequência, não são unidas as diferentes partes da partícula, levando à sua deteção como dois objetos distintos (4.14b), e menores do que se acredita ser, quando na realidade se trata de uma única partícula. Abaixo apresenta-se a comparação utilizando uma operação de dilatação com *kernel* de 10×10 e duas iterações:



(a) Deteção aplicando um *kernel* de 5×5 e uma iteração



(b) Deteção aplicando um *kernel* de 10×10 e duas iterações

Figura 4.15: Visualização dos contornos detetados aplicando diferentes tamanhos de *kernel* e número de iterações.

Comparando a imagem 4.15b com a versão anterior 4.15a, é possível observar que, embora a partícula central ainda não tenha sido detetada na sua totalidade, o resultado é visivelmente mais satisfatório. Tal como sucede com o olho humano, também o algoritmo de segmentação é influenciado pela névoa difusa que envolve a partícula, o que dificulta a perceção dos seus contornos completos. Ainda assim, nesta nova versão, a partícula é reconhecida como uma única entidade coesa, ao contrário da situação anterior em que foi segmentada em dois fragmentos.

Este resultado é particularmente relevante do ponto de vista da análise morfológica,

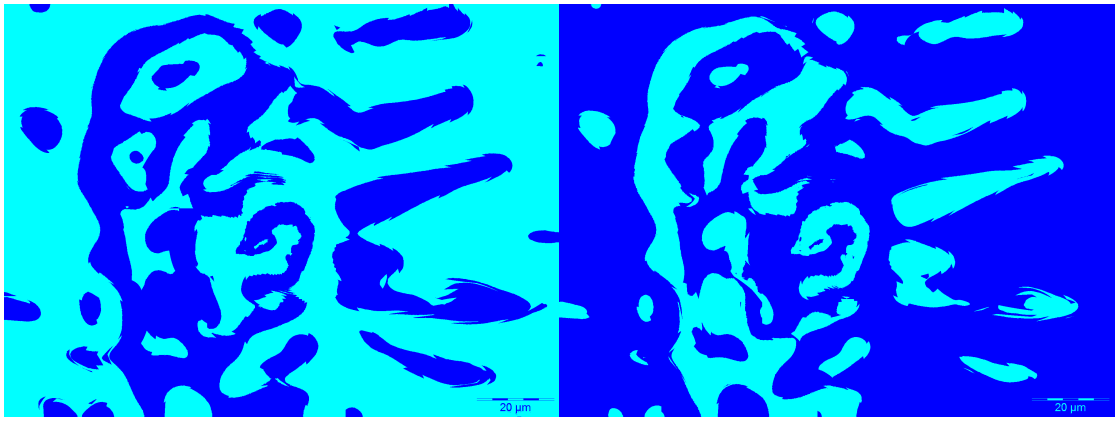
uma vez que a partícula, sendo considerada como um único objeto, permite uma estimativa mais fiável de propriedades geométricas. Apesar de essa falha afetar ligeiramente a estimativa dos dois pontos mais distantes, dado que uma das extremidades não é totalmente captada, acredita-se que o impacto é relativamente reduzido pois a maior parte da estrutura foi corretamente delineada e os pontos extremos estimados continuam a fornecer uma aproximação útil da dimensão máxima da partícula. Ainda que a medida resultante possa subestimar ligeiramente o comprimento real, o grau de erro é, neste caso, aceitável para os objetivos pretendidos.

4.6.2 Deteção das Bordas das Partículas Utilizando *Clustering*

Na etapa de segmentação, foi implementado o método de *clustering K-means*, explorando diferentes formas de aplicação. Numa primeira abordagem, o método foi testado em imagens convertidas para escala de cinza, nas imagens previamente processadas com os parâmetros definidos no capítulo anterior. Posteriormente, o *K-means* foi também aplicado a imagens convertidas para o espaço de cores *L*a*b** e para o espaço HSV. Esta diversidade de testes teve como objetivo avaliar de que forma o espaço cromático e o pré-processamento influenciam a eficácia da segmentação.

Para a aplicação do algoritmo *K-means*, definiu-se um critério de paragem composto por um número máximo de 100 iterações e uma tolerância de 0.1, correspondente à variação mínima entre iterações sucessivas. O número de *clusters* foi fixado em $k = 2$, com o objetivo de separar a imagem em duas classes distintas: partículas e fundo. A inicialização dos centroides foi realizada por via de duas estratégias distintas. Na primeira, os centroides iniciais são escolhidos de forma a estarem mais afastados entre si, enquanto na segunda são escolhidos aleatoriamente no espaço dos dados. Para mitigar a influência da aleatoriedade, foram definidas 10 tentativas por execução, sendo selecionada a melhor solução entre elas. Este processo completo foi repetido 20 vezes, para reduzir ainda mais o impacto da aleatoriedade na convergência final do algoritmo.

Na figura 4.16 apresenta-se o resultado visual da segmentação, aplicada à imagem 4.8a, obtida com os parâmetros descritos.

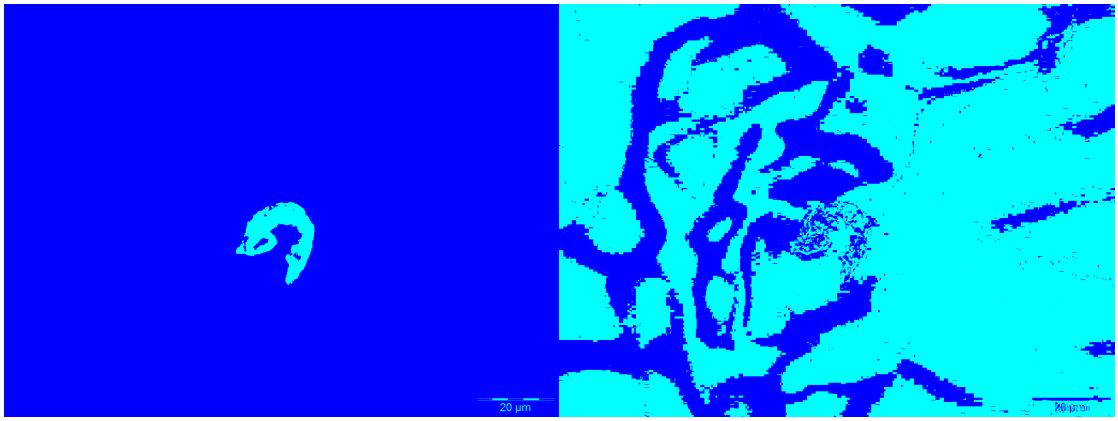


(a) *Clustering* por proximidade (b) *Clustering* por proximidade

Figura 4.16: Visualização da segmentação por *clustering* utilizando centroides de forma a estarem mais afastados (esquerda) e de modo aleatório (direita).

Apesar de a partícula de interesse (4.8b) aparentar ter sido em parte identificada, a segmentação também evidenciou um elevado número de regiões de ruído no fundo.

A seguir, apresentam-se os resultados obtidos para a segmentação da imagem nos espaços de cor *L*a*b** e HSV.



(a) *Clustering* no espaço *L*a*b** (b) *Clustering* no espaço HSV

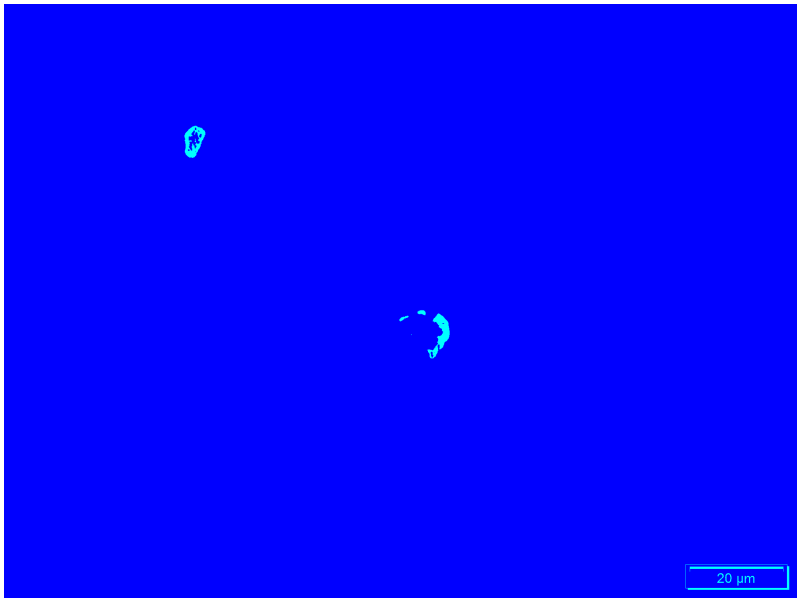
Figura 4.17: Visualização da segmentação por *clustering* feita em imagens no espaço de cores *L*a*b** e HSV.

A segmentação apresentou melhores resultados em imagens no espaço *L*a*b** do que em HSV. Isto deve-se a que *L*a*b** representa as cores de forma semelhante à perceção humana, captando de forma mais consistente diferenças visuais, enquanto HSV é mais sensível à iluminação.

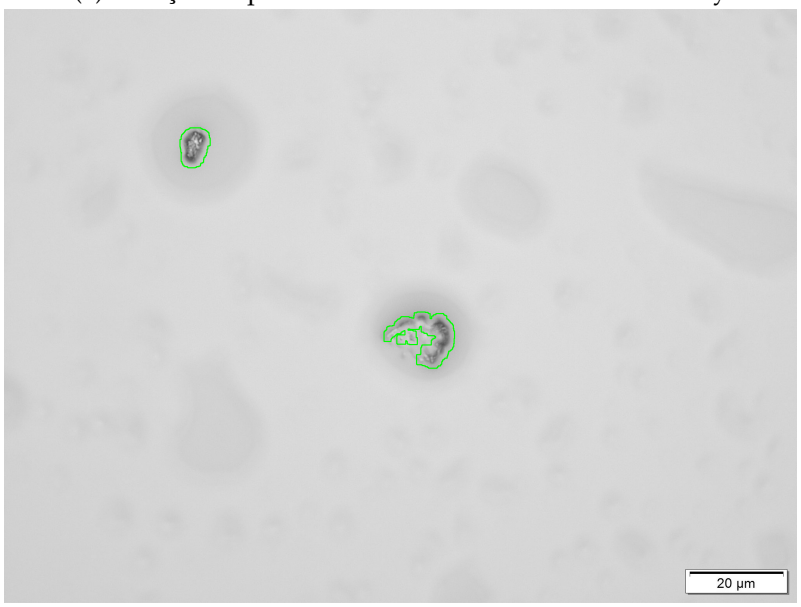
4.6.3 Comparação entre *Clustering* e Canny

Dada a diversidade de métodos testados na etapa de segmentação, numa primeira análise preliminar apenas dois se revelaram promissores: o método de *clustering K-means*, aplicado no espaço de cor *L*a*b**, e o método de deteção de contornos de Canny. Para

uma comparação mais clara entre ambos, a imagem 4.14a foi também segmentada recorrendo ao *clustering* por *K-means*. A Figura 4.18 mostra lado a lado os resultados dos dois métodos, permitindo uma avaliação direta da sua eficácia.



(a) Deteção de partículas utilizando o método de Canny



(b) Deteção de partículas utilizando *clustering* por *K-means*

Figura 4.18: Visualização do resultado da segmentação feita por dois processos diferentes.

Na Figura 4.18 apresentam-se os resultados obtidos em paralelo. A observação permite concluir que, embora o *clustering* em *L*a*b** tenha potencial em determinadas situações, o método de Canny revelou-se globalmente mais eficaz e consistente na deteção de partículas. Este resultado pode dever-se ao facto de o método de Canny oferecer maior flexibilidade na definição dos contornos das partículas, uma vez que os resultados podem ser refinados através da aplicação de operações morfológicas, bem como pelo ajuste

dos parâmetros das funções. Já no caso do clustering, a segmentação é mais rígida, isto é, quando os objetos surgem separados na imagem, não existe forma de os agrupar posteriormente de modo eficaz. Assim, optou-se por descartar o *clustering* em *L*a*b** e prosseguir o trabalho com base na segmentação através do método de Canny.

4.7 Conversão de Escala

Dada a presença da barra de escala nas imagens 4.1a, 4.8a e 4.14a, esta foi utilizada como referência para a conversão de unidades de comprimento de píxeis para micrómetros. Como se pode observar, a barra apresenta regiões compostas exclusivamente por píxeis pretos e brancos. Considerando que os píxeis brancos possuem níveis de intensidade próximos de 255 e os pretos próximos de 0, procedeu-se à filtragem com base nestes valores para isolar a área correspondente à barra, filtrando todos aqueles com intensidade de brilho superior a 240 ou inferior a 15.

Após esta filtragem, todos os píxeis foram convertidos em branco absoluto (255), para garantir uma região contínua e uniforme, facilitando a extração da sua extensão. Foi então criada uma máscara binária que contém somente os píxeis pertencentes à barra de escala, como ilustrado na imagem 4.19a. Importa salientar que o número que acompanha a barra de escala também é composto por píxeis brancos e pretos, com níveis de intensidade semelhantes aos da própria barra. Por esse motivo, não é viável simplesmente contar o número total de píxeis brancos na máscara criada após a filtragem de intensidade, uma vez que essa contagem incluiria também os píxeis pertencentes ao número. Para ultrapassar essa limitação, foi necessário aplicar a Transformada de Hough, de forma a identificar apenas os segmentos que correspondem a linhas perfeitamente horizontais, características da barra de escala, excluindo assim os elementos textuais.

(a) Máscara criada para isolar a barra de escala

(b) Linha detetada pela Transformada de Hough

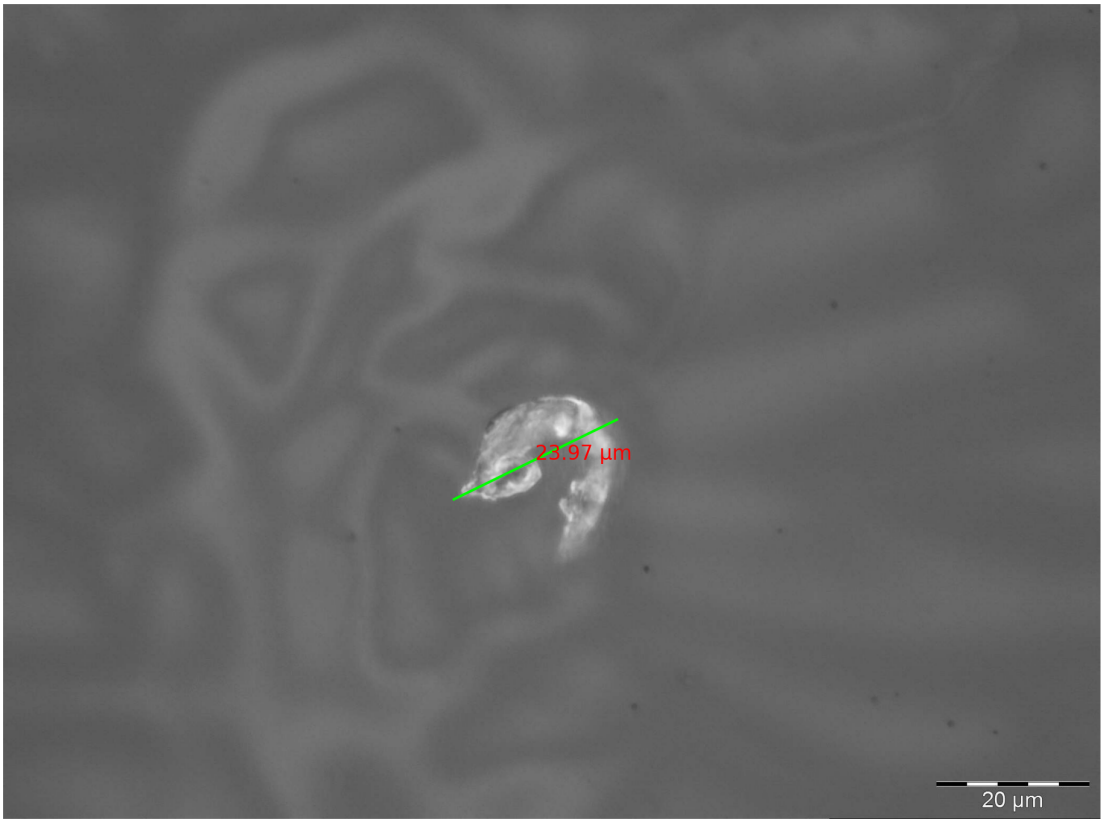
Figura 4.19: Visualização da máscara criada apenas contendo a barra de escala (cima) e da linha detetada pela Transformada de Hough (baixo).

A partir da figura 4.19, pode concluir-se que o comprimento da barra de escala foi corretamente detetado. Esta conclusão é suportada pela imagem 4.19b, onde a linha vermelha sobreposta à imagem representa a deteção obtida por meio da Transformada de Hough, coincidindo com a extensão real da barra.

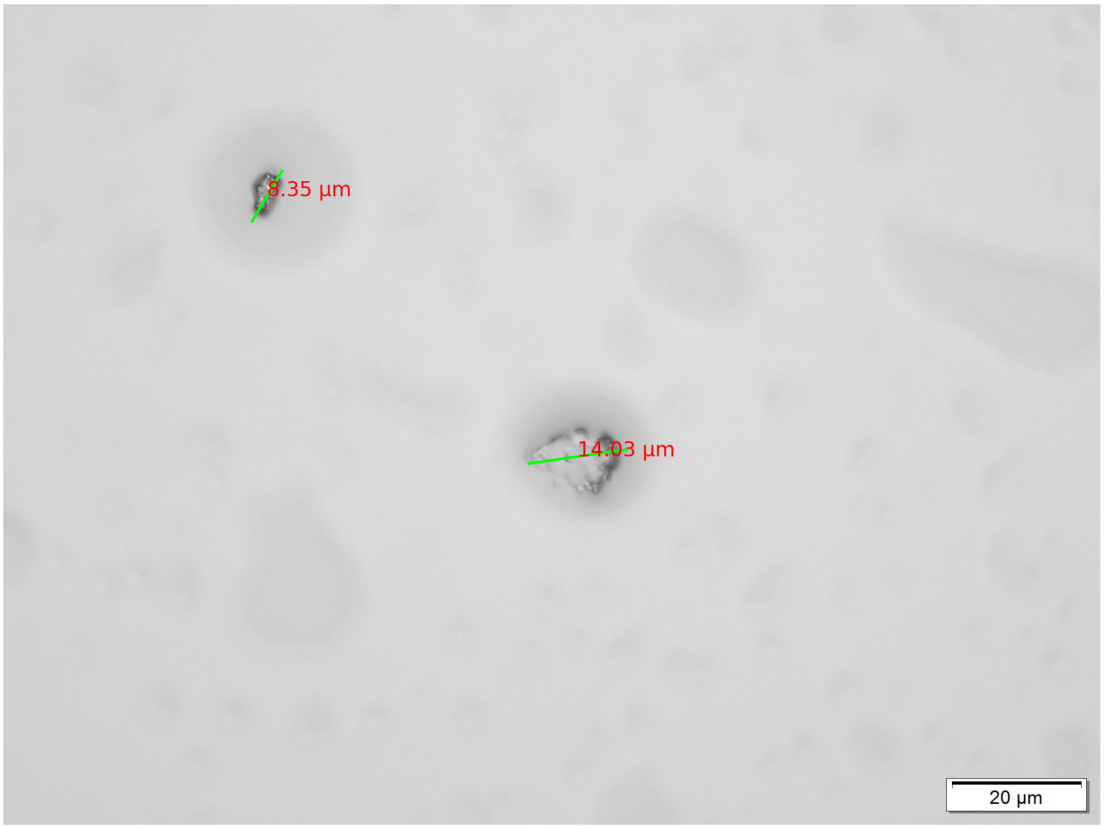
4.7.1 Cálculo do Diâmetro das Partículas

Por fim, resta medir o diâmetro da partícula em píxeis e, posteriormente, convertê-lo para micrómetros. Considerando, por exemplo, a imagem 4.19b, onde 20 micrómetros correspondem a 262 píxeis, é possível estabelecer o fator de conversão necessário. O único *input* exigido ao utilizador é o valor real associado à barra de escala presente na imagem.

Para o cálculo do diâmetro, foram determinadas todas as distâncias entre os pares de pontos pertencentes à partícula segmentada, selecionando-se a maior dessas distâncias. Este valor representa o diâmetro máximo da partícula, e a sua conversão para unidades reais é feita através do fator previamente calculado. Abaixo temos dois exemplos de deteção do diâmetro maior, referente às imagens 4.8a e 4.14a:



(a) Representação do maior diâmetro de 4.8a



(b) Representação do maior diâmetro de 4.14a

Figura 4.20: Visualização dos diâmetros das partículas referentes às imagens 4.8a (cima) e 4.14a (baixo).

Na figura acima (4.20), o segmento de reta verde representa o maior diâmetro, enquanto os valores em vermelho indicam o seu comprimento real, em micrómetros. Na imagem 4.20b é possível verificar que, apesar da deteção incompleta anteriormente mencionada, causada pela falha na união de algumas regiões da partícula (4.15b), o cálculo do maior diâmetro não foi significativamente comprometido. O traçado do diâmetro atravessa a estrutura de forma coerente com a sua extensão, evidenciando que, medindo o diâmetro diretamente na imagem binária resultante da aplicação da operação morfológica de dilatação, mesmo com uma segmentação não totalmente contínua, os pontos mais distantes da partícula foram corretamente identificados.

4.7.2 Validação dos Resultados

A validação dos resultados obtidos em laboratório constitui uma etapa essencial para aferir a fiabilidade do algoritmo desenvolvido. Idealmente, este processo deveria ter sido realizado através da comparação direta com relatórios laboratoriais de referência, produzidos por técnicos especializados, relativamente às mesmas amostras analisadas nesta dissertação. A disponibilização desses relatórios permitiria validar de forma automática a contagem e a medição da dimensão das partículas identificadas pelo algoritmo, assegurando um processo de avaliação mais objetivo e rigoroso.

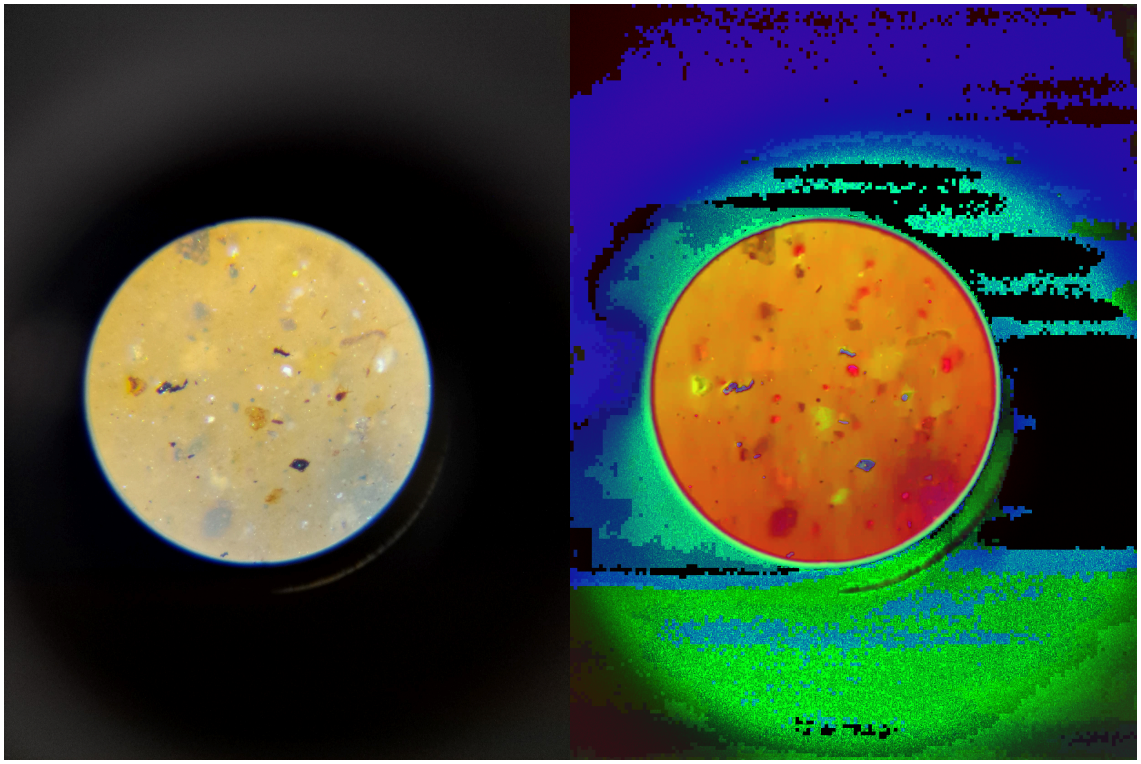
O conjunto de dados disponível era composto por apenas 16 imagens laboratoriais. Para definir os parâmetros de processamento, foi selecionada a imagem mais desafiante (4.14a), caracterizada por a partícula se eoncontrar parcialmente coberta por óleo lubrificante, de modo a assegurar maior generalização. Os parâmetros ajustados com base nesta imagem revelaram-se eficazes em 14 dos 15 casos restantes, permitindo a correta deteção e medição das partículas. A única exceção ocorreu numa amostra na qual a cor de uma das partículas presentes era muito semelhante à cor do fundo.

No entanto, esses relatórios não estavam disponíveis no âmbito deste trabalho. Assim, a validação foi efetuada indiretamente, através de verificação visual realizada em conjunto com engenheiros químicos da GALP, que confirmaram a correspondência geral entre as partículas identificadas pelo algoritmo e as observáveis nas imagens. Embora este método proporcione uma indicação qualitativa da fiabilidade da deteção, introduz alguma subjetividade na avaliação e limita a robustez da validação quantitativa.

4.8 Passagem para o Terreno

4.8.1 Introdução

Embora o algoritmo desenvolvido tenha demonstrado bom desempenho na deteção de partículas metálicas em imagens de ferrografia, a sua aplicação em amostras recolhidas no terreno não gerou os resultados de interesse. Este insucesso deve-se, na maioria, ao facto de o processo de deteção inicial basear-se na conversão da imagem original para tons de cinzento, seguida da aplicação do operador de Canny e operações morfológicas, eficiente em imagens de ferrografia devido ao fundo ser praticamente homogéneo e as partículas metálicas se destacarem nitidamente pelo contraste. No entanto, nas imagens de terreno, a variabilidade cromática, a presença de texturas complexas, o ruído visual e as alterações de iluminação criam regiões de elevado contraste que não correspondem às partículas de interesse, levando o algoritmo a detetar preferencialmente regiões mais escuras, contrariando o objetivo pretendido, uma vez que as partículas metálicas, por serem refletoras, tendem a apresentar tons mais claros ou brancos em imagens capturadas sem grandes ampliações, sendo essas as que efetivamente se pretendem identificar. Como consequência, o algoritmo tende a detetar erroneamente regiões mais escuras, simplesmente porque estas áreas apresentam transições abruptas de intensidade realçadas pelo aumento do contraste e operador de Canny, que se baseia na análise do gradiente de intensidade da imagem para identificar transições abruptas entre zonas claras e escuras, que geralmente correspondem a bordas; este processo envolve uma etapa de limiarização, onde apenas os gradientes cuja magnitude ultrapassa um determinado valor são considerados bordas reais. Em imagens com elevado contraste local, como as de terreno, regiões escuras rodeadas por áreas mais claras tendem a gerar gradientes fortes, excedendo esse limiar e sendo assim erroneamente classificadas como bordas relevantes. Como resultado o algoritmo acaba por detetar contornos em torno das áreas mais escuras, mesmo que estas não correspondam a partículas metálicas. Na tentativa de resolver este problema surgiu a hipótese de explorar outros espaços de cor, nomeadamente o HSV. Abaixo temos um exemplo da representação de uma das duas imagens que compõem o conjunto de dados no espaço de cores HSV:



(a) Imagem de terreno original (b) Imagem convertida em HSV

Figura 4.21: Visualização de uma imagem de terreno original (esquerda) e a sua conversão para o espaço de cores HSV.

Ao observar a imagem original (4.21a), a amostra apresenta iluminação relativamente homogénea, sendo apenas visível uma mancha acinzentada no canto inferior direito que levanta dúvidas quanto à uniformidade da luz no momento da captura. A conversão para o espaço de cores HSV (4.21b) confirma essa suspeita, observa-se que, à medida que se desce na imagem, a saturação e o valor do fundo diminuem progressivamente, o que evidencia que a iluminação não foi uniforme durante a aquisição. Esta falta de homogeneidade luminosa pode introduzir interferências em análises posteriores baseadas em limiares de intensidade ou cor, reforçando a importância de assegurar condições consistentes de iluminação no processo de captura.

4.8.2 Escolha dos Intervalos de valores

No OpenCV, o espaço de cores HSV é representado de forma diferente do modelo tradicional. O matiz (H) varia de 0 a 180 em vez de 0 a 360 graus, pois o OpenCV utiliza um intervalo reduzido para otimização computacional, a saturação (S) e o valor (V) variam de 0 a 255, permitindo um maior nível de precisão quando há necessidade de manipulação das cores.

A escolha dos intervalos de valores para a deteção das partículas metálicas baseia-se na característica visual já mencionada dessas partículas e na forma como elas são representadas no espaço de cor HSV. Os intervalos escolhidos foram:

```
lower_hsv = np.array([0, 10, 215])
upper_hsv = np.array([180, 74, 255])
```

Esses valores foram definidos com base nas seguintes considerações:

- **Matiz (H)**: O intervalo escolhido de 0 a 180 cobre toda a faixa de matizes disponível no modelo HSV do OpenCV. Como as partículas metálicas podem refletir a luz de diferentes formas, abrangendo matizes variados, optou-se por um intervalo amplo para garantir que diferentes tonalidades fossem consideradas na segmentação.
- **Saturação (S)**: A faixa entre 13 e 74 foi escolhida porque as partículas metálicas possuem uma coloração geralmente pouco saturada devido à sua superfície refletora, se fossem considerados valores muito baixos de saturação poderiam incluir elementos indesejados como sombras, enquanto valores muito altos poderiam descartar regiões de interesse.
- **Valor (V)**: O intervalo entre 215 e 255 foi escolhido porque as partículas metálicas são tipicamente brilhantes devido, mais uma vez, à sua alta reflexividade, a restrição para valores elevados de V ajuda a minimizar a inclusão de objetos mais escuros ou sombras na segmentação.

Antes da obtenção destes intervalos, começaram por ser considerados intervalos mais abrangentes, no entanto, a luz na imagem não é uniforme, o que causou sombras e reflexos que dificultaram a segmentação das partículas metálicas. Para contornar esse problema, foram analisadas diferentes regiões da imagem ao passar o cursor sobre áreas onde se sabia que havia partículas metálicas, permitindo observar os valores HSV nessas regiões, sendo então definida uma margem adequada para os intervalos de segmentação. Abaixo apresenta-se o resultado da segmentação obtida, complementado com a aplicação sucessiva das operações morfológicas de dilatação e erosão:

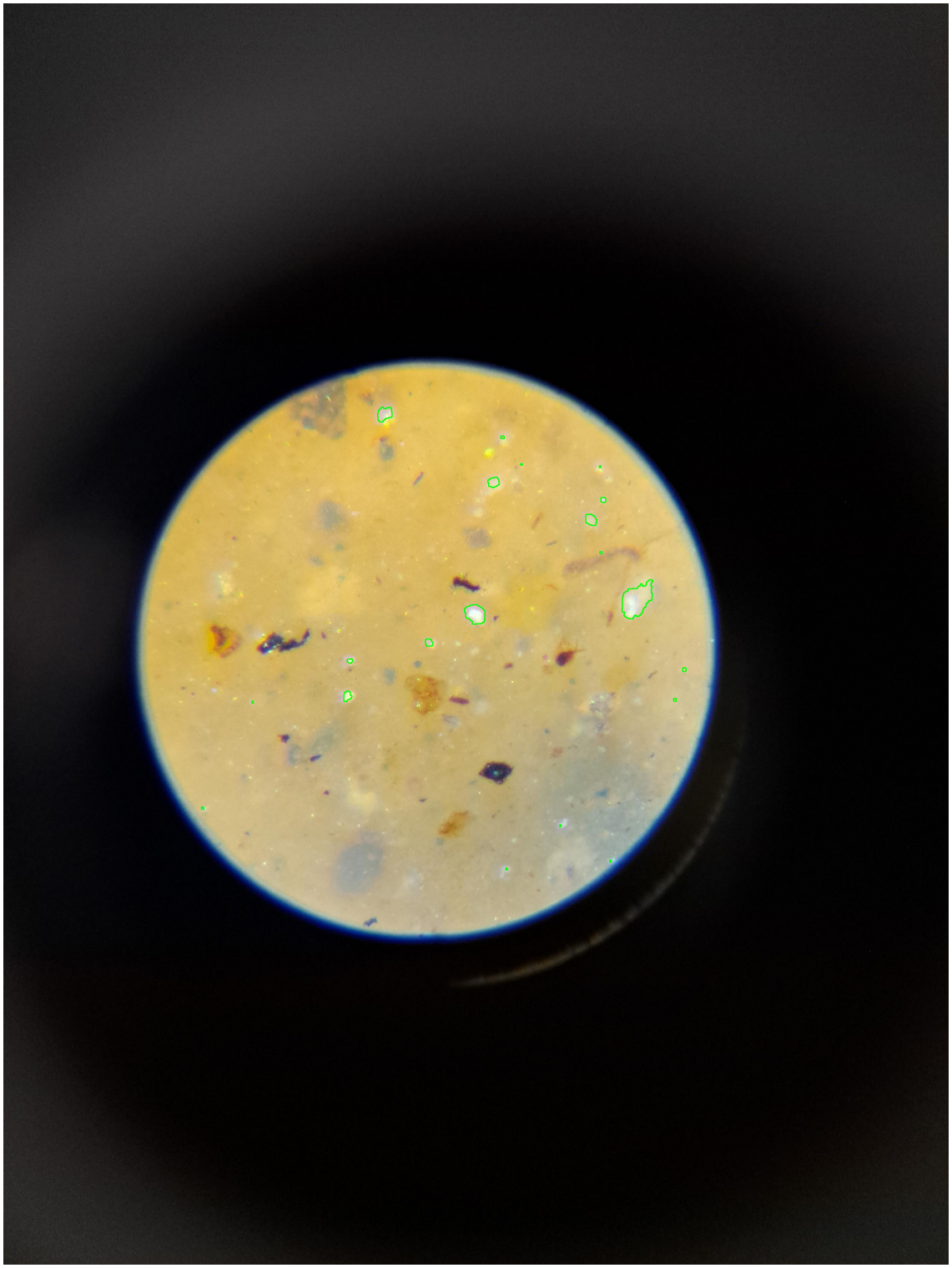


Figura 4.22: Segmentação com os intervalos escolhidos

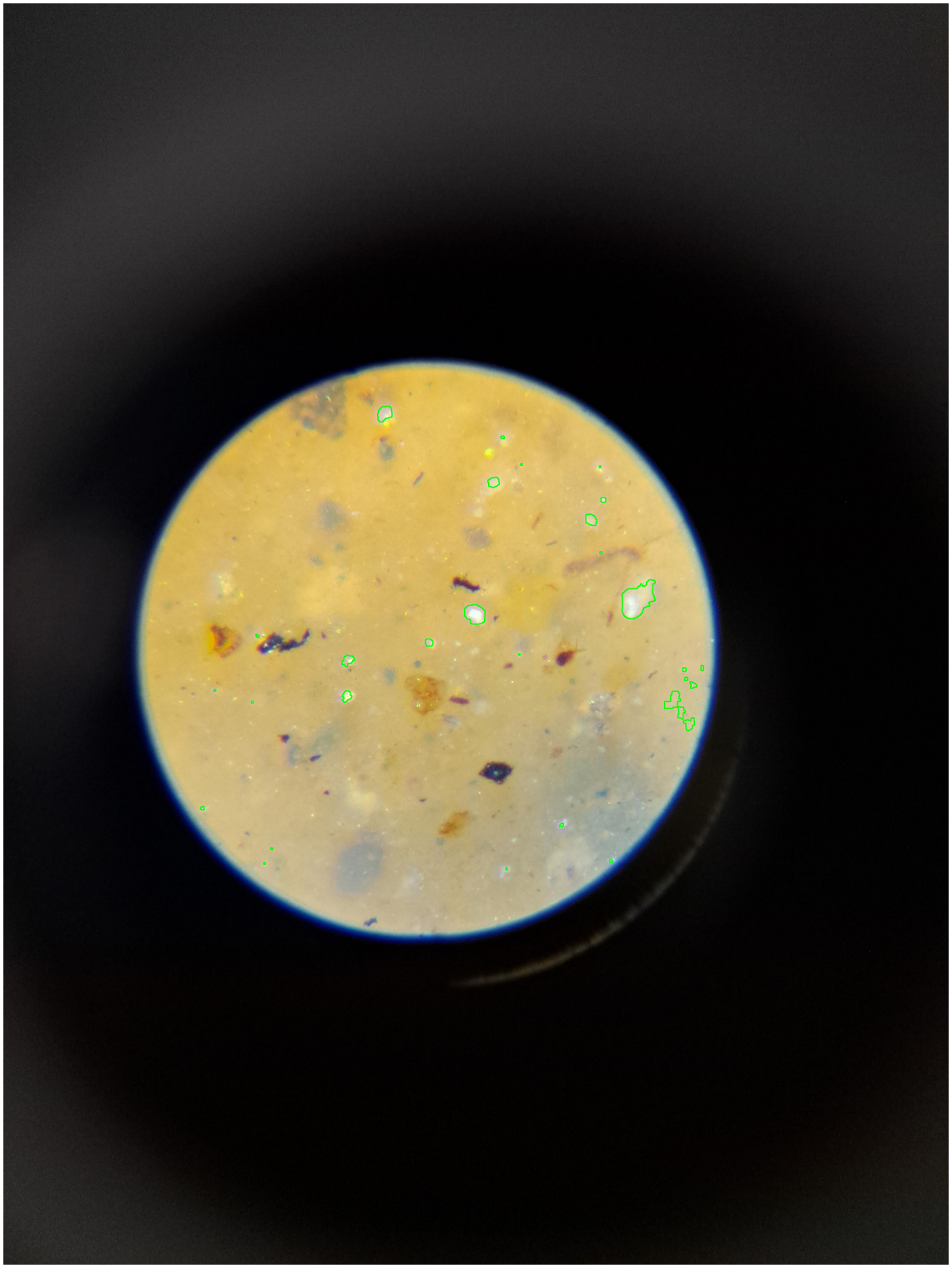


Figura 4.23: Segmentação com intervalos mais abragentes

Como podemos observar na imagem 4.23, a utilização de um intervalo ligeiramente mais abrangente para o canal de luminância, de 220 a 255 em vez de 215 a 255, resulta numa segmentação menos precisa. Na parte direita da imagem, é visível a deteção de alguns pigmentos que, pela sua morfologia e intensidade, não correspondem a partículas

brancas (potencialmente metálicas), sendo antes resultado de reflexos ou variações locais na iluminação. Este efeito evidencia, mais uma vez, a influência da falta de uniformidade na luz incidente durante a captura da imagem, fator que pode comprometer a fiabilidade da segmentação baseada apenas em limiares fixos no espaço de cores HSV.

Tanto a deteção de contornos como a determinação do diâmetro máximo foram realizadas de forma análoga ao processo aplicado nas imagens obtidas em laboratório, com a diferença de que, nas imagens de terreno, se presume que as partículas metálicas apresentam uma cor semelhante ao branco, não havendo ainda confirmação técnica. No entanto, nesta situação, não foi utilizada a Transformada de Hough para calcular o fator de conversão, uma vez que estas imagens não contêm uma barra de escala visível como acontece nas imagens laboratoriais. Como alternativa, recorreu-se à informação sobre a ampliação utilizada durante a captura, neste caso uma ampliação de 120×, assim todos os valores de diâmetro obtidos (em píxeis) foram divididos por 120, de forma a serem expressos em micrómetros.

5

Conclusão e Trabalho Futuro

5.1 Conclusão

A presente dissertação teve como objetivo o desenvolvimento de um algoritmo automatizado para a deteção e medição de partículas metálicas em lubrificantes, recorrendo a técnicas de processamento digital de imagem com a biblioteca OpenCV.

Nos ensaios realizados em imagens laboratoriais, captadas sob condições controladas de iluminação e foco, o processamento de imagens revelou-se eficaz. O algoritmo conseguiu identificar e quantificar as partículas com precisão e consistência, demonstrando a validade da metodologia em ambiente controlado. Importa salientar que a utilização de valores elevados nos parâmetros de suavização e contraste contribuiu decisivamente para estes resultados, permitindo compensar situações em que algumas imagens apresentavam ruído acentuado ou se encontravam parcialmente desfocadas.

Contudo, quando aplicado a imagens recolhidas no terreno, o desempenho foi significativamente afetado, sendo necessária a criação de um protocolo que assegure que as imagens são adquiridas de forma análoga. Diferenças substanciais nas condições de aquisição, nomeadamente variações de foco, iluminação heterogénea e qualidade das amostras, comprometeram a eficácia do método, impossibilitando a replicação dos bons resultados obtidos em laboratório.

Perante estas limitações foi testada uma nova abordagem baseada na segmentação de cores através do espaço HSV, definindo limiares específicos para cada um dos canais. No entanto, o conjunto de dados disponível para esta fase era significativamente reduzido, o que limitou as abordagens passíveis de serem testadas e inviabilizou a validação do método. A inexistência de relatórios técnicos complementares também dificultou a avaliação rigorosa dos resultados, tanto nas imagens de campo como nas imagens de laboratório.

Apesar destas restrições, o trabalho realizado constitui um contributo relevante, ao demonstrar o potencial da visão computacional na monitorização de lubrificantes industriais e ao identificar os principais obstáculos à sua aplicação prática.

5.2 Trabalho Futuro

Os resultados obtidos no presente trabalho permitem identificar várias oportunidades de evolução e aprofundamento da investigação. Embora o algoritmo determinístico desenvolvido tenha demonstrado potencial em ambiente laboratorial, os desafios encontrados na aplicação em condições reais evidenciam a necessidade de melhorias metodológicas e de alargamento da base de dados disponível.

Um primeiro passo será a criação de um conjunto de dados mais extenso e diversificado, abrangendo diferentes condições de iluminação, níveis de foco e variabilidade das amostras. A recolha sistemática de imagens em contexto industrial, aliada a observações a microscópio de modo a obter um relatório técnico detalhado referente a cada uma, e condições de operação, permitirá enriquecer o processo de validação do algoritmo e aumentar a sua eficácia.

Outra linha de evolução prende-se com a incorporação de técnicas de aprendizagem automática e, em particular, de redes neuronais convolucionais (RNC). Estas abordagens têm-se revelado eficazes em problemas de segmentação e classificação de imagens em contextos complexos e poderiam superar as limitações do processamento tradicional observado nas imagens de campo.

Do ponto de vista metodológico, importa igualmente explorar estratégias híbridas que combinem técnicas clássicas de processamento digital de imagem, como aquelas testadas neste projeto, com métodos baseados em aprendizagem automática.

Por fim, recomenda-se a realização de testes em cenários industriais reais, com recolha de dados em tempo contínuo. Essa abordagem permitiria avaliar não apenas a precisão do algoritmo, mas também o seu desempenho em termos de tempo de processamento.

Bibliografia

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (ver p. i).

[2] R. Mistry e R. Maynus. «Lubrication - Crucial for rotating machines». Em: *2012 Petroleum and Chemical Industry Conference (PCIC)*. 2012, pp. 1–9. DOI: 10.1109/PCICON.2012.6549673 (ver p. 1).

[3] S. Rizvi. *A Comprehensive Review of Lubricant Technology Selection, and Design*. ASTM International, 2009. ISBN: 9780803170001. URL: https://books.google.pt/books?id=azwM0AEACAAJ (ver p. 1).

[4] K. C. Ludema. *Friction, Wear, Lubrication: A Textbook in Tribology*. 1st. CRC Press, 1996. ISBN: 9780429190490 (ver p. 1).

[5] A. Verma, S. Gahtori e J. Sharma. «Analysis of the Various Lubricants Used in the Different Machining Processes to Improve the Quality of Life of Products and Personnel's Hacks». Em: *JRTDD* 06.2s (2023-03), pp. 140–144. URL: https://jrtdd.com/index.php/journal/article/view/272 (ver p. 2).

[6] D. Puhan. «Lubricant and Lubricant Additives». Em: *Tribology in Materials and Manufacturing*. Ed. por A. Patnaik, T. Singh e V. Kukshal. Rijeka: IntechOpen, 2020. Cap. 9. DOI: 10.5772/intechopen.93830. URL: https://doi.org/10.5772/intechopen.93830 (ver p. 2).

[7] S. Mia e N. Ohno. «Relation between low temperature fluidity and sound velocity of lubricating oil». Em: *Tribology International* 43.5 (2010). Special Issue on Second International Conference on Advanced Tribology (iCAT2008), pp. 1043–1047. ISSN: 0301-679X. DOI: https://doi.org/10.1016/j.triboint.2009.12.027. URL: https://www.sciencedirect.com/science/article/pii/S0301679X09003648 (ver p. 2).

[8] P. C. Barbosa. *Porque Utilizamos Lubrificantes na Indústria?* Accessed: 2024-12-25. 2024. URL: https://paulocbarbosa.pt/porque-utilizamos-lubrificantes-na-industria/ (ver p. 2).

[9] H. Liu et al. «The mechanisms and applications of friction energy dissipation». Em: *Friction* 11.6 (2022-08), pp. 839–864. ISSN: 2223-7704. DOI: 10.1007/s40544-022-0639-0. URL: http://dx.doi.org/10.1007/s40544-022-0639-0 (ver p. 2).

[10] G. K. Nikas. «Review of Studies on the Detrimental Effects of Solid Contaminants in Lubricated Machine Element Contacts». Em: *Reliability Engineering Advances*. Ed. por G. I. Hayworth. New York: Nova Science Publishers, Inc, 2009-01, pp. 1–36. ISBN: 978-1-60692-329-0 (ver p. 2).

[11] Y. Sinha et al. «Significance of Effective Lubrication in Mitigating System Failures — A Wind Turbine Gearbox Case Study». Em: *Wind Engineering* 38.4 (2014), pp. 441–449. DOI: 10.1260/0309-524X.38.4.441 (ver p. 2).

[12] W. Needelman e P. Madhavan. «Review of Lubricant Contamination and Diesel Engine Wear». Em: *SAE Technical Paper* 881827 (1988). DOI: 10.4271/881827 (ver p. 2).

[13] M. K. A. Ali et al. *Effect of Lubricant Contaminants on Tribological Characteristics During Boundary Lubrication Reciprocating Sliding*. 2017. DOI: https://doi.org/10.48550/arXiv.1710.04448 (ver p. 2).

[14] A. Bouchireb e M. R. S. and. *Effect of solid particles on gear tooth failure*. 2015-05. DOI: https://doi.org/10.1007/s11771-015-2685-5 (ver p. 2).

[15] T. Khan, M. Broderick e C. Taylor. «Investigating the industrial impact of hydraulic oil contamination on tool wear during machining and the development of a novel quantification methodology». Em: *International Journal of Advanced Manufacturing Technology* 112 (2021), pp. 589–600. DOI: 10.1007/s00170-020-06370-y (ver p. 3).

[16] S. Benes. *Debris Analysis: Alternatives to Traditional Oil Testing*. https://assets.thermofisher.com/TFS-Assets/MSD/Reference-Materials/Debris-Alternatives-Traditional-Oil-Testing-SEM-White-Paper.pdf. Accessed: 2024-12-26 (ver p. 3).

[17] D. Scott e V. Westcott. «Predictive maintenance by ferrography». Em: *Wear* 44.1 (1977), pp. 173–182. ISSN: 0043-1648. DOI: https://doi.org/10.1016/0043-1648(77)90094-1. URL: https://www.sciencedirect.com/science/article/pii/0043164877900941 (ver p. 3).

[18] E. Lyons. «Digital Image Processing: an Overview». Em: *Computer* 10.8 (1977), pp. 12–14. DOI: 10.1109/C-M.1977.217813 (ver p. 5).

[19] M. Cannon. «How Digital Image Processing Became Really Easy». Em: *High Speed Photography, Videography, and Photonics V*. Ed. por H. C. Johnson. Vol. 0832. International Society for Optics e Photonics. SPIE, 1988, pp. 2–7. DOI: 10.1117/12.942200. URL: https://doi.org/10.1117/12.942200 (ver p. 5).

[20] S. More e D. Mishra. «Developments in Image Processing using Deep learning and Reinforcement learning». Em: *2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG)*. 2023, pp. 1–9. DOI: 10.1109/ICTBIG59752.2023.10456257 (ver p. 5).

[21] S. Mandour. «An Exhaustive Review of Neutrosophic Logic in Addressing Image Processing Issues». Em: *Neutrosophic Systems with Applications* 12 (2023-12), pp. 36–55. DOI: 10.61356/j.nswa.2023.110 (ver p. 5).

[22] M. G. Ionita e H. G. Coanda. «An Improved Automatic Periodic Noise Removal Algorithm for Microscopic Images». Em: *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. 2022, pp. 1–4. DOI: 10.1109/ECAI54874.2022.9847494 (ver p. 5).

[23] T. Julliand, V. Nozick e H. Talbot. «Image Noise and Digital Image Forensics». Em: *Digital-Forensics and Watermarking*. Ed. por Y.-Q. Shi et al. Cham: Springer International Publishing, 2016, pp. 3–17. ISBN: 978-3-319-31960-5 (ver p. 5).

[24] S. Chokkalingam, K. Karuppanan e M. Sowmya. «Performance analysis of various lymphocytes images de-noising filters over a microscopic blood smear image». Em: *International Journal of Pharma and Bio Sciences* 4 (2013-10), B1250–B1258 (ver p. 5).

[25] M. Alavi e M. Kargari. «A new contrast enhancement method for Color dark and low-light images». Em: *2022 9th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. 2022, pp. 1–7. DOI: 10.1109/CFIS54774.2022.9756426 (ver pp. 5, 33).

[26] M. Kaur, J. Kaur e J. Kaur. «Survey of Contrast Enhancement Techniques based on Histogram Equalization». Em: *International Journal of Advanced Computer Science and Applications* 2.7 (2011). DOI: 10.14569/IJACSA.2011.020721. URL: http://dx.doi.org/10.14569/IJACSA.2011.020721 (ver p. 5).

[27] Y. Chang et al. «Automatic Contrast-Limited Adaptive Histogram Equalization With Dual Gamma Correction». Em: *IEEE Access* 6 (2018), pp. 11782–11792. DOI: 10.1109/ACCESS.2018.2797872 (ver p. 6).

[28] D. Ferraretti et al. «Spot Detection in Images with Noisy Background». Em: *Image Analysis and Processing – ICIAP 2011*. Vol. 6978. Lecture Notes in Computer Science. Springer, 2011, pp. 575–584. DOI: 10.1007/978-3-642-24085-0_59. URL: https://doi.org/10.1007/978-3-642-24085-0_59 (ver p. 6).

[29] S. Naik, S. B. K e V. KC. «Image Segmentation Techniques: A Comprehensive Review». Em: *International Research Journal of Modernization in Engineering, Technology and Science (IRJMETS)* 06.02 (2024-02). Impact Factor: 7.868, Peer-Reviewed, Open Access, Fully Refereed International Journal, pp. 1757–1765. ISSN: 2582-5208. DOI: 10.56726/IRJMETS49604. URL: https://www.irjmets.com (ver p. 6).

[30] F. Bertholdo, E. Valle e A. Araújo. «Layout-aware limiarization for readability enhancement of degraded historical documents». Em: 2009-09, pp. 131–134. DOI: 10.1145/1600193.1600223 (ver p. 6).

[31] N. S. Baghel, D. K. Sahu e V. Namdeo. «Optimization of Methods for Image Segmentation by using Thresholding Techniques». Em: *International Journal for Research in Applied Science and Engineering Technology (IJRASET)* 8.II (2020-02), pp. 222–222. ISSN: 2321-9653. DOI: 10.22214/ijraset.2020.2032. URL: https://doi.org/10.22214/ijraset.2020.2032 (ver p. 6).

[32] C. Shanthi, R. K. Porpatham e N. Pappa. «Image Analysis for Particle Size Distribution». Em: *International Journal of Engineering and Technology (IJET)* 6.3 (2014-06), pp. 1340–1345. ISSN: 0975-4024. URL: https://www.researchgate.net/publication/287319918_Image_Analysis_for_Particle_Size_Distributionn (ver p. 6).

[33] A. M. Raid et al. «Image Restoration Based on Morphological Operations». Em: *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)* 4.3 (2014-06), pp. 9–18. DOI: 10.5121/ijcseit.2014.4302 (ver p. 6).

[34] E. R. Dougherty e R. de Alencar Lotufo. «Binary Erosion and Dilation». Em: SPIE, 2003, pp. 1–25. DOI: 10.1117/3.501104.ch1 (ver p. 6).

[35] P. Ganesan e G. Sajiv. «A comprehensive study of edge detection for image processing applications». Em: *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. IEEE, 2017-03, pp. 1–6. DOI: 10.1109/iciiecs.2017.8275968. URL: http://dx.doi.org/10.1109/ICIIECS.2017.8275968 (ver p. 6).

[36] S. Dražić, N. Sladoje e J. Lindblad. «Estimation of Feret's diameter from pixel coverage representation of a shape». Em: *Pattern Recognition Letters* 80 (2016-09), pp. 37–45. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2016.04.021. URL: http://dx.doi.org/10.1016/j.patrec.2016.04.021 (ver p. 6).

[37] P. KARAKUŞ. «MACHINE WHELL EDGE DETECTION MORPHOLOGICAL OPERATIONS». Em: *Konya Journal of Engineering Sciences* (2024-02), pp. 251–262. ISSN: 2147-9364. DOI: 10.36306/konjes.1418523. URL: http://dx.doi.org/10.36306/konjes.1418523 (ver pp. 6, 33).

[38] M. Mabaso, D. Withey e B. Twala. «Spot Detection in Microscopy Images using Convolutional Neural Network with Sliding-Window Approach». Em: *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018) - Volume 2: BIOIMAGING*. SCITEPRESS - Science e Technology Publications, 2018, pp. 67–74. DOI: 10.5220/0006724200670074 (ver pp. 6, 7).

[39] S. Sakib et al. «An Overview of Convolutional Neural Network: Its Architecture and Applications». Em: (2018-11). DOI: 10.20944/preprints201811.0546.v1. URL: http://dx.doi.org/10.20944/PREPRINTS201811.0546.V1 (ver p. 7).

[40] J. Brunekreef. «Sliding Window Protocols». Em: *Algebraic Specification of Communication Protocols*. Cambridge University Press, 1993-09, pp. 71–112. DOI: 10.1017/cbo9780511721625.005. URL: http://dx.doi.org/10.1017/CBO9780511721625.005 (ver p. 7).

[41] Y. M. Kadah. «Motion artifact suppression in MRI using k-Space overlap processing». Em: *2008 National Radio Science Conference*. IEEE, 2008-03, pp. 1–9. DOI: 10.1109/nrsc.2008.4542321. URL: http://dx.doi.org/10.1109/NRSC.2008.4542321 (ver p. 7).

[42] V. Tongur, A. B. Batıbay e M. Karakoyun. «A Review on Measurement of Particle Sizes by Image Processing Techniques». Em: *Journal of Soft Computing and Artificial Intelligence* 04.01 (2023-06), pp. 15–28. DOI: 10.55195/jscai.1218662. URL: https://www.researchgate.net/publication/368616214_A_Review_on_Measurement_of_Particle_Sizes_by_Image_Processing_Techniques (ver p. 7).

[43] H. Chun et al. «Wafer particle inspection technique using computer vision based on a color space transform model». Em: *The International Journal of Advanced Manufacturing Technology* 127 (2023-07), pp. 1–9. DOI: 10.1007/s00170-023-11888-y (ver p. 7).

[44] K. Rameshkumar et al. «Machine Learning Approach for Predicting the Solid Particle Lubricant Contamination in a Spherical Roller Bearing». Em: *IEEE Access* 12 (2024-04), pp. 78680–78700. DOI: 10.1109/ACCESS.2024.3408807. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10546961 (ver p. 7).

[45] D. M. Scott. *Industrial Process Sensors*. CRC Press, 2018-10. ISBN: 9781315219950. DOI: 10.1201/9781315219950. URL: http://dx.doi.org/10.1201/9781315219950 (ver p. 8).

[46] S.-J. Royer et al. «Computer vision segmentation model—deep learning for categorizing microplastic debris». Em: *Frontiers in Environmental Science* 12 (2024-07). DOI: https://doi.org/10.3389/fenvs.2024.1386292 (ver p. 8).

[47] P. Corke. «Images and Image Processing». Em: *Robotics, Vision and Control*. Springer International Publishing, 2017, pp. 359–411. ISBN: 9783319544137. DOI: 10.1007/978-3-319-54413-7_12. URL: http://dx.doi.org/10.1007/978-3-319-54413-7_12 (ver p. 9).

[48] R. C. Gonzalez e R. E. Woods. *Digital Image Processing*. 4th. Upper Saddle River, NJ, USA: Prentice Hall, 2018. ISBN: 978-0133356724 (ver pp. 9, 11, 27, 29).

[49] C. Solomon e T. Breckon. *Fundamentals of Digital Image Processing*. 1st. John Wiley & Sons, Ltd, 2011. ISBN: 978 0 470 84472 4 (ver pp. 9, 13).

[50] B. HUNT. «PROSPECTS FOR IMAGE RESTORATION». Em: *International Journal of Modern Physics C* 05.01 (1994-02), pp. 151–178. ISSN: 1793-6586. DOI: 10.1142/s0129183194000118. URL: http://dx.doi.org/10.1142/S0129183194000118 (ver p. 9).

[51] S. Supiyandi et al. «Pengenalan Gambar Dasar Menggunakan Python dan OpenCV». Em: *Jurnal Sistem Informasi dan Ilmu Komputer* 2.4 (2024-11), pp. 52–61. ISSN: 2986-5158. DOI: 10.59581/jusiik-widyakarya.v2i4.4200. URL: http://dx.doi.org/10.59581/jusiik-widyakarya.v2i4.4200 (ver p. 9).

[52] E. Vocaturo, E. Zumpano e P. Veltri. «Image pre-processing in computer vision systems for melanoma detection». Em: *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2018-12, pp. 2117–2124. DOI: 10.1109/bibm.2018.8621507. URL: http://dx.doi.org/10.1109/BIBM.2018.8621507 (ver p. 10).

[53] A. Bali e S. N. Singh. «A Review on the Strategies and Techniques of Image Segmentation». Em: *2015 Fifth International Conference on Advanced Computing & amp; Communication Technologies*. IEEE, 2015-02, pp. 113–120. DOI: 10.1109/acct.2015.63. URL: http://dx.doi.org/10.1109/ACCT.2015.63 (ver p. 10).

[54] G. Kumar e P. K. Bhatia. «A Detailed Review of Feature Extraction in Image Processing Systems». Em: *2014 Fourth International Conference on Advanced Computing & Communication Technologies*. IEEE, 2014-02, pp. 5–12. DOI: 10.1109/acct.2014.74. URL: http://dx.doi.org/10.1109/ACCT.2014.74 (ver p. 10).

[55] Z. Wu e J. Robinson. «Edge-preserving colour-to-greyscale conversion». Em: *IET Image Processing* 8.4 (2014-04), pp. 252–260. ISSN: 1751-9667. DOI: 10.1049/iet-ipr.2013.0348. URL: http://dx.doi.org/10.1049/iet-ipr.2013.0348 (ver p. 13).

[56] D. A. Kerr. *The CIE XYZ and xyY Color Spaces*. https://graphics.stanford.edu/courses/cs148-10-summer/docs/2010--kerr--cie_xyz.pdf. Issue 1. 2010 (ver p. 14).

[57] OpenCV Developers. *Color Conversions — OpenCV Documentation*. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html. Accessed: July 29, 2025. 2018 (ver pp. 14, 15).

[58] S. Saifullah et al. «K-Means Segmentation Based-on Lab Color Space for Embryo Detection in Incubated Egg». Em: *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika* 8.2 (2022-07), p. 175. ISSN: 2338-3070. DOI: 10.26555/jiteki.v8i2.23724. URL: http://dx.doi.org/10.26555/jiteki.v8i2.23724 (ver p. 15).

[59] N. Khamdi, M. Susantok e P. Leopard. «Pendeteksian Objek Bola dengan Metode Color Filtering HSV pada Robot Soccer Humanoid». Em: *JURNAL NASIONAL TEKNIK ELEKTRO* 6.2 (2017-07), p. 123. ISSN: 2302-2949. DOI: 10.25077/jnte.v6n2.398.2017. URL: http://dx.doi.org/10.25077/jnte.v6n2.398.2017 (ver p. 16).

[60] C. Junhua e L. Jing. «Research on Color Image Classification Based on HSV Color Space». Em: *2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control*. IEEE, 2012-12, pp. 944–947. DOI: 10.1109/imccc.2012.226. URL: http://dx.doi.org/10.1109/IMCCC.2012.226 (ver p. 16).

[61] R. Padmavathy e R. Priya. «Enhancement of Medical Images Using Histogram Equalization Techniques». Em: *International Journal of Engineering & Technology* 7.2.8 (2018), pp. 237–240. URL: https://www.sciencepubco.com/index.php/ijet/article/view/14811 (ver p. 17).

[62] O. Patel, Y. P. S. Maravi e S. Sharma. «A Comparative Study of Histogram Equalization Based Image Enhancement Techniques for Brightness Preservation and Contrast Enhancement». Em: *Signal & Image Processing : An International Journal* 4.5 (2013-11), pp. 11–25. ISSN: 0976-710X. DOI: 10.5121/sipij.2013.4502. URL: http://dx.doi.org/10.5121/sipij.2013.4502 (ver p. 17).

[63] A. Boschetti et al. «High Dynamic Range image tone mapping based on local Histogram Equalization». Em: 2010-07, pp. 1130–1135. DOI: 10.1109/ICME.2010.5583305 (ver pp. 18–20).

[64] OpenCV Team. *Histogram Equalization — OpenCV Documentation*. https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html. Accessed: 15-Jun-2025. 2025 (ver p. 19).

[65] T. Kryjak et al. «Real-Time CLAHE Algorithm Implementation in SoC FPGA Device for 4K UHD Video Stream». Em: *Electronics* 11.14 (2022-07), p. 2248. ISSN: 2079-9292. DOI: 10.3390/electronics11142248. URL: http://dx.doi.org/10.3390/electronics11142248 (ver p. 20).

[66] C. A. Glasbey e G. B. Horgan. *Image analysis for the biological sciences*. 1st. John Wiley & Sons, Ltd, 1995. ISBN: 0471937266 (ver pp. 25, 27).

[67] OpenCV Team. *OpenCV Documentation*. https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html. Accessed: July 29, 2025. 2025 (ver p. 29).

[68] L. Wang, Y. Zhang e J. Feng. «On the Euclidean distance of images». Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (2005-08), pp. 1334–1339. ISSN: 0162-8828. DOI: 10.1109/tpami.2005.165. URL: http://dx.doi.org/10.1109/TPAMI.2005.165 (ver p. 29).

[69] C. Tomasi e R. Manduchi. «Bilateral filtering for gray and color images». Em: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. ICCV-98. Narosa Publishing House, pp. 839–846. DOI: 10.1109/iccv.1998.710815. URL: http://dx.doi.org/10.1109/ICCV.1998.710815 (ver p. 32).

[70] U. Sara, M. Akter e M. S. Uddin. «Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study». Em: *Journal of Computer and Communications* 07.03 (2019), pp. 8–18. ISSN: 2327-5227. DOI: 10.4236/jcc.2019.73002. URL: http://dx.doi.org/10.4236/jcc.2019.73002 (ver pp. 33, 34).

[71] S. Maruyama. «Properties of the SSIM metric in medical image assessment: Correspondence between measurements and the spatial frequency spectrum». Em: (2022-08). DOI: 10.21203/rs.3.rs-1872101/v1. URL: http://dx.doi.org/10.21203/rs.3.rs-1872101/v1 (ver p. 34).

[72] R. Sun et al. «Survey of Image Edge Detection». Em: *Frontiers in Signal Processing* 2 (2022-03). DOI: 10.3389/frsip.2022.826967 (ver p. 34).

[73] H. Agrawal e K. Desai. «CANNY EDGE DETECTION: A COMPREHENSIVE REVIEW». Em: *International Journal of Technical Research & amp; Science* 9.Spl (2024-06), pp. 27–35. ISSN: 2454-2024. DOI: 10.30780/specialissue-iset-2024/023. URL: http://dx.doi.org/10.30780/specialissue-ISET-2024/023 (ver p. 34).

[74] F. Hossain, M. Kumar e M. Abu. «Hardware Design and Implementation of Adaptive Canny Edge Detection Algorithm». Em: *International Journal of Computer Applications* 124.9 (2015-08), pp. 31–38. ISSN: 0975-8887. DOI: 10.5120/ijca2015905446. URL: http://dx.doi.org/10.5120/ijca2015905446 (ver pp. 36, 37).

[75] N. Otsu. «A Threshold Selection Method from Gray-Level Histograms». Em: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979-01), pp. 62–66. ISSN: 2168-2909. DOI: 10.1109/tsmc.1979.4310076. URL: http://dx.doi.org/10.1109/TSMC.1979.4310076 (ver p. 38).

[76] M. Fang, G. Yue e Q. Yu. «The Study on An Application of Otsu Method in Canny Operator». Em: *Proceedings of the 2009 International Symposium on Information Processing (ISIP'09)*. Huangshan, P. R. China: Academy Publisher, 2009, pp. 109–112 (ver p. 38).

[77] X. M. Zhao, W. X. Wang e L. P. Wang. «Parameter optimal determination for canny edge detection». Em: *The Imaging Science Journal* 59.6 (2011-11), pp. 332–341. ISSN: 1743-131X. DOI: 10.1179/136821910x12867873897517. URL: http://dx.doi.org/10.1179/136821910X12867873897517 (ver p. 39).

[78] J. S. Low et al. «Seeding on Samples for Accelerating K-Means Clustering». Em: *Proceedings of the 3rd International Conference on Big Data and Internet of Things*. BDIOT 2019. ACM, 2019-08. DOI: 10.1145/3361758.3361774. URL: http://dx.doi.org/10.1145/3361758.3361774 (ver p. 42).

[79] S. Lou, X. Jiang e P. J. Scott. «Applications of Morphological Operations in Surface Metrology and Dimensional Metrology». Em: *Journal of Physics: Conference Series* 483 (2014-03), p. 012020. ISSN: 1742-6596. DOI: 10.1088/1742-6596/483/1/012020. URL: http://dx.doi.org/10.1088/1742-6596/483/1/012020 (ver p. 43).

[80] K. A. M. Said e A. B. Jambek. «Analysis of Image Processing Using Morphological Erosion and Dilation». Em: *Journal of Physics: Conference Series* 2071.1 (2021-10), p. 012033. ISSN: 1742-6596. DOI: 10.1088/1742-6596/2071/1/012033. URL: http://dx.doi.org/10.1088/1742-6596/2071/1/012033 (ver p. 43).

[81] H. Xu, X. Xu e Y. Zuo. «Applying morphology to improve Canny operators image segmentation method». Em: *The Journal of Engineering* (2019), pp. 8816–8819. DOI: https://doi.org/10.1049/joe.2018.9113. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/joe.2018.9113 (ver pp. 44, 46).

[82] S. Suzuki e K. be. «Topological structural analysis of digitized binary images by border following». Em: *Computer Vision, Graphics, and Image Processing* 30.1 (1985-04), pp. 32–46. ISSN: 0734-189X. DOI: 10.1016/0734-189x(85)90016-7. URL: http://dx.doi.org/10.1016/0734-189x(85)90016-7 (ver pp. 47, 48).

[83] W. H. Walton. «Feret's Statistical Diameter as a Measure of Particle Size». Em: *Nature* 162.4125 (1948), pp. 329–330. DOI: 10.1038/162329b0 (ver p. 50).

[84] M. FOKKINGA. «The Hough transform». Em: *Journal of Functional Programming* 21.2 (2011-02), pp. 129–133. ISSN: 1469-7653. DOI: 10.1017/s0956796810000341. URL: http://dx.doi.org/10.1017/S0956796810000341 (ver p. 51).

[85] M. Sonka, V. Hlavac e R. Boyle. *Image Processing, Analysis, and Machine Vision*. 4ª ed. Cengage Learning, 2013. ISBN: 978-1-133-59360-7 (ver p. 51).

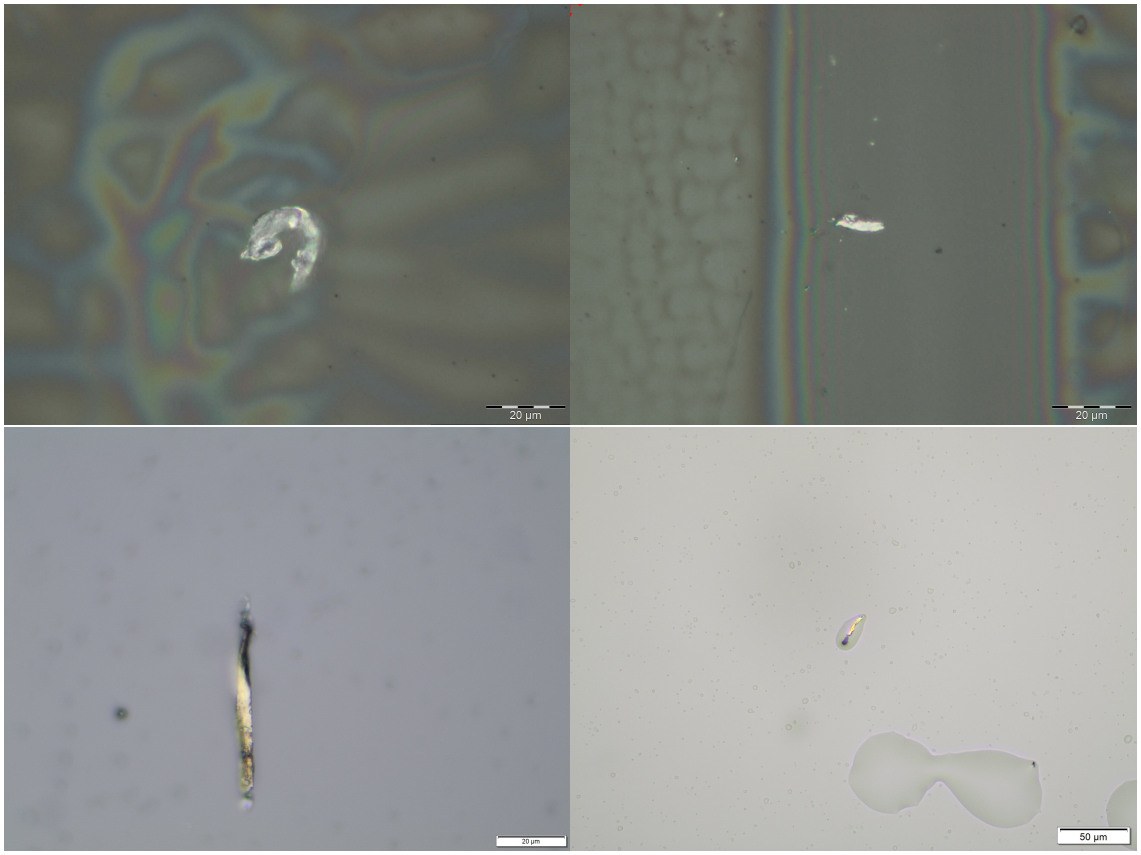
[86] J. Ferner et al. *Persistence-based Hough Transform for Line Detection*. 2025-04. DOI: 10.48550/ARXIV.2504.16114. URL: https://arxiv.org/abs/2504.16114 (ver pp. 51, 52).

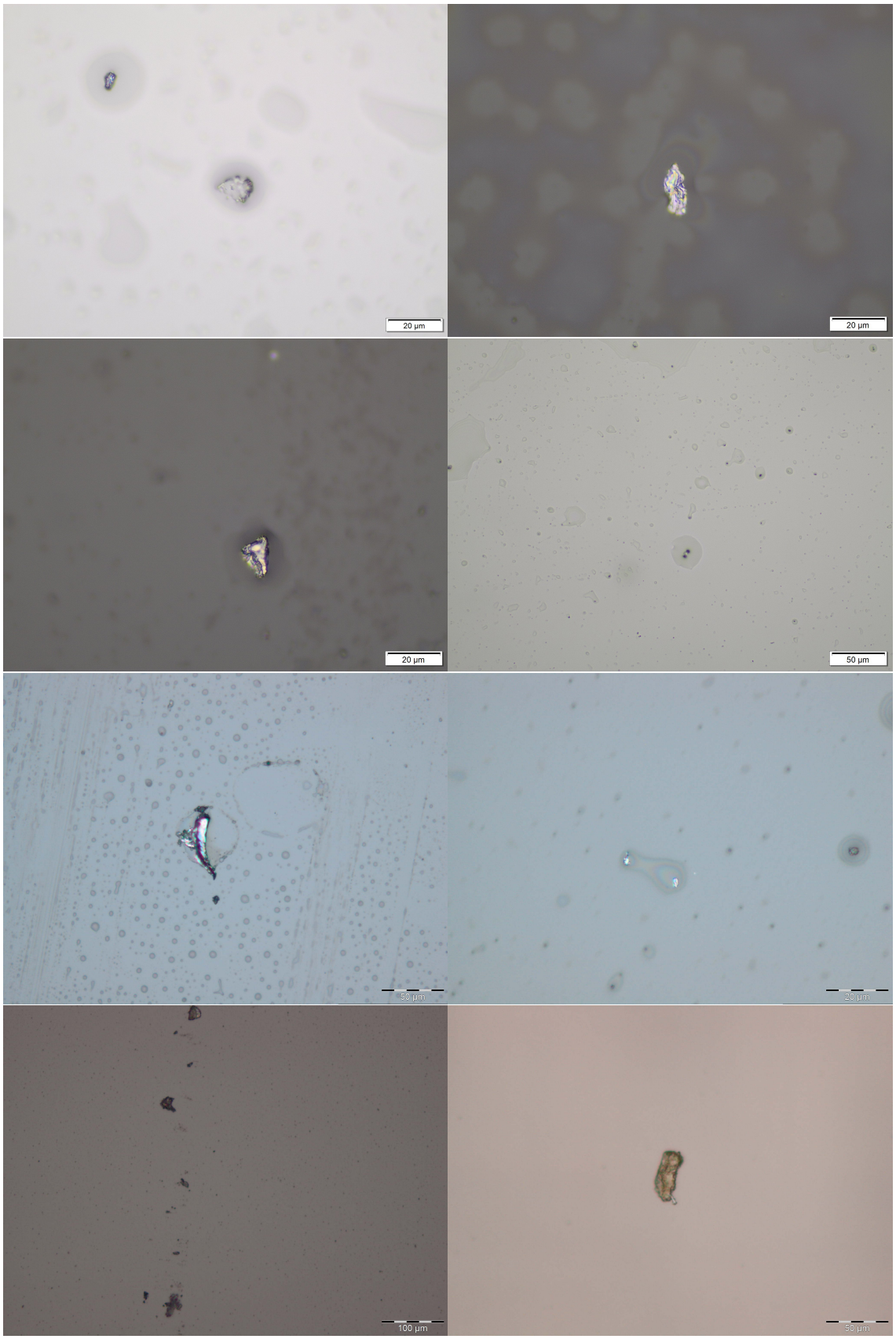
I

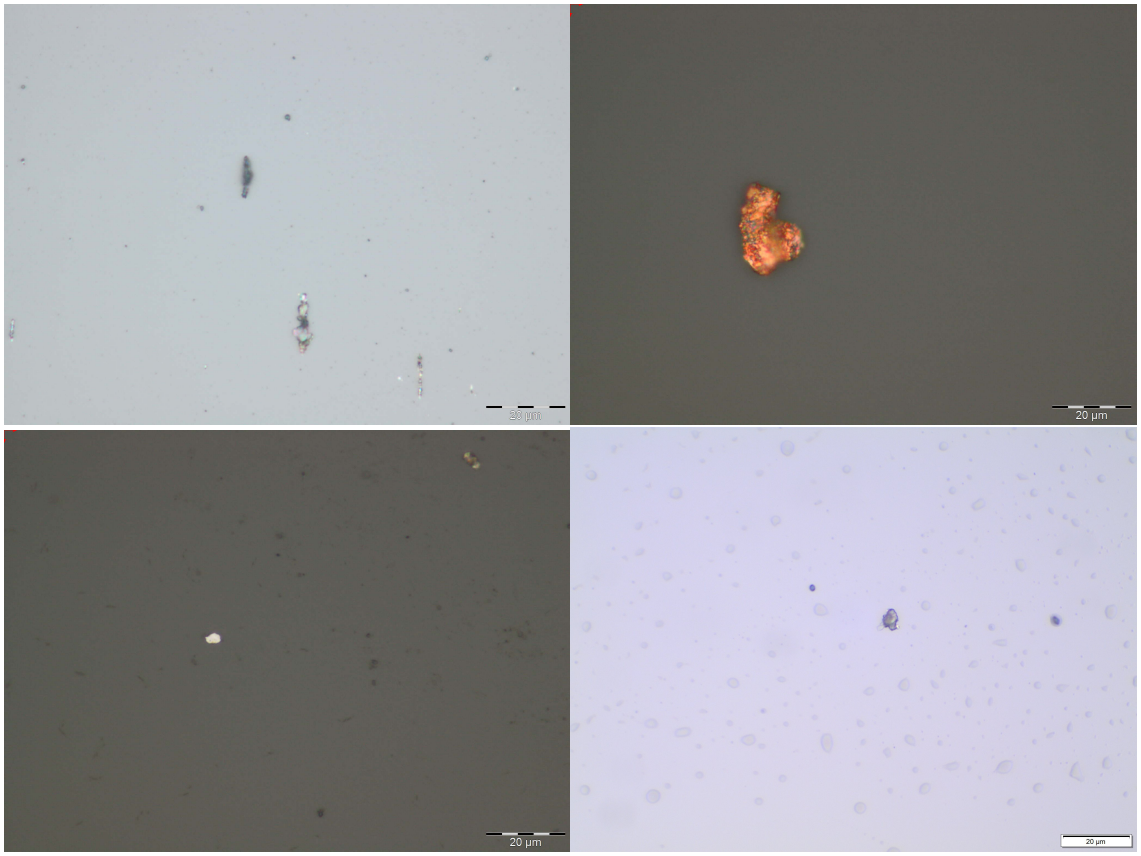
Anexo 1

Este anexo contém todas as imagens, de laboratório e terreno, que compõem conjunto de dados fornecido pela GALP.

I.1 Conjunto de dados das Imagens de Laboratório




20 µm
20 µm
20 µm
50 µm
100 µm
50 µm


20 µm
20 µm
20 µm
20 µm

I.2 Conjunto de dados das Imagens de Terreno