UNIVERSIDADE FEDERAL DE SÃO CARLOS - UFSCar
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA - CCET
DEPARTAMENTO DE ENGENHARIA MECÂNICA - DEMec

João Pedro Gonçalves Ernandes

CLASSIFICAÇÃO DE DEFEITOS A PARTIR DE IMAGENS DE SUPERFÍCIES FABRICADAS POR MANUFATURA ADITIVA ROBOTIZADA

São Carlos - SP
2025

João Pedro Gonçalves Ernandes

CLASSIFICAÇÃO DE DEFEITOS A PARTIR DE IMAGENS DE SUPERFÍCIES FABRICADAS POR MANUFATURA ADITIVA ROBOTIZADA

Trabalho de conclusão de curso apresentado ao curso de graduação em Engenharia Mecânica da Universidade Federal de São Carlos, para obtenção do título de bacharel em Engenharia Mecânica.
Orientador: Prof. Dr. Sidney Bruce Shiki

São Carlos - SP

2025

FUNDAÇÃO UNIVERSIDADE FEDERAL DE SÃO CARLOS

COORDENAÇÃO DO CURSO DE ENGENHARIA MECÂNICA - CCEMec/CCET
Rod. Washington Luís km 235 - SP-310, s/n - Bairro Monjolinho, São Carlos/SP, CEP 13565-905
Telefone: (16) 33519703 - http://www.ufscar.br

DP-TCC-FA nº 16/2025/CCEMec/CCET

Graduação: Defesa Pública de Trabalho de Conclusão de Curso

Folha Aprovação (GDP-TCC-FA)

FOLHA DE APROVAÇÃO

JOAO PEDRO GONCALVES ERNANDES

CLASSIFICAÇÃO DE DEFEITOS A PARTIR DE IMAGENS DE SUPERFÍCIES FABRICADAS POR MANUFATURA ADITIVA ROBOTIZADA

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos – Campus São Carlos

São Carlos, 26 de fevereiro de 2025

ASSINATURAS E CIÊNCIAS

Cargo/Função	Nome Completo
Orientador	Sidney Bruce Shiki
Membro da Banca 1	João Gustavo Pereira da Silva
Membro da Banca 2	Gustavo Franco Barbosa

Documento assinado eletronicamente por **Sidney Bruce Shiki**, **Docente**, em 26/02/2025, às 09:12, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Joao Gustavo Pereira da Silva**, **Docente**, em 26/02/2025, às 09:15, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Gustavo Franco Barbosa**, **Professor(a) do Ensino Superior**, em 26/02/2025, às 10:51, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufscar.br/autenticacao, informando o código verificador **1757077** e o código CRC **350F0A5B**.

Referência: Caso responda a este documento, indicar expressamente o Processo nº 23112.005200/2025-53

SEI nº 1757077

Modelo de Documento: Grad: Defesa TCC: Folha Aprovação, versão de 02/Agosto/2019

Dedico a minha família por sempre me apoiar e aos meus amigos por sempre estarem ao meu lado.

Agradecimentos

Aos meus pais, Rodrigo e Renata, por me apoiarem durante toda minha vida e me darem condições de chegar até aqui, sem eles nada disso seria possível. Também a minha irmã, Júlia, pelo apoio.

Aos meus avós, tias e primos pelo apoio durante essa jornada e especialmente ao meu avô Joaquim, que acredito que estaria muito feliz.

Aos meus amigos Discípulos do Pepe, por estarem sempre ao meu lado em momentos bons e ruins, nem sempre sendo uma boa influência para os estudos e pontualidade, mas no fim, bons exemplos, cada um de seu jeito. Especialmente ao Thiago, por ter aturado dividir apartamento comigo por 5 anos.

Aos meus companheiros na Engrenar Jr., especialmente o time de marketing, por apoiar minhas decisões e me dar a oportunidade de ter grandes aprendizados.

Ao professor Bruce, pelos ensinamentos, pela enorme paciência e pela disposição, mesmo eu estando longe de ser o melhor orientado. Obrigado pela oportunidade.

À Universidade Federal de São Carlos, por proporcionar um bom período de graduação, me permitindo experienciar diversos momentos, conhecer inúmeras pessoas e aprender sobre engenharia e sobre a vida.

Resumo

Com a rápida evolução das indústrias nos últimos anos, novos métodos de produção mais flexíveis e menos custosos são requisitados para atender as demandas. Entre os métodos que mais crescem, se destaca a manufatura aditiva, também conhecida como impressão 3D. Tal processo se mostra vantajoso para a indústria por possuir um design mais flexível, produzindo peças com geometrias complexas a custos baixos e uma boa coerência com o modelo base. Ainda dentro da manufatura aditiva, existe o processo no qual uma extrusora é acoplada a um manipulador robótico, o que permite ainda mais flexibilidade durante a produção. Entretanto, essa flexibilidade, além de trazer vantagens, também aumenta o número de variáveis no processo e, portanto, aumenta a possibilidade de erros. Nesse contexto, o presente trabalho busca desenvolver um algoritmo capaz de classificar os erros que ocorrem nas superfícies de uma peça enquanto ela é produzida no processo de manufatura robotizada, como forma de viabilizar seu monitoramento. Essa classificação foi realizada por um algoritmo de aprendizado de máquina através da análise de fotos obtidas por uma câmera ligada a um mini-computador Raspberry. Como resultado, foi obtido um algoritmo capaz de classificar, com boa precisão, os defeitos que ocorrem durante a fabricação de uma peça via manufatura aditiva robotizada. Através dessas análises, pode-se realizar um trabalho preventivo para que tais erros não se repitam ou apareçam com menor frequência ou até mesmo implementar um sistema de correção durante o processo.

Palavras-chave: manufatura aditiva robotizada. análise de superfície. algoritmo de aprendizado de máquina

Abstract

With the rapid evolution of industries in recent years, new, more flexible, and cost-effective production methods are required to meet growing demands. Among the fastest-growing methods, additive manufacturing, also known as 3D printing, stands out. This process offers significant advantages for the industry, such as a more flexible design, the ability to produce parts with complex geometries at low costs, and strong consistency with the original model. Within additive manufacturing, there is a specific process where an extruder is mounted on a robotic manipulator, allowing for even greater flexibility during production. However, this flexibility, while advantageous, also increases the number of process variables and, consequently, the potential for errors. In this context, the present work aims to develop an algorithm capable of classifying surface defects that occur on a part during its production using robotic additive manufacturing, as a means of enabling effective monitoring. This classification was performed by a machine learning algorithm through the analysis of images captured by a camera connected to a Raspberry Pi mini-computer. The goal was to achieve an algorithm capable of accurately classifying defects that arise during the robotic additive manufacturing process. Through these analyses, preventive measures can be taken to reduce the frequency of such errors or even eliminate them, as well as to implement a correction system during the manufacturing process.

Keywords: robotic additive manufacturing. surface analysis. machine learning algorithm.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

CAD	Computer Aided Design
CMAR	Célula de Manufatura Aditiva Robotizada
CNN	Convolutional Neural Network
KNN	K-Nearest Neighbors
LBP	Local Binary Pattern
PID	Proportional Integral Derivative
YOLO	You Only Look Once

Sumário

1. Introdução

Nos últimos anos, o setor industrial tem passado por uma rápida evolução, demandando métodos de produção mais flexíveis e economicamente viáveis para atender às exigências do mercado em constante mudança. Nesse contexto, destaca-se a manufatura aditiva, popularmente conhecida como impressão 3D, como um dos métodos em ascensão que merece atenção especial. Tal processo se trata da fabricação a partir de um modelo feito em um software CAD (do inglês *Computer Aided Design*), que é dividido em camadas pelo software utilizado pela impressora, e então a mesma deposita material sob uma plataforma aquecida, construindo-o camada por camada (JIN; ZHANG; GU, 2019).

Esse método de fabricação se mostra extremamente vantajoso na indústria, pois possui um design mais flexível, produzindo peças com geometrias complexas e custos baixos e uma boa coerência com o modelo base. Além disso, a impressão 3D possui um bom custo por eficiência e não produz quase nenhum desperdício de material se comparado com os métodos de produção convencionais. Esses aspectos juntos tornam essa tecnologia atrativa para diversas aplicações (GAO et al., 2015).

Ainda dentro da manufatura aditiva, se destaca o método no qual é utilizado um braço robótico equipado com uma extrusora, que é alimentada principalmente por *pellets*. Tal processo se mostra muito atrativo por trazer mais flexibilidade na seleção de materiais que podem ser usados em formato mais bruto, além do braço robótico proporcionar precisão e uma grande quantidade de graus de liberdade para movimentação do cabeço de impressão (WANG et al., 2016). Por um lado, a ampliação do número de variáveis na produção oferece a oportunidade de desenvolver modelos mais complexos. No entanto, por outro lado, essa maior complexidade também resulta em um aumento significativo de incertezas e possíveis erros ao longo do processo.

Entre os erros mais comuns observados na impressão 3D, destacam-se os erros de superfície, que podem ser atribuídos a uma variedade de fatores. Dois dos principais fatores são a deposição inadequada de material, resultando em áreas sub-extrudadas ou sobre-extrudadas, que podem afetar negativamente o funcionamento das peças impressas (JIN et al., 2021). É fundamental identificar e compreender essas causas potenciais para implementar medidas corretivas eficazes e obter resultados de impressão de alta qualidade.

Algoritmos treinados estão cada vez mais presentes em diversas tarefas, sendo amplamente utilizados para o monitoramento da impressão 3D (JIN; ZHANG; GU, 2019). A implementação automatizada dessas técnicas justifica-se ao auxiliar na detecção uniforme e contínua de erros, quando realizada de maneira adequada. Essa detecção permite a coleta de dados, cuja análise pode viabilizar a identificação de áreas de melhoria e aprimoramento do processo.

Nesse sentido, o presente trabalho tem como objetivo desenvolver um algoritmo de aprendizado de máquina que possa classificar os erros identificados em imagens capturadas durante o processo de impressão 3D. Para isso, foi utilizado um microcomputador Raspberry Pi 3, equipado com uma câmera acoplada à extrusora, a fim de obter as imagens necessárias para análise. O algoritmo desenvolvido permitiu a identificação e classificação dos erros, contribuindo para uma melhor compreensão e aprimoramento do processo de impressão 3D.

1.1. Objetivos

1.1.1. Objetivos Gerais

Implementar um algoritmo de aprendizado de máquina capaz de classificar os erros nas texturas de superfícies de peças fabricadas via manufatura aditiva robotizada.

1.1.2. Objetivos específicos

- Estudar formas de análise de imagens via algoritmo de aprendizado de máquina para efetuar a classificação de texturas de peças fabricadas por meio de manufatura aditiva;
- Efetuar experimentos para impressão de corpos de prova para o monitoramento de textura superficial das peças;
- Observar o efeito de parâmetros de impressão na textura superficial de peças a partir do algoritmo desenvolvido;
- Otimizar o algoritmo, se necessário, para diminuir possíveis erros de classificação.

2. Revisão bibliográfica

A manufatura aditiva é um método de produção que permite o desenvolvimento de peças complexas com pouco desperdício de material e custos relativamente baixos e portanto, possui um grande potencial em inúmeras áreas como a aeroespacial, a indústria automotiva e a área médica, por exemplo (LIU et al., 2019). Dessa forma, inúmeros pesquisadores voltam seus esforços para desenvolver melhorias para essa tecnologia.

Um dos focos dessas pesquisas é a utilização de algoritmos para o monitoramento da impressão 3D, de forma a localizar erros e criar soluções para posteriormente preveni-los. Em alguns casos, essa detecção é utilizada até mesmo para realizar correções durante o processo.

2.1. Manufatura aditiva

A manufatura aditiva é um método de produção no qual, um desenho em um software CAD é dividido em camadas para sua produção. O contínuo e progressivo crescimento da experiência nessa área devido a diversas pesquisas, mostram o otimismo esperado para o futuro dessa tecnologia. Por conta da possibilidade da produção de estruturas mais leves e mais complexas, a manufatura aditiva mostrou possíveis aplicações nas mais diversas áreas como a indústria aeroespacial, automotiva e até médica. Porém, para que este método de produção se torne o principal utilizado na indústria, alguns passos ainda precisam ser dados. Sua gama de materiais ainda é limitada, sua precisão ainda é baixa para produzir partes que requeiram um acabamento mais fino, além de existirem limitações no tamanho das peças produzidas (WONG; HERNANDEZ, 2012).

Um dos maiores problemas para a implementação da manufatura aditiva na indústria é a dificuldade na fabricação de produtos de grandes dimensões. A partir disso, Shah et al. (2019) estudaram a adaptação de uma impressora 3D em larga escala para utilização industrial. Um protótipo foi construído e concluiu-se que alguns quesitos deveriam ser levados em consideração quanto a essa adaptação, entre eles, os mais importantes seriam ligados a temperatura de derretimento do filamento, além da necessidade de um ambiente fechado para evitar mudanças de condições.

Liu et al. (2017), na tentativa de diminuir o custo na impressão de grandes produtos para uso industrial da manufatura aditiva, estudaram a utilização de materiais plásticos em forma de *pellets*. No fim, utilizando uma extrusora com dois estágios de parafusos, foi

construído um protótipo que obteve êxito ao imprimir dois moldes de produção. Embora os moldes tenham sido imprimidos com sucesso, foi necessário a utilização de uma extrusora com pressão estável e um sistema de extrusão com controle do fluxo de derretimento devido a problemas com o derretimento e fluxo instáveis dos *pellets*.

Como uma tentativa de inovar a impressão 3D, Wang et al. (2016) desenvolveram o sistema de *fused pellet modeling* (FPM), que se trata de uma mini extrusora colocada em um robô industrial alimentado por materiais em formato granular. Nesse sistema, um parafuso de progressão especialmente desenvolvido é responsável por atribuir à impressão maior velocidade e qualidade e, além disso, o robô industrial proporciona ainda mais flexibilidade ao processo.

2.2. Algoritmos de visão computacional para análise de impressão 3D

Para monitorar o processo de impressão 3D, Jin, Zhang e Gu (2019) utilizaram uma *Convolutional Neural Network* (CNN), um algoritmo de *Deep Learning* que interpreta imagens, para verificar se na impressão ocorria deposição de material em excesso ou em falta. O algoritmo é primeiramente treinado para conseguir reconhecer padrões do processo para que, posteriormente, consiga reconhecer erros através da comparação de fotos tiradas durante o mesmo com o banco de dados analisado. Dessa maneira, a impressão pode ser corrigida simultaneamente, prevenindo erros. Como resultado, eles obtiveram um algoritmo capaz de prever com mais de 98% de precisão a qualidade da peça impressa.

Nuchitprasitchai, Roggemann e Pearce (2017) projetaram um sistema para monitoramento da impressão 3D através de 3 pares de câmeras espaçadas de modo a cobrir completamente o processo. Essas câmeras enviam imagens para um mini-computador Raspberry programado em Python, que as interpreta e percebe possíveis erros. O método retratado no artigo foi apresentado como bem sucedido, sendo 100% eficaz na detecção tanto da impressão normal quanto das falhas.

Holzmond e Li (2017), desenvolveram um método de detecção de erros utilizando duas câmeras através de *three-dimensional digital image correlation* (3D-DIC), um método óptico no qual são capturadas imagens estereoscópicas, que são correlacionadas espacialmente para fornecer informações sobre a geometria da peça estudada. Como resultado, o método desenvolvido se mostrou capaz de detectar e localizar tanto defeitos de área, como gotas, como defeitos gerais, como baixo fluxo de filamento. Após finalizar o

trabalho com sucesso, foi concluído que o método ainda possuia margem para aprimoramento.

Posteriormente, Petsiuk e Pearce (2020) desenvolveram um sistema de monitoramento através de um hardware baseado em visão computacional, que faz uma análise inteligente das camadas da impressão 3D, com o intuito de monitorar erros na produção e gerar possíveis ações para corrigi-los durante a fabricação. Esse sistema é formado por uma única câmera, posicionada em uma das arestas da impressão, de forma a conseguir detectar a maioria dos erros. As fotos tiradas pela câmera são então comparadas com o modelo base em um computador, que indica as ações necessárias para a correção. Como descrito pelos autores, o referido trabalho se mostrou uma boa ferramenta para poupar material e tempo na impressão 3D.

Jin et al. (2021) apresentaram um modelo onde uma câmera era colocada na extrusora e era utilizado um sistema de detecção chamado *you only look once* (YOLO) para identificar erros que causam imperfeições no produto final, de forma a corrigir tais erros e assegurar a implementação da impressão 3D na indústria. Como conclusão, o algoritmo se mostrou preciso na identificação de erros, mas foram encontrados desafios em mudar os parâmetros da impressão durante a análise.

Liu et al. (2019) utilizaram um *proportion integrative derivative* (PID), uma técnica de controle de processos, para desenvolver um sistema de controle de qualidade da impressão 3D durante o processo. Os parâmetros do mesmo eram ajustados com auxílio de um algoritmo diagnóstico de imagem baseado em análise textural, responsável por indicar defeitos em imagens adquiridas ao decorrer do processo. Como conclusão, os autores obtiveram um modelo satisfatório, embora ainda tivesse algumas limitações relacionadas a modelos complexos e ao tempo de resposta da máquina ao controle.

2.3. Machine Learning aplicado na manufatura aditiva

Qi et al. (2019) analisaram a utilização de redes neurais na manufatura aditiva, discutindo sua efetividade em superar obstáculos ligados à conexão entre parâmetros de processo, estrutura e performance das peças. A pesquisa enfatizou o uso das redes neurais em atividades como monitoramento em tempo real, avaliação de qualidade e criação de modelos 3D, tirando proveito de sua habilidade de identificar padrões complexos sem a exigência de

modelos físicos explícitos. Embora tenham feito contribuições relevantes, os autores destacaram obstáculos, como a fusão de dados desiguais e restrições computacionais, propondo soluções e perspectivas futuras para expandir o efeito dessas tecnologias na manufatura aditiva.

Wang et al. (2020) analisaram o funcionamento do aprendizado de máquina em diferentes etapas da manufatura aditiva, abrangendo o projeto de materiais, otimização de parâmetros de processo e monitoramento de defeitos. A pesquisa destacou a aplicação de métodos como o aprendizado supervisionado e redes neurais convolucionais para aprimorar a eficácia e a qualidade do procedimento. Além disso, os escritores ressaltaram a capacidade do aprendizado de máquina para automação e detecção de irregularidades, sugerindo novos caminhos de investigação para superar obstáculos técnicos e expandir as utilizações no setor industrial.

2.4. Considerações a respeito da literatura

A revisão da literatura ajuda a evidenciar tendências recentes na viabilização de fabricação de peças de grande porte via manufatura aditiva. Destaca-se ainda o emprego do uso de manipuladores robóticos para esse tipo de procedimento, fator que incorpora a flexibilidade de manobras e precisão desse tipo de máquina ao processo de impressão 3D. Observa-se ainda a grande relevância atual de se implementar sistemas de monitoramento do processo de impressão para garantir a deposição adequada de material na peça, o que implica também na possibilidade de evitar desperdício de material e corrigir eventuais falhas durante o procedimento. Além disso, também é possível notar implementações de algoritmos de aprendizado de máquina na tentativa de otimizar o processo da manufatura aditiva. Nesse sentido, foram observados vários trabalhos usando técnicas de processamento de imagens associadas a algoritmos de inteligência artificial.

3. Fundamentação teórica

Para a análise e classificação das superfícies das peças impressas serão utilizados algoritmos de visão computacional e algoritmos de aprendizado de máquina. Estes, serão mais detalhados nas seções deste tópico.

3.1. Local Binary Pattern (LBP)

O LBP é um operador de texturas que rotula os pixels de uma imagem através da comparação de um pixel central com seus vizinhos mais próximos e representa essas comparações com um vetor de números binários (PIETIKÄINEN et al., 2011). Primeiramente a imagem é traduzida de RGB, que é o conjunto de vetores que formam uma imagem colorida, para escala de cinzas. Isso ocorre pois uma imagem em RGB possui 3 valores para cada pixel, um valor para vermelho (R), um para verde (G) e um para azul (B), enquanto uma imagem em escala de cinzas possui apenas um valor entre o branco e o preto, com tons de cinza intermediários, o que facilita a análise pelo algoritmo. Após isso, é feita uma comparação entre cada pixel e sua vizinhança. Onde cada pixel vizinho receberá 1 caso seu número na escala de cinzas seja menor ou igual ao número dado ao ponto central, e receberá 0 caso contrário, como exemplificado na Figura 1. Isso forma um padrão binário que representa a relação entre um pixel central e os demais ao seu redor.

Figura 1 - Funcionamento do LBP.

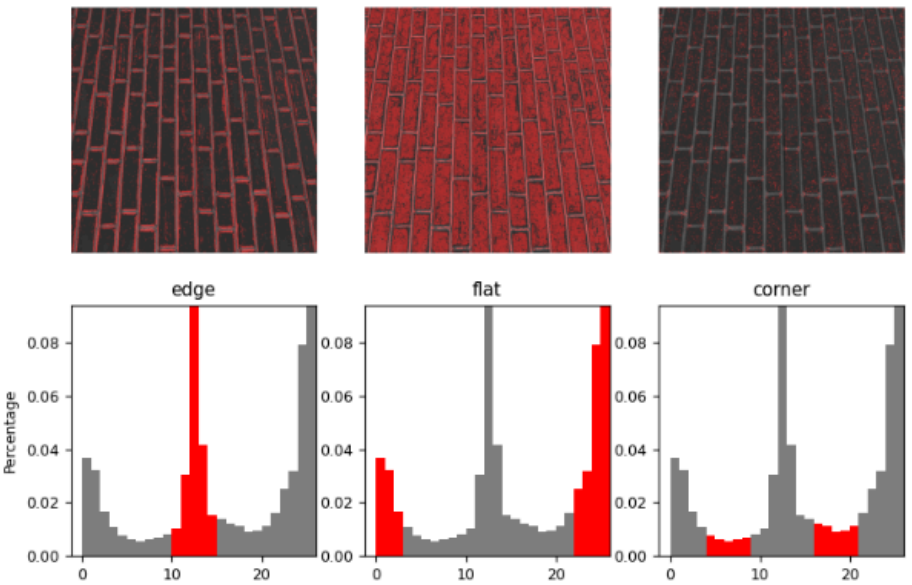
70	30	30
0	50	60
20	30	90

1	0	0
0		1
0	0	1

Fonte: Elaborado pelo próprio autor

A partir disso, um vetor será criado contendo os zeros e uns e seus valores serão somados conforme sua posição de 0 a 7, no qual a posição 0 tem valor 20, a posição 1 tem valor 21 e assim por diante. Desse modo, é possível avaliar o quão distintos são pontos próximos. A Figura 2 exemplifica a análise feita através do LBP bem como um histograma dos valores extraídos em toda imagem.

Figura 2 - Análise feita por LBP



Fonte: https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_local_binary_pattern.html.

3.2. Aprendizado de máquina

O uso de aprendizado de máquina (*Machine Learning*) em visão computacional consiste na aplicação de algoritmos capazes de aprender a partir de dados visuais, como imagens e vídeos, para realizar tarefas como classificação, detecção de objetos, segmentação e análise preditiva. No presente trabalho serão utilizados modelos de *Random Forest* e *K-Nearest Neighbors* (KNN), que serão explicados abaixo.

Além disso, vale destacar que algoritmos de aprendizado de máquina funcionam com base em números aleatórios e, portanto, podem sofrer a influência deles e gerar resultados não satisfatórios. Desse modo, é importante que o usuário tome as medidas possíveis para diminuir essa aleatoriedade, mantendo uma base de dados com números suficientes de amostras e com resultados bem distribuídos para evitar enviesamento.

3.2.1. Random Forest

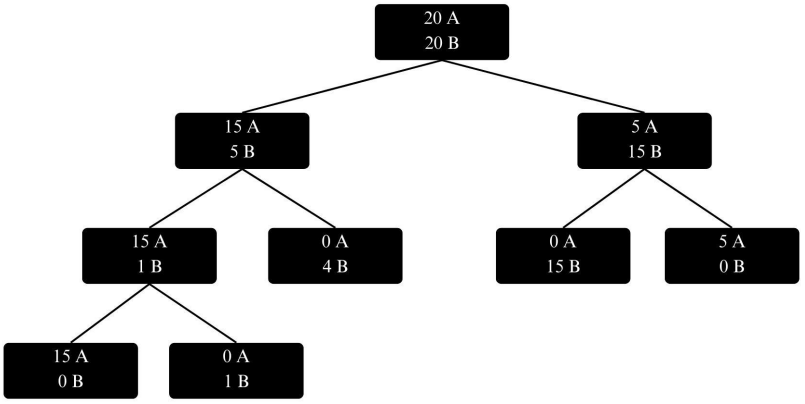
O algoritmo Random Forest é um dos métodos mais eficazes e amplamente utilizados para classificação e regressão em aprendizado de máquina. Ele se baseia na combinação de múltiplas árvores de decisão para melhorar a precisão das previsões e reduzir o risco de

overfitting, situação na qual o algoritmo se acostuma tanto com a base de dados de treinamento, que se torna incapaz de classificar um indivíduo novo (Biau & Scornet, 2016).

O funcionamento do Random Forest se baseia na construção de um conjunto de árvores de decisão, cada uma treinada com um subconjunto aleatório dos dados. Cada árvore toma decisões com base em critérios de divisão, como o índice de Gini, por exemplo, para dividir os dados em subgrupos mais homogêneos. Uma vez que todas as árvores são construídas, a previsão final do Random Forest é obtida por meio da junção das previsões individuais das árvores. No caso de classificação, a classe final é determinada por maioria de votos entre as árvores.

Explicando de forma resumida através da Figura 3, que exemplifica uma árvore de decisão aplicada em dados divididos em dois grupos, A e B: Primeiramente, os grupos estão divididos de forma igual. Nesse momento, o algoritmo usaria uma lógica para a separação desses indivíduos, como por exemplo, se o valor de uma característica x for maior que um valor, ele vai pra direita, se for menor ou igual a esse mesmo valor, ele vai pra esquerda. Desse modo são são obtidas duas ramificações (o número de ramificações não é limitada a duas, a característica pode dividir em mais faixas) e, a partir daqui, o algoritmo identifica se as ramificações estão homogêneas (se possuem indivíduos de apenas uma classe) e, assim, decidem continuar ou não a separação.

Figura 3 - Funcionamento de uma árvore de decisão.



Fonte: Elaborado pelo próprio autor

Vale ressaltar que o número de ramificações que a árvore terá pode ser decidido pelo usuário. Isso é importante pois uma árvore com um número alto de ramificações pode estar sujeita a se tornar ruim na classificação de novos indivíduos por estar muito acostumada com

a base de dados de treinamento. Além disso, dando fim a explicação, para a classificação de um novo indivíduo, ele passará pelas divisões realizadas na árvore de decisão até chegar em um ponto onde não há mais ramificações e então será classificado de acordo com o grupo em maioria neste ponto.

Uma das grandes vantagens do Random Forest é sua capacidade de lidar com dados de alta dimensionalidade e sua robustez contra ruídos. No entanto, sua complexidade computacional pode ser um desafio para grandes conjuntos de dados, uma vez que o tempo de treinamento cresce com o número de árvores na floresta.

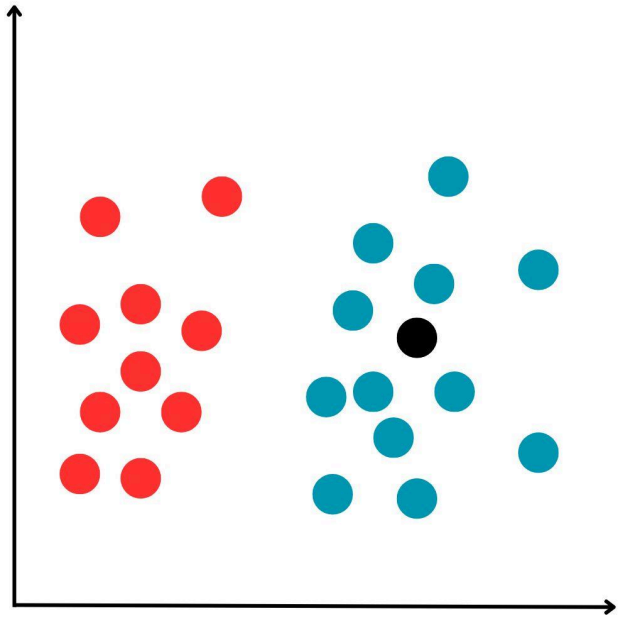
3.2.2. K-Nearest Neighbors

O *K-Nearest Neighbors* (KNN) é um método de classificação que se baseia na similaridade entre os atributos dentro de um conjunto de dados. Ele pode ser considerado como um algoritmo de aprendizado preguiçoso, pois não constroi um modelo explícito durante sua fase de treinamento. Ao invés disso, ele guarda todos os atributos dos dados utilizados no treinamento e os usa diretamente para classificar novos indivíduos.

O funcionamento do KNN funciona com base na identificação dos k vizinhos mais próximos de um dado a ser classificado. Para determinar a proximidade, o algoritmo utiliza medidas de distância, como a distância Euclidiana, por exemplo. Uma vez que são identificados os vizinhos mais próximos, a classificação do novo indivíduo é definida através de um critério de maioria, ou seja, é atribuída a ele a classe mais frequente entre seus vizinhos identificados (Guo et al., 2003).

De maneira mais fácil, explicando através da Figura 4: Supondo que o algoritmo de KNN está buscando classificar um indivíduo em dados com duas características, divididos em dois grupos, um azul e um vermelho. É possível pensar que o algoritmo criaria um espaço com duas dimensões (o número de características), com cada característica sendo representada em um eixo e, então, distribuiria os indivíduos nesse espaço. Quando fosse adicionado um novo indivíduo, ele seria inserido nesse espaço e, através disso, seria possível dizer a qual grupo ele pertence observando sua vizinhança. Na Figura 4, o novo indivíduo está representado como uma bola preta e, como ele está rodeado por bolas azuis, ele será classificado como azul. Se a vizinhança estivesse mais diversificada, o algoritmo levaria em conta variáveis como distância entre os vizinhos, grupo dos vizinhos, entre outros pontos.

Figura 4 - Funcionamento do algoritmo de KNN.



Fonte: Elaborado pelo próprio autor

Uma das principais desvantagens do KNN é sua dependência da escolha do valor de k. Valores pequenos de k podem tornar o algoritmo mais sensível a ruídos na base de dados, enquanto valores muito altos podem prejudicar os limites de decisão. Além disso, o KNN tem um custo computacional elevado na fase de classificação, pois exige o cálculo das distâncias entre o novo indivíduo e todos os dados do conjunto de treinamento.

3.2.3. Redes Neurais Convolucionais na classificação de imagens

As Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNNs) são redes neurais projetadas especificamente para processar dados estruturados espacialmente, sendo um algoritmo de aprendizagem profunda comumente utilizado para classificar imagens, por exemplo (JIN et al., 2019). Inspiradas no funcionamento do córtex visual humano, elas analisam uma imagem de forma hierárquica, identificando desde padrões simples, como bordas, até estruturas complexas, como objetos inteiros.

O processo começa com camadas convolucionais que aplicam filtros à imagem, gerando mapas de características que destacam padrões específicos. Em seguida, camadas de pooling reduzem a dimensionalidade desses mapas, preservando as informações mais relevantes e tornando o modelo mais eficiente. Por fim, as camadas totalmente conectadas

utilizam as características extraídas para realizar a tarefa final, como classificação ou regressão. De maneira resumida, como dito em Jin et al. (2019), as CNNs transformam uma imagem em uma matriz de números que representam sua categoria, e então o modelo é treinado com essa matriz de números e utilizado para prever resultados em novos dados.

De maneira simplificada, utilizando como exemplo um algoritmo que tenta classificar imagens como casas: Primeiramente, durante o treinamento, o algoritmo tentaria criar filtros, através de matrizes "escaneando" as imagens, que representassem características que possibilitam a identificação de uma casa. Para pessoas, essas características poderiam ser a presença de janelas, portas, telhado, paredes, entre outras, por exemplo. Após criar uma certa quantidade de filtros, o algoritmo faz otimizações e, por fim, está pronto para a classificação. Durante a classificação de um novo indivíduo, as matrizes com os filtros varrem a imagem procurando as características obtidas no treinamento, ou seja, se a imagem possui janelas, portas, telhado, etc. Por fim, quanto mais dessas características uma imagem possuir, maior a chance dela representar uma casa.

As CNNs são invariantes a deslocamentos e variações em escala de tamanho, tornando-as extremamente eficazes em visão computacional. Elas aprendem automaticamente as características relevantes dos dados, eliminando a necessidade de extração manual de atributos. Por isso, são amplamente usadas em tarefas como classificação de imagens, detecção de objetos e inspeção visual, sendo uma ferramenta indispensável em diversos campos tecnológicos.

3.3. Avaliação do desempenho

Para avaliar a eficácia de um algoritmo em tarefas de classificação, utilizam-se métricas como precisão (precision), sensibilidade (recall) e F1-score, que fornecem informações detalhadas sobre o desempenho do modelo:

- Precisão (Precision): Mede a proporção de previsões corretas para uma determinada classe em relação ao total de previsões feitas para essa classe. Um valor alto de precisão indica que a maioria das amostras classificadas como pertencentes a uma classe realmente pertence a ela.
- Sensibilidade (Recall): Também chamada de revocação, avalia a capacidade do modelo de identificar corretamente todas as amostras de uma classe

específica. Um alto recall significa que poucas instâncias dessa classe foram erroneamente ignoradas.

- F1-Score: Representa a média harmônica entre precisão e sensibilidade, equilibrando ambas as métricas. É especialmente útil quando há desbalanceamento entre as classes, garantindo que o modelo não favoreça apenas uma delas.

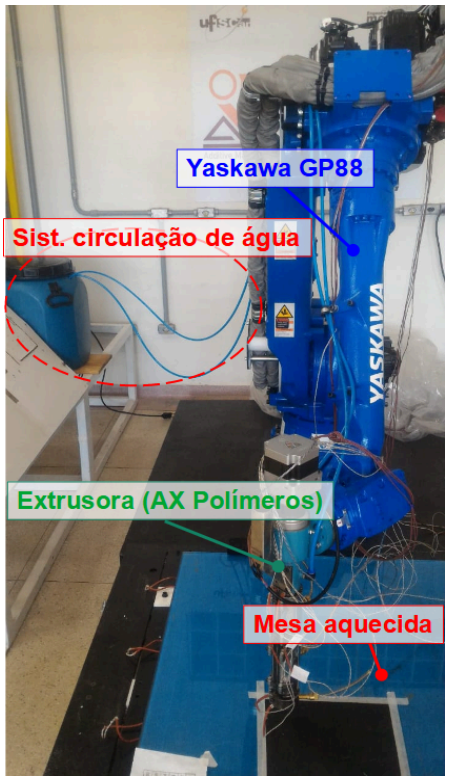
Essas métricas são fundamentais para interpretar os resultados de uma CNN e garantir que ela esteja classificando imagens com precisão e eficiência.

Além disso, também serão utilizadas para ajudar no entendimento dos modelos, matrizes de confusão. Essas matrizes possuem representadas em suas linhas o real grupo das imagens e em suas colunas o grupo no qual as imagens foram classificadas pelo algoritmo, de modo que a diagonal principal dessa matriz mostre os indivíduos classificados de maneira correta e os demais espaços, os indivíduos classificados de forma errônea. Essa matriz possibilita visualizar de forma mais geral quais foram os principais erros cometidos pelo algoritmo.

4. Materiais e métodos

O presente projeto de pesquisa tem como principal objetivo apresentar a análise de diferentes tipos de algoritmo de aprendizado de máquina na classificação de textura superfícies de peças manufaturas via impressão 3D robotizada através de imagens capturadas durante o processo. Para isso, será utilizada uma célula de manufatura aditiva robotizada (CMAR) equipada com uma impressora 3D. A CMAR é composta por um manipulador robótico Yaskawa GP88, capaz de suportar até 88 kg e com 6 eixos de movimentação, que será responsável por controlar uma extrusora mono-rosca fabricada pela AX Polímeros (Pulquerio et al., 2024). Essa extrusora possui uma rotação de até aproximadamente 60 RPM e é alimentada com *pellets* poliméricos. Além disso, a CMAR é equipada com uma mesa aquecida, composta por três chapas de aquecimento com potência de 2 kW cada, utilizada para aprimorar a aderência e garantir a homogeneização da temperatura da peça durante o processo de impressão. Por fim, o sistema conta com um sistema de circulação de água, composto por uma bomba e um reservatório de 50 L, responsável pelo resfriamento da extrusora na região de alimentação da máquina. A Figura 5 apresenta uma ilustração dos subsistemas que compõem a CMAR.

Figura 5 - Subsistemas da célula de manufatura aditiva robotizada.

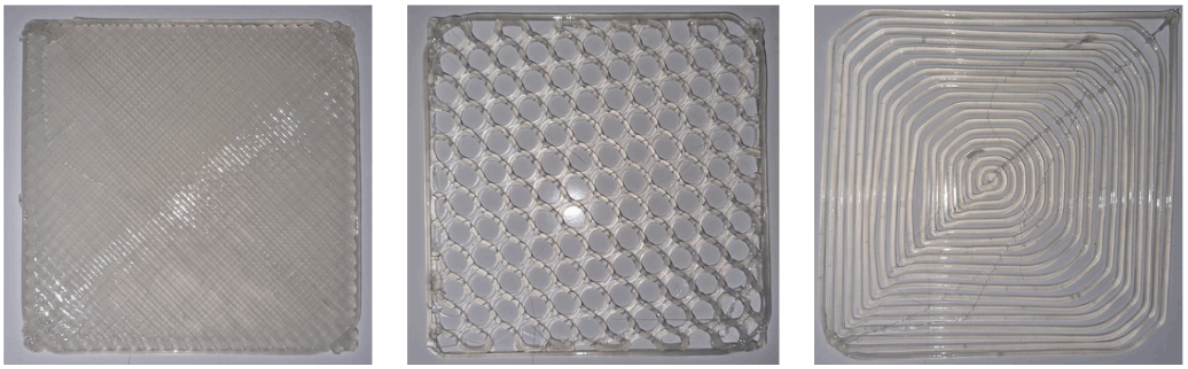


Fonte: elaborada pelo próprio autor.

A fabricação de peças na CMAR foi realizada por meio do software CAD SIEMENS Nx, utilizado para gerar uma geometria tridimensional da peça. Em seguida, o arquivo CAD foi fatiado pelo software Ultimaker Cura, no qual foram configurados parâmetros como altura das camadas a serem impressas e velocidade de impressão. Esse processo resultou na geração de um G-Code, comumente usado para impressão 3D em máquinas tradicionais. No caso específico deste projeto, o G-Code foi passado pela ferramenta RoboDK, que gerou o código para a programação offline do manipulador robótico responsável por movimentar-se durante a fabricação da peça.

Foram impressas peças com 3 diferentes texturas: Linear, Colmeia e Concêntrica, que podem ser observadas na Figura 6. Além disso, cada textura será impressa de 3 maneiras diferentes: regular (sem erros), sub extrudada (com falta de material) e com bolhas (quando o material não é suficientemente aquecido). Isso será feito para simular diferentes ocasiões da impressão para que o algoritmo seja testado de forma mais abrangente.

Figura 6 - Texturas impressas.



Fonte: Elaborada pelo próprio autor

Foi aplicado um sistema de visão computacional para a captura de imagens das camadas depositadas a partir do CMAR. O sistema foi composto pelos seguintes componentes:

- Câmera Raspberry Pi 5MP (sensor OV5647, abertura (F): 1.8);

- 1 computador de placa única, modelo Raspberry Pi 3 B+, com processador de 1GHz Single-core, GPIO (do inglês General Purpose Input/Output) de 40 pinos e memória de 512MB.

A câmera foi acoplada em um suporte próximo a impressão para que as imagens pudessem ser capturadas durante a mesma. Um algoritmo em Python para captura do processo foi implementado na Raspberry Pi e, com isso, foi possível obter os frames da impressão para análise.

A partir disso, foram realizadas 4 análises diferentes com as imagens obtidas, divididas em 9 classes diferentes, mostradas na Figura 7. Para gerar a impressão regular e os defeitos de sub extrusão e bolhas, alguns parâmetros da impressão foram variados, os valores de velocidade de impressão, referente a velocidade de movimentação do manipulador robótico, e da rotação do fuso da extrusora, para as três variações, pode ser observado na Tabela 1. Além disso, para a geração das bolhas, foi necessário utilizar material sem um processo de secagem antes, permitindo que umidade entrasse na extrusora. Os demais parâmetros foram mantidos constantes durante a impressão. A extrusora possui três zonas de aquecimento do material, que foram mantidas em 175, 180 e 190 graus Celsius, sendo a última referente ao bico da extrusora.

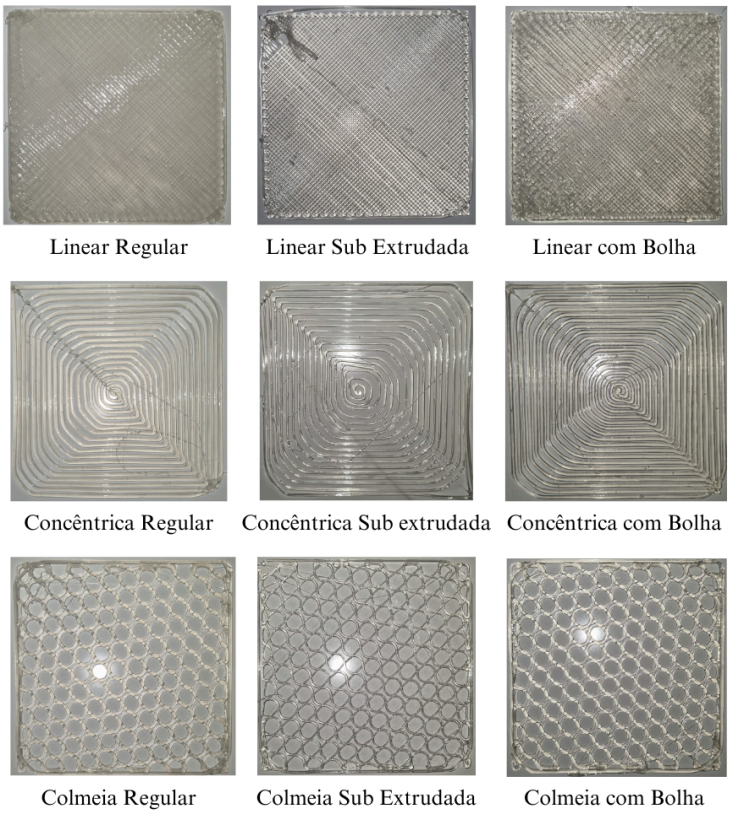
Tabela 1 - Parâmetros da impressão.

Parâmetro	**Regular**	**Sub Extrudada**	**Com Bolha**
Velocidade de impressão (mm/s)	15	15	15
Rotação do fuso (RPM)	30	10	30

Fonte: elaborada pelo próprio autor

Para os algoritmos de CNN, KNN e Random Forest, foi utilizada a linguagem de programação Python. Para o algoritmo de CNN, foi utilizada a biblioteca TensorFlow, já para os algoritmos de KNN e Random Forest, foi utilizada a biblioteca Scikit-learn. Além disso, para as duas últimas análises foi utilizado o método de Local Binary Pattern, através da biblioteca Scikit-image antes dos algoritmos referidos porque tanto o Random Forest, quanto o K-Nearest Neighbors não são ideais para a classificação de imagem.

Figura 7 - Classes das amostras.



Fonte: Elaborada pelo próprio autor

5. Resultados e discussão

Foram realizadas análises com com os 3 tipos de algoritmo: Redes Neurais Convolucionais, *Random Forest* e *K-Nearest Neighbors*. Através da captura das impressões, foi possível extrair 375 imagens. Além disso, com o objetivo de testar o desempenho das Redes Neurais Convolucionais com um número menor de amostras, foram selecionadas 89 das 375 imagens. A distribuição desses grupos nas texturas impressas é exibida na Tabela 2.

Tabela 2 - Número de imagens por textura para cada grupo.

	Colmeia	**Concêntrica**	**Linear**
Grupo de 375 imagens	82	154	139
Grupo de 89 imagens	31	26	32

Fonte: elaborada pelo próprio autor

5.1. Análise utilizando Redes Neurais Convolucionais

A primeira análise realizada com a Rede Neural Convolucional (CNN) utilizou um conjunto de 375 imagens, classificadas em nove categorias distintas. O desempenho do modelo foi avaliado por meio das métricas precisão (precision), sensibilidade (recall) e F1-score, conforme apresentado na Tabela 3.

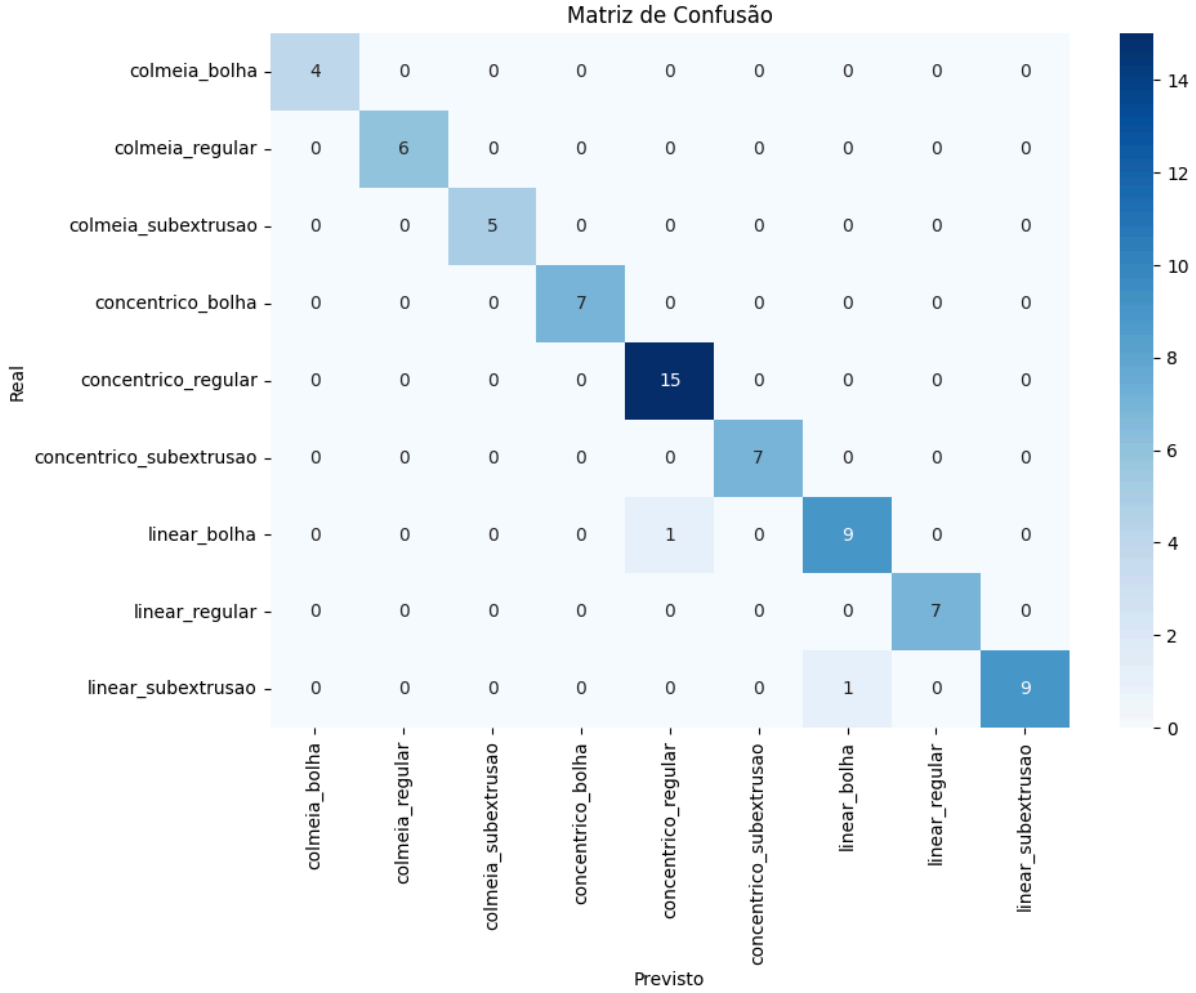
Tabela 3 - Resultado da Análise com CNN utilizando 375 imagens.

	precisão	**sensibilidade**	**f1-score**	**amostras**
Colmeia com Bolha	1.00	1.00	1.00	4
Colmeia Regular	1.00	1.00	1.00	6
Colmeia Sub Extrudada	1.00	1.00	1.00	5
Concêntrica com Bolha	1.00	1.00	1.00	7
Concêntrica Regular	0.94	1.00	0.97	15
Concêntrica Sub Extrudada	1.00	1.00	1.00	7
Linear com Bolha	0.90	0.90	0.90	10
Linear Regular	1.00	1.00	1.00	7
Linear Sub Extrudada	1.00	0.90	0.95	10
Acurácia Total	0.97			

Fonte: elaborada pelo próprio autor

Os resultados demonstram um alto desempenho da CNN, com F1-score igual ou superior a 0.90 para todas as classes. As imagens da textura colmeia obtiveram todas precisão e sensibilidade 1, mesmo tendo menos amostras no total, o que mostra que essa textura pode ser considerada de classificação mais fácil para o algoritmo. As imagens das textura linear e concêntrica também conseguiram um bom desempenho no geral, mas nem todas possuem precisão 1 (o que indica que o modelo classificou imagens de outra textura como elas) ou sensibilidade 1 (o que indica que o modelo classificou algumas de suas imagens em outras classes).

Figura 8 - Matriz de confusão da Análise com CNN utilizando 375 imagens.



Fonte: elaborada pelo próprio autor

Analisando a matriz de confusão dessa análise, contida na Figura 8, é possível observar que uma imagem de textura linear com bolha foi classificada como concêntrica regular e também que houve uma confusão na classificação de uma das imagens dentro da

textura linear. Isso pode sugerir que essas texturas são mais difíceis de serem diferenciadas no algoritmo, ou que a quantidade de amostras dessas classes foi insuficiente.

Posteriormente, foi realizada uma segunda análise com as Redes Neurais Convolucionais (CNN) utilizando um conjunto reduzido de 89 imagens, com o objetivo de observar a influência do número de amostras no desempenho do algoritmo de CNN. Nesse caso, foram utilizadas as mesmas métricas para análise. Os resultados podem ser vistos na Tabela 4.

Tabela 4 - Resultado da Análise com CNN utilizando 89 imagens.

	precisão	**sensibilidade**	**f1-score**	**amostras**
Colmeia com Bolha	0.00	0.00	0.00	2
Colmeia Regular	0.40	1.00	0.57	2
Colmeia Sub Extrudada	1.00	0.50	0.67	2
Concêntrica com Bolha	0.00	0.00	0.00	1
Concêntrica Regular	1.00	0.50	0.67	2
Concêntrica Sub Extrudada	0.50	1.00	0.67	1
Linear com Bolha	0.67	1.00	0.80	2
Linear Regular	1.00	1.00	1.00	1
Linear Sub Extrudada	0.00	0.00	0.00	2
Acurácia Total	0.51			

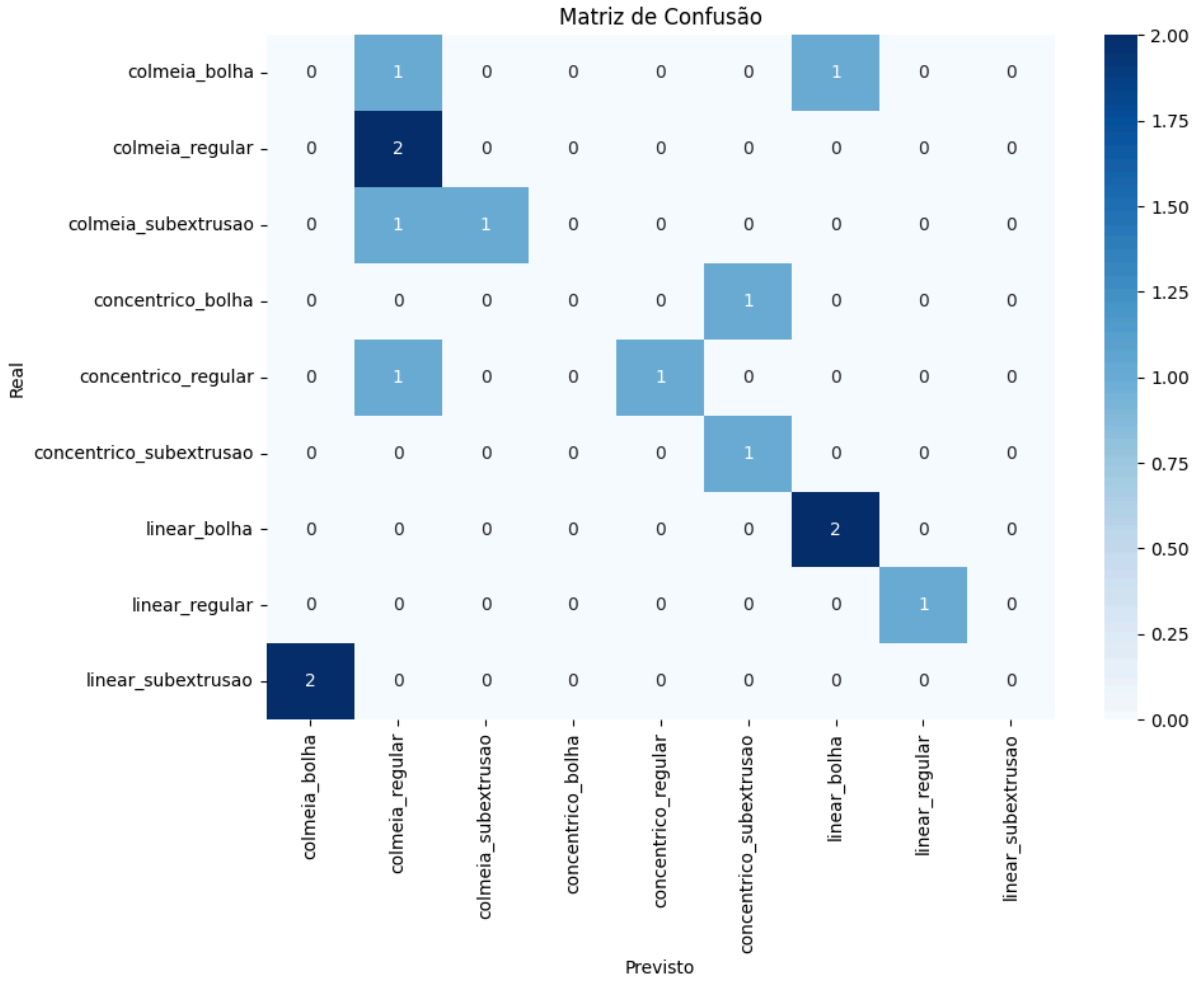
Fonte: elaborada pelo próprio autor

Os resultados desta análise indicam uma grande variação no desempenho do modelo em comparação com a análise anterior que utilizou 375 imagens. Algumas observações importantes incluem desempenho 0 em algumas classes, indicando que o modelo não conseguiu classificar nenhuma imagem dessas categorias e, também, a maioria das classes que obtiveram precisão 1, possuem sensibilidade menor, o que indica que o modelo não consegue classificar de maneira assertiva todas as amostras dessas classes. Além disso, existem ocasiões onde a precisão é menor que 1 e a sensibilidade é 1, o que indica que imagens erradas foram atribuídas para essa classe.

Observando a Matriz de confusão contida na Figura 9 é possível ver que houveram muitas classificações erradas. Os resultados desta análise sugerem que o baixo número de

amostras pode ter impactado negativamente o aprendizado da CNN, reduzindo sua capacidade de generalização.

Figura 9 - Matriz de confusão da Análise com CNN utilizando 89 imagens.



Fonte: elaborada pelo próprio autor

5.2. Análise utilizando Random Forest

Nesta análise, foi utilizado o algoritmo Random Forest para a classificação das imagens, utilizando um conjunto de 375 amostras. A Tabela 5 apresenta os resultados obtidos, considerando as métricas de precisão (precision), sensibilidade (recall) e F1-score.

A análise dos resultados revela algumas tendências importantes. Os resultados tiveram bastante alteração, enquanto algumas classes ficaram com precisão 1, outras ficaram com precisão 0. Classes como a Colmeia com bolha, Colmeia com Sub Extrusão e a Linear com Sub Extrusão tiveram bons resultados em precisão, mas baixas sensibilidades, sugerindo

que essas classes tiveram suas amostras classificadas erroneamente. Além disso, a classe Linear com Bolha teve um F1-Score baixo, o que sugere que o modelo pode estar confundindo essa classe com outras semelhantes, possivelmente devido a padrões visuais próximos.

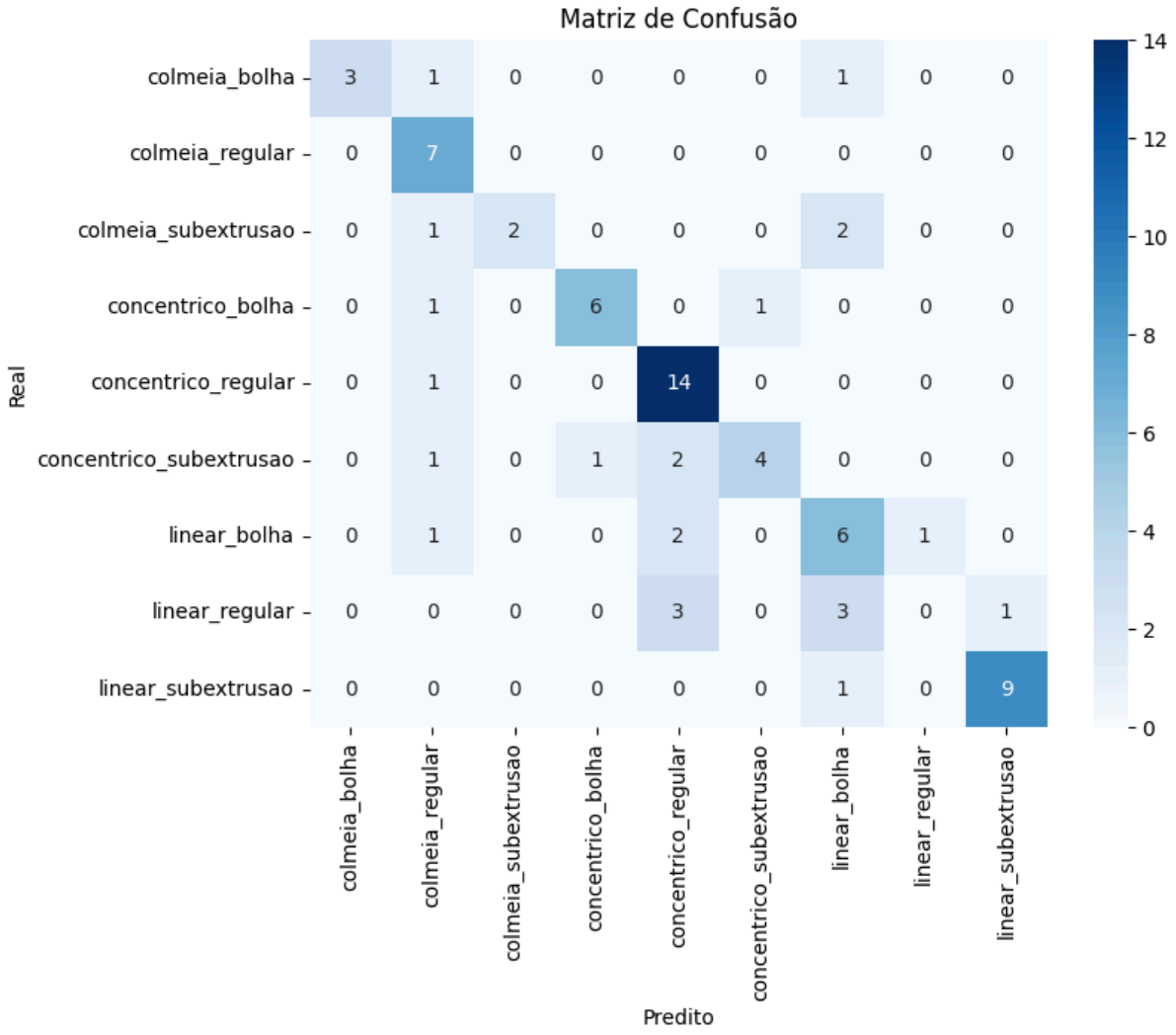
Tabela 5 - Resultado da Análise com Random Forest utilizando 375 imagens.

	precisão	**sensibilidade**	**f1-score**	**amostras**
Colmeia com Bolha	1.00	0.60	0.75	5
Colmeia Regular	0.54	1.00	0.70	7
Colmeia Sub Extrudada	1.00	0.40	0.57	5
Concêntrica com Bolha	0.86	0.75	0.80	8
Concêntrica Regular	0.67	0.93	0.78	15
Concêntrica Sub Extrudada	0.80	0.50	0.62	8
Linear com Bolha	0.46	0.60	0.52	10
Linear Regular	0.00	0.00	0.00	7
Linear Sub Extrudada	0.90	0.90	0.90	10
Acurácia Total	0.68			

Fonte: elaborada pelo próprio autor

Os resultados indicam que o Random Forest apresentou desempenho satisfatório em algumas classes, mas teve dificuldades na generalização para outras categorias, o que também pode ser visto na matriz de confusão contida na Figura 10.

Figura 10 - Matriz de confusão da Análise com Random Forest utilizando 375 imagens.



Fonte: elaborada pelo próprio autor

5.3. Análise utilizando K-Nearest Neighbors com 375 imagens

Nesta etapa, foi utilizada a abordagem de K-Nearest Neighbors (KNN) para a classificação das imagens. A Tabela 6 apresenta os resultados obtidos, considerando as métricas de precisão (precision), sensibilidade (recall) e F1-score.

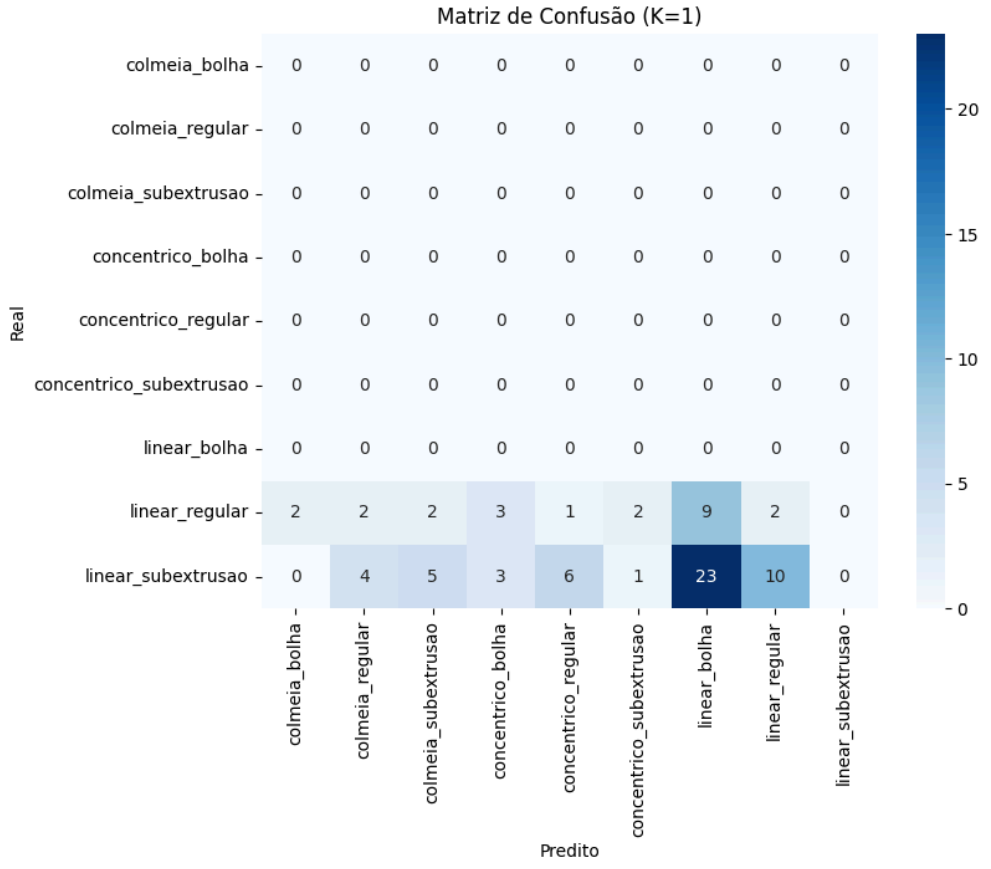
Os resultados obtidos demonstram um desempenho extremamente insatisfatório do KNN na classificação das imagens. Diversos fatores fazem parte dessa baixa performance, como ausência de classificação, o baixo desempenho da classe Linear Regular, mesmo sendo a única classe com classificação correta, e o erro total na classe Linear com Sub Extrusão, mesmo sendo a classe com maior número de amostras.

Tabela 6 - Resultado da Análise com KNN utilizando 375 imagens.

	precisão	sensibilidade	f1-score	amostras
Colmeia com Bolha	0.00	0.00	0.00	0
Colmeia Regular	0.00	0.00	0.00	0
Colmeia Sub Extrudada	0.00	0.00	0.00	0
Concêntrica com Bolha	0.00	0.00	0.00	0
Concêntrica Regular	0.00	0.00	0.00	0
Concêntrica Sub Extrudada	0.00	0.00	0.00	0
Linear com Bolha	0.00	0.00	0.00	0
Linear Regular	0.17	0.09	0.11	23
Linear Sub Extrudada	0.00	0.00	0.00	52
Acurácia Total	0.05			

Fonte: elaborada pelo próprio autor

Figura 11 - Matriz de confusão da Análise com KNN utilizando 375 imagens.



Fonte: elaborada pelo próprio autor

A Figura 11 apresenta as matrizes de confusão dos testes realizados, evidenciando as dificuldades do modelo KNN em classificar corretamente as imagens. Esses resultados indicam que o KNN pode não ser a abordagem ideal para esse tipo de classificação de imagens.

6. Considerações finais

Neste trabalho, foram realizadas quatro análises distintas utilizando diferentes abordagens de aprendizado de máquina para a classificação de imagens de impressões 3D. Os algoritmos testados foram Redes Neurais Convolucionais (CNN), Random Forest (RF) e K-Nearest Neighbors (KNN), com experimentos conduzidos em conjuntos de 375 e 89 imagens. A acurácia dos modelos pode ser observada na Tabela 7.

Tabela 7 - Acurácia dos modelos.

	Redes Neurais		**Random Forest**	**KNN**
Número de amostras	375	89	375	375
Acurácia do modelo	0.97	0.51	0.68	0.05

Fonte: elaborada pelo próprio autor

Os resultados obtidos evidenciam grande variação no desempenho dos modelos, destacando a importância da escolha do algoritmo para esse tipo de problema. A CNN demonstrou o melhor desempenho geral, obtendo valores máximos de precisão, recall e F1-score na análise com 375 imagens, com uma acurácia total de 97%. Isso confirma a eficácia das redes neurais convolucionais para reconhecimento de padrões em imagens, tornando-as a abordagem mais indicada para a tarefa proposta.

Já o Random Forest, embora tenha apresentado resultados satisfatórios em algumas classes, terminando com uma acurácia de 68%, revelou dificuldades em identificar certos padrões. O desempenho do modelo foi inferior ao da CNN, possivelmente devido à dependência de características extraídas manualmente, enquanto a CNN aprende automaticamente os padrões relevantes.

A análise com KNN foi a que apresentou os piores resultados, com F1-score próximo de zero na maioria das classes e uma acurácia de 5%. O modelo falhou completamente na identificação de padrões na maioria das categorias, possivelmente devido à alta dimensionalidade dos dados e à escolha inadequada de hiperparâmetros. Esse resultado indica que o KNN não é adequado para esse tipo de problema, sendo necessária a adoção de métodos mais avançados.

Além disso, a análise realizada com apenas 89 imagens revelou uma queda significativa na performance da CNN, indo de 97% para 51% de acurácia, indicando que o

tamanho do conjunto de dados influencia diretamente na capacidade de generalização do modelo. Isso reforça a necessidade de um volume adequado de amostras para garantir uma classificação precisa e confiável.

Referências

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 14724: Informação e documentação: trabalhos acadêmicos: apresentação. Rio de Janeiro, 2003. 11 p.

BRASIL. Medida Provisória n.1.569-9, de 11 de dezembro de 1997. Estabelece multa em operações de importação, e da outras providências. **Diário Oficial [da] República Federativa do Brasil**, Poder Executivo, Brasília, DF, 14 dez. 1997. Seção 1, p. 29514.

BRAYNER, A. R. A.; MEDEIROS, C. B. Incorporação do tempo em SGB orientado a objetos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 9., 1994, São Paulo. Anais... São Paulo: USP, 1994. p. 16-29.

GARCIA, J. C. R.Revista científicas eletrônicas: discussões em sete momentos. **Ciência da Informação**,Brasília,DF,v. 40, n. 1, jan./abr. 2011, p. 3-7. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652011000100009&lng=pt&nrm=iso&tlng=pt>. Acesso em: 11 abr. 2017.

LIMA, J. A. DE O. Pesquisa-ação em Ciência da Informação. In: MULLER, S. P. M. (Org.). **Métodos para a pesquisa em Ciência da Informação**. Brasília: Thesaurus, 2007. cap. 3, p. 63-82.

MENDONÇA, E. S. Estudo dos elementos de pesquisa das teses de doutorado em ciência da informação do convênio IBICT/UFRJ-ECO. **Ciência da Informação**, Brasília, v. 40 , n. 3 , p. 396-412 , set./dez. 2011.

THEREZA, W. B. **Ambiente para especificação de aplicações multimídia com suporte de qualidade de serviço.** 2004. 100f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de São Carlos, São Carlos, 2006.

UNIVERSIDADE FEDERAL DE SÃO CARLOS. Biblioteca Comunitária. **Guia para elaboração de Referências**: de acordo com ABNT NBR 6023/2002**.** Disponível em: <http://www.bco.ufscar.br/servicos-bco/capacitacao-do-usuario/normalizacao-de-trabalhos/guia-para-elaboracao-de-referencias/view>. Acesso em: 12 abr. 2017.

YIN, R. K. **Estudo de caso**: planejamento e métodos. 5. ed. Porto Alegre: Bookman, 2015. 290 p.

BIAU, Gérard; SCORNET, Erwan. A random forest guided tour. Test, v. 25, p. 197-227, 2016.

GAO, Wei et al. The status, challenges, and future of additive manufacturing in engineering. Computer-Aided Design, v. 69, p. 65-89, 2015.

GUO, Gongde et al. KNN model-based approach in classification. In: **On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings.** Springer Berlin Heidelberg, 2003. p. 986-996.

HOLZMOND, Oliver; LI, Xiaodong. In situ real time defect detection of 3D printed parts. **Additive Manufacturing**, v. 17, p. 135-142, 2017.

JIN, Zeqing et al. Precise localization and semantic segmentation detection of printing conditions in fused filament fabrication technologies using machine learning. **Additive Manufacturing**, v. 37, p. 101696, 2021.

JIN, Zeqing; ZHANG, Zhizhou; GU, Grace X. Autonomous in-situ correction of fused deposition modeling printers using computer vision and deep learning. **Manufacturing Letters**, v. 22, p. 11-15, 2019.

LIU, Chenang et al. Image analysis-based closed loop quality control for additive manufacturing with fused filament fabrication. **Journal of Manufacturing Systems**, v. 51, p. 75-86, 2019.

LIU, Xiaojun et al. A large‐scale double‐stage‐screw 3 D printer for fused deposition of plastic pellets. **Journal of Applied Polymer Science**, v. 134, n. 31, p. 45147, 2017.

LU, Jiwen; LIONG, Venice Erin; ZHOU, Jie. Simultaneous local binary feature learning and encoding for homogeneous and heterogeneous face recognition. **IEEE transactions on pattern analysis and machine intelligence**, v. 40, n. 8, p. 1979-1993, 2017.

NUCHITPRASITCHAI, Siranee; ROGGEMANN, Michael C.; PEARCE, Joshua M. Three hundred and sixty degree real-time monitoring of 3-D printing using computer analysis of two camera views. **Journal of Manufacturing and Materials Processing**, v. 1, n. 1, p. 2, 2017.

PETSIUK, Aliaksei L.; PEARCE, Joshua M. Open source computer vision-based layer-wise 3D printing analysis. **Additive Manufacturing**, v. 36, p. 101473, 2020.

PIETIKÄINEN, Matti et al. Computer vision using local binary patterns. Springer Science & Business Media, 2011.

Pulquerio, E. C., Barbosa, G. F., & Shiki, S. B. (2024). Robotic additive manufacturing system: development of suitable range of process parameters for 3D printing of a large-sized object in PLA polymer. Progress in Additive Manufacturing, 1–12. Springer.

QI, Xinbo et al. Applying neural-network-based machine learning to additive manufacturing: current applications, challenges, and future perspectives. Engineering, v. 5, n. 4, p. 721-729, 2019.

SHAH, J. et al. Large-scale 3D printers for additive manufacturing: design considerations and challenges. **The International Journal of Advanced Manufacturing Technology**, v. 104, n. 9, p. 3679-3693, 2019.
WANG, Chengcheng et al. Machine learning in additive manufacturing: State-of-the-art and perspectives. Additive Manufacturing, v. 36, p. 101538, 2020.

WANG, Zhiyuan et al. Large-scale deposition system by an industrial robot (I): design of fused pellet modeling system and extrusion process analysis. **3D printing and additive manufacturing**, v. 3, n. 1, p. 39-47, 2016.

WONG, Kaufui V.; HERNANDEZ, Aldo. A review of additive manufacturing. **International scholarly research notices**, v. 2012, 2012.

Apêndice A - Código com algoritmo de CNN

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import os
import numpy as np
import random
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from google.colab import drive

# Fixando a aleatoriedade
seed = 44
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)

# Definindo os parâmetros
drive.mount('/content/drive')
data_dir = "/content/drive/MyDrive/dataset"
img_size = (128, 128)
batch_size = 32
num_classes = len(os.listdir(data_dir))

# Carregando pré-processando as imagens
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
```

```
)

val_generator = datagen.flow_from_directory(
    data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation',
    shuffle=False  # Importante para matriz de confusão
)

# Definindo o modelo CNN
model = keras.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(128, 128,
3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])

# Compilando o modelo
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Treinando o modelo
model.fit(train_generator, validation_data=val_generator, epochs=10)

# Salvando o modelo
model.save("cnn_model.h5")
```

```
# Função para testar o modelo com imagens de validação automaticamente
def predict_validation_images(model, val_generator):
    images, labels = next(val_generator)  # Pegando um batch de validação
    predictions = model.predict(images)
    predicted_classes = np.argmax(predictions, axis=1)
    actual_classes = np.argmax(labels, axis=1)
    class_labels = list(val_generator.class_indices.keys())

    for i in range(len(images)):
        print(f"Imagem {i+1}: Previsto = {class_labels[predicted_classes[i]]}, Real = {class_labels[actual_classes[i]]}")

# Carregando modelo salvo
model = keras.models.load_model("cnn_model.h5")

# Prevendo todas as imagens do conjunto de validação para matriz de confusão
val_images, val_labels = [], []

for _ in range(len(val_generator)):
    images, labels = next(val_generator)
    val_images.append(images)
    val_labels.append(labels)

val_images = np.concatenate(val_images)
val_labels = np.concatenate(val_labels)

# Fazendo previsões no conjunto de validação
predictions = model.predict(val_images)
predicted_classes = np.argmax(predictions, axis=1)
actual_classes = np.argmax(val_labels, axis=1)

# Gerando matriz de confusão
conf_matrix = confusion_matrix(actual_classes, predicted_classes)
```

```
class_labels = list(val_generator.class_indices.keys())


# Plotando a matriz de confusão

plt.figure(figsize=(10, 7))

sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=class_labels, yticklabels=class_labels)

plt.xlabel("Previsto")

plt.ylabel("Real")

plt.title("Matriz de Confusão")

plt.show()


# Exibindo relatório de classificação

print(classification_report(actual_classes, predicted_classes,
target_names=class_labels))
```

Apêndice B - Código com algoritmo de Random Forest

```
import cv2

import numpy as np

import os

import random

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

from google.colab import drive

from skimage.feature import local_binary_pattern

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

from sklearn.tree import plot_tree


# Travando a SEED para garantir reprodutibilidade

SEED = 42

random.seed(SEED)

np.random.seed(SEED)


# Definindo o caminho do dataset no Google Drive

drive.mount('/content/drive')

dataset_path = "/content/drive/MyDrive/datasetpplus"


# Parâmetros do LBP

radius = 3  # Raio do vizinho

n_points = 8 * radius  # Número de pontos vizinhos
```

```
# Listando as classes (nomes das pastas)
classes = sorted(os.listdir(dataset_path))  # Ordena para manter a mesma ordem sempre

X = []  # Vetores de características
y = []  # Rótulos

# Processando cada imagem em cada classe
for label, class_name in enumerate(classes):
    class_path = os.path.join(dataset_path, class_name)

    for img_name in sorted(os.listdir(class_path)):  # Ordena para garantir mesma ordem sempre
        img_path = os.path.join(class_path, img_name)

        # Carregando a imagem em escala de cinza
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if img is None:
            continue  # Ignorar imagens inválidas

        # Aplicando LBP
        lbp = local_binary_pattern(img, n_points, radius, method="uniform")

        # Criando histograma normalizado
        hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), density=True)

        # Armazenando os dados
        X.append(hist)
```

```
        y.append(label)


# Convertendo para arrays numpy
X = np.array(X)
y = np.array(y)


# Separando em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=SEED, stratify=y)


# Treinando o Random Forest
clf = RandomForestClassifier(n_estimators=100, random_state=SEED)
clf.fit(X_train, y_train)


# Fazendo previsões
y_pred = clf.predict(X_test)


# Avaliando o modelo
acc = accuracy_score(y_test, y_pred)
print(f"Acurácia: {acc:.4f}")


# Gerando matriz de confusão
cm = confusion_matrix(y_test, y_pred)


# Criando um heatmap para visualizar a matriz de confusão
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=classes,
yticklabels=classes)
plt.xlabel("Predito")
plt.ylabel("Real")
```

```
plt.title("Matriz de Confusão")

plt.show()


# Exibindo uma árvore do Random Forest (apenas a primeira, para não ficar pesado)

plt.figure(figsize=(20, 10))

plot_tree(clf.estimators_[0], filled=True, feature_names=[f'Bin {i}' for i in range(len(X_train[0]))], class_names=classes, rounded=True)

plt.title("Exemplo de Árvore do Random Forest")

plt.show()


# Gerando relatório de classificação

report = classification_report(y_test, y_pred, target_names=classes, output_dict=True)


# Convertendo para um DataFrame e exibir como tabela

df_report = pd.DataFrame(report).transpose()

print("\nRelatório de Classificação:\n")

print(df_report)


# Exibindo tabela formatada

plt.figure(figsize=(8, 4))

sns.heatmap(df_report.iloc[:-1, :].T, annot=True, cmap="coolwarm", linewidths=0.5, fmt=".2f")

plt.title("Métricas de Classificação")

plt.show()
```

Apêndice C - Código com algoritmo de Random Forest

```
import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

from google.colab import drive

from mpl_toolkits.mplot3d import Axes3D

from skimage.feature import local_binary_pattern

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
ConfusionMatrixDisplay, classification_report

from sklearn.decomposition import PCA

import pandas as pd

import seaborn as sns


# Parâmetros do LBP

radius = 3  # Raio do vizinho

n_points = 8 * radius  # Número de pontos vizinhos

method = 'uniform'


# Definindo o caminho do dataset no Google Drive

drive.mount('/content/drive')

folder_path = "/content/drive/MyDrive/datasetpplus"  # Ajuste para seu
caminho no Drive


# Função para extrair LBP

def extract_lbp_features(image):

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    lbp = local_binary_pattern(gray, n_points, radius, method)
```

```
    hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, n_points +
3), range=(0, n_points + 2))
    hist = hist.astype('float') / (hist.sum() + 1e-6)  # Normaliza
    return hist


# Carregando imagens e rótulos
X, y = [], []
labels = sorted(os.listdir(folder_path))  # Ordena os rótulos para
remover aleatoriedade
for label in labels:
    label_path = os.path.join(folder_path, label)
    if os.path.isdir(label_path):
        for file in sorted(os.listdir(label_path)):  # Ordena arquivos
para manter ordem fixa
            img_path = os.path.join(label_path, file)
            image = cv2.imread(img_path)
            if image is not None:
                X.append(extract_lbp_features(image))
                y.append(label)

X = np.array(X)
y = np.array(y)

# Divisão fixa dos dados (80% treino, 20% teste)
train_size = int(0.8 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Testando diferentes valores de K
k_values = range(1, 21)
```

```
accuracies = []

for k in k_values:

    knn = KNeighborsClassifier(n_neighbors=k)

    knn.fit(X_train, y_train)

    y_pred = knn.predict(X_test)

    accuracies.append(accuracy_score(y_test, y_pred))


# Plotando gráfico da acurácia vs. K

plt.figure(figsize=(8, 5))

plt.plot(k_values, accuracies, marker='o', linestyle='-')

plt.xlabel('Número de Vizinhos (K)')

plt.ylabel('Acurácia')

plt.title('Acurácia do KNN para diferentes valores de K')

plt.grid()

plt.show()


# Melhor K

best_k = k_values[np.argmax(accuracies)]

best_knn = KNeighborsClassifier(n_neighbors=best_k)

best_knn.fit(X_train, y_train)

y_pred_best = best_knn.predict(X_test)

final_accuracy = accuracy_score(y_test, y_pred_best)

print(f'Acurácia final com K={best_k}: {final_accuracy:.4f}')


# Matriz de confusão

cm = confusion_matrix(y_test, y_pred_best, labels=labels)

plt.figure(figsize=(8,6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels,
yticklabels=labels)
```

```
plt.xlabel('Predito')

plt.ylabel('Real')

plt.title(f'Matriz de Confusão (K={best_k})')

plt.show()


# Relatório de classificação

report = classification_report(y_test, y_pred_best,
target_names=labels, output_dict=True)

df_report = pd.DataFrame(report).transpose()

print(df_report)


# Reduzindo dimensionalidade para 3D

pca = PCA(n_components=3)

X_pca = pca.fit_transform(X)


# Separando treino e teste em 3D

X_train_pca, X_test_pca = X_pca[:train_size], X_pca[train_size:]


# Criando gráfico 3D

fig = plt.figure(figsize=(10, 7))

ax = fig.add_subplot(111, projection='3d')


# Cores para cada classe

unique_labels = np.unique(y)

colors = plt.cm.jet(np.linspace(0, 1, len(unique_labels)))

label_color_map = {label: colors[i] for i, label in
enumerate(unique_labels)}


# Plotando pontos de treino

target_colors = [label_color_map[label] for label in y_train]
```

```
ax.scatter(X_train_pca[:, 0], X_train_pca[:, 1], X_train_pca[:, 2],
c=target_colors, marker='o', label='Treino')


# Plotando pontos de teste

target_colors_test = [label_color_map[label] for label in y_test]

ax.scatter(X_test_pca[:, 0], X_test_pca[:, 1], X_test_pca[:, 2],
c=target_colors_test, marker='x', label='Teste')


ax.set_xlabel('Componente Principal 1')

ax.set_ylabel('Componente Principal 2')

ax.set_zlabel('Componente Principal 3')

ax.set_title('Distribuição das Amostras no Espaço PCA 3D')


# Criando legenda

legend_patches = [plt.Line2D([0], [0], marker='o', color='w',
markerfacecolor=colors[i], markersize=10, label=label) for i, label in
enumerate(unique_labels)]

plt.legend(handles=legend_patches, title="Classes")

plt.show()
```