UNIVERSIDADE FEDERAL DE SÃO CARLOS – UFSCAR

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA – CCET

DEPARTAMENTO DE ENGENHARIA MECÂNICA – DEMEC

THIAGO HENRIQUE ARAGÃO SILVA

ALGORITMO DE VISÃO COMPUTACIONAL PARA MONITORAMENTO DIMENSIONAL E IDENTIFICAÇÃO DE DEFEITOS EM PEÇAS FABRICADAS POR MANUFATURA ADITIVA ROBOTIZADA

SÃO CARLOS

2025

THIAGO HENRIQUE ARAGÃO SILVA

ALGORITMO DE VISÃO COMPUTACIONAL PARA MONITORAMENTO DIMENSIONAL E IDENTIFICAÇÃO DE DEFEITOS EM PEÇAS FABRICADAS POR MANUFATURA ADITIVA ROBOTIZADA

Trabalho de conclusão de curso apresentada 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
Coorientador: Prof. Dr. Luis Antonio Oliveira Araujo

São Carlos
2025

*

Thiago Henrique Aragão Silva
Algoritmo de Visão Computacional para Monitoramento Dimensional e Identificação de Defeitos em Peças Fabricadas por Manufatura Aditiva Robotizada/ Thiago Henrique Aragão Silva. – São Carlos, 2025-
43p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Sidney Bruce Shiki

Trabalho de conclusão de curso – Universidade Federal de São Carlos – UFSCar
Centro de Ciências Exatas e de Tecnologia – CCET
Departamento de Engenharia Mecânica – DEMec, 2025.

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

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 17/2025/CCEMec/CCET

Graduação: Defesa Pública de Trabalho de Conclusão de Curso

Folha Aprovação (GDP-TCC-FA)

FOLHA DE APROVAÇÃO

THIAGO HENRIQUE ARAGAO SILVA

ALGORITMO DE VISÃO COMPUTACIONAL PARA MONITORAMENTO DIMENSIONAL E ANÁLISE DA QUALIDADE EM PEÇAS FABRICADAS POR MANUFATURA ADITIVA ROBOTIZADA

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos – Campus São Carlos

São Carlos, 25 de fevereiro de 2025

ASSINATURAS E CIÊNCIAS

Cargo/Função	Nome Completo
Orientador	Sidney Bruce Shiki
Membro da Banca 1	Luis Antônio Oliveira Araujo
Membro da Banca 2	Eduardo Costa Pulquerio

Documento assinado eletronicamente por **Luis Antonio Oliveira Araujo**, **Docente**, em 25/02/2025, às 15:59, 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 **Sidney Bruce Shiki**, **Docente**, em 25/02/2025, às 16:12, 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 **1757110** e o código CRC **71C3AC8E**.

Referência: Caso responda a este documento, indicar expressamente o Processo nº 23112.005198/2025-12

SEI nº 1757110

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

Documento assinado digitalmente
EDUARDO COSTA PULQUERIO
Data: 25/02/2025 18:56:27-0300
Verifique em https://validar.iti.gov.br

Dedico este trabalho aos meus pais, pelo apoio e incentivo constantes. Sem vocês, nada disso seria possível.

AGRADECIMENTOS

A jornada para a conclusão deste trabalho foi marcada por desafios, aprendizados e muito crescimento. Nada disso teria sido possível sem o apoio de pessoas especiais que estiveram ao meu lado em cada etapa.

Agradeço à minha família, que sempre me incentivou e acreditou no meu potencial, estando presente nos momentos decisivos e desafiadores. O suporte de vocês foi fundamental para que eu chegasse até aqui.

Aos meus amigos, que compartilharam essa caminhada acadêmica comigo, enfrentando desafios, trocando conhecimento e trazendo leveza para os momentos mais intensos. A parceria de vocês tornou essa experiência ainda mais enriquecedora e especial.

Aos meus professores do curso de Engenharia Mecânica, sou imensamente grato pelo conhecimento transmitido, pela paciência e pelo compromisso em formar não apenas profissionais, mas também cidadãos críticos e preparados para os desafios do futuro. Em especial, agradeço ao meu orientador, Sidney Bruce Shiki, cuja orientação, dedicação e apoio foram essenciais para o desenvolvimento deste trabalho.

A todos que, de alguma forma, contribuíram para esta conquista, meu sincero obrigado!

"The way to succeed is to double your failure rate."

Thomas J. Watson

RESUMO

A crescente complexidade dos produtos gerou a necessidade de novas técnicas na produção, como a manufatura aditiva (impressão 3D). Neste trabalho foi desenvolvido um algoritmo de visão computacional para monitorar dimensões e retilineidade de peças ao longo do processo de manufatura aditiva robotizada. O sistema utilizou técnicas avançadas de processamento de imagens para identificar inconformidades geométricas, assegurando alta precisão e qualidade nas peças produzidas. A pesquisa implementou uma célula de manufatura aditiva robotizada (CMAR) que trabalhou com polímero ABS. O algoritmo foi testado e validado com sucesso. Os resultados evidenciaram que o sistema foi eficiente na detecção de falhas dimensionais e geométricas, contribuindo para a integridade das peças impressas. Além disso, o monitoramento automatizado proporcionou benefícios significativos, como a redução de custos devido ao menor desperdício de material, a melhoria da qualidade do produto final e a promoção de práticas mais sustentáveis no processo de fabricação.

Palavras-chave: Visão computacional. Manufatura aditiva robotizada. Monitoramento dimensional.

ABSTRACT

The increasing complexity of products has necessitated the development of new production techniques, such as additive manufacturing (3D printing). This work developed a computer vision algorithm to monitor the dimensions and straightness of parts throughout the robotic additive manufacturing process. The system used advanced image processing techniques to identify geometric non conformities, ensuring high precision and quality in the produced parts. The research implemented a robotic additive manufacturing cell (CMAR) that operated with ABS polymer. The algorithm was rigorously tested and successfully validated. The results showed that the system was effective in detecting dimensional and geometric faults, contributing to the integrity of the printed parts. Furthermore, automated monitoring offered significant benefits, including cost reduction due to less material waste, improved final product quality, and the promotion of more sustainable manufacturing practices.

Keywords: Computer vision, Robotic additive manufacturing, Dimensional monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FPM	Fused Pellet Modeling
MA	Manufatura Aditiva
CMAR	Célula de Manufatura Aditiva Robotizada
CAD	Computer-Aided Design
SLA	Stereolithography
SLS	Selective Laser Sintering
DoF	Degree of Freedom
FFF	Fused Filament Fabrication
RI	Região de Interesse

SUMÁRIO

1 INTRODUÇÃO

O aumento da qualidade de produtos e as constantes atualizações que os tornaram mais complexos geraram a necessidade de criação de novas técnicas e ferramentas para o processo de produção. Nesse sentido, a manufatura aditiva (MA), mais conhecida como impressão 3D, tem ganhado cada vez mais destaque no desenvolvimento de produtos de diferentes materiais.

A MA é um processo de fabricação que se baseia na adição consecutiva de um determinado material, formando camadas que dão origem a um produto (VOLPATO, 2017). Para realizar essa tarefa, as camadas são adicionadas de acordo com uma representação tridimensional feita a partir de um software CAD (do inglês *computer-aided design*). Esse processo possui inúmeras vantagens quando comparado à manufatura tradicional, como sustentabilidade devido ao menor desperdício de material, utilização mais eficiente de energia, grande liberdade geométrica na fabricação, possibilidade de prototipagem rápida e versatilidade que contribui para a fabricação de geometrias complexas (VOLPATO, 2017). Em suma, trata-se de um processo automatizado que possibilita fabricar produtos mais complexos e de diferentes formas com menor desperdício de material.

Partindo do mesmo princípio de fabricação, a necessidade de produzir peças maiores com mais rapidez deu origem à manufatura aditiva robotizada. Ela consiste na utilização de uma extrusora em tamanho reduzido acoplada à extremidade de um braço robótico, o que permite maior flexibilidade e rapidez para a manufatura (WANG, 2016). Nesse sistema, é utilizado material na forma de grãos (pellets) que são fundidos e transmitidos até o bocal da extrusora, onde ocorre a deposição do material. Consequentemente, eliminam-se defeitos de deposição que acontecem na manufatura aditiva convencional.

Porém, a complexidade dessas peças carece de um controle de qualidade das diferentes etapas do processo para verificar e corrigir possíveis inconformidades durante a impressão 3D. Além das dimensões, características geométricas como a retilineidade são fundamentais para garantir a precisão e a funcionalidade das peças. Nesse contexto, o monitoramento do processo de MA por meio de captura de imagens tem sido bastante difundido ao longo dos últimos anos, graças aos avanços tecnológicos em termos de hardware e ao aprendizado de máquina (*machine learning*). O aprendizado de máquina consiste em um sistema capaz de modificar seu comportamento autonomamente de acordo com sua própria experiência, visando melhorar seu desempenho.

A demanda por essa tecnologia tem aumentado com a abundância de dados disponíveis, que possibilitam a extração de informações relevantes (MAHESH, 2020). Nos últimos anos, essa técnica integrada à tecnologia dos gêmeos digitais tem sido cada vez mais implementada no monitoramento do processo de impressão 3D por meio de captura de imagens. O método dos

gêmeos digitais baseia-se na utilização de algoritmos capazes de comparar uma imagem real de uma peça sendo impressa com um banco de dados de imagens virtuais que indicam como ela devia estar em determinado momento do processo (PARASKEVOUDIS; KARAYANNIS; KOUMOULOS, 2020).

Nesse sentido, Moretti, Rossi e Senin (2021) desenvolveram um algoritmo de rede neural convolucional que possibilita a identificação de defeitos nas camadas da impressão 3D durante o processo de fabricação. Esse algoritmo foi implementado em MATLAB e é capaz de extrair informações do código de fabricação (G-Code - linguagem de programação para máquinas-ferramentas) de cada camada da peça para reconstruir, digitalmente, a geometria dos limites de cada camada e, por fim, alinhar o processo de extrusão ao caminho gerado em vez do disponibilizado pela extrusora. Para isso, utilizaram uma câmera que gera as informações visuais para o computador, que, por sua vez, faz o processamento das capturas e compara os gêmeos digitais, detectando os defeitos geométricos do objeto físico, emitindo um alerta ou encerrando o processo para dar início às correções necessárias. Apesar do tempo de processamento ser longo, o modelo foi capaz de identificar defeitos locais e globais, como sobre-extrusão e deslocamento da peça, respectivamente.

Com base nessas inovações, o presente trabalho desenvolveu um algoritmo de visão computacional para o monitoramento dimensional e a verificação de retilineidade de peças durante o processo de manufatura aditiva robotizada. O objetivo é otimizar a detecção de inconformidades, como desvios dimensionais e irregularidades geométricas, garantindo a qualidade do produto final de maneira mais eficiente e precisa. A retilineidade, em particular, é uma característica crítica para peças que exigem alta precisão, e o algoritmo proposto foi projetado para avaliar essa propriedade de forma automatizada, contribuindo para a melhoria contínua dos processos de fabricação.

2 REVISÃO DA LITERATURA

A manufatura aditiva é um processo de fabricação que tem ganhado espaço nos últimos anos. Atualmente, esse processo evoluiu com os avanços tecnológicos permitindo a integração de braços robóticos com a extrusora. Contudo, a complexidade dessa tecnologia pode gerar maiores dificuldades para garantir qualidade da fabricação e o dimensionamento correto da peça.

Atualmente, pesquisas têm desenvolvido o controle da qualidade da impressão 3D baseado principalmente na utilização de técnicas de visão computacional. Com auxílio do aprendizado de máquina, é possível monitorar o processo de fabricação de modo a identificar defeitos mais rapidamente e, consequentemente, reduzir custos e tempo de operação. Além disso, pode-se fazer correções em tempo real que contribuem para a fabricação correta evitando desperdícios e diminuindo o tempo de operação.

2.1 MANUFATURA ADITIVA E MANUFATURA ADITIVA ROBOTIZADA

A manufatura aditiva (MA), mais conhecida como impressão 3D, representa um processo de fabricação capaz de dar forma a um objeto físico a partir de um modelo digital, geralmente a partir de um arquivo CAD com a geometria da peça. O produto desse processo é resultado da adição sucessiva, em forma de camadas, de um determinado material (WONG; HERNANDEZ, 2012).

Esse processo possui inúmeras vantagens quando comparado à manufatura tradicional, como sustentabilidade devido ao menor desperdício de material, utilização mais eficiente de energia, grande liberdade geométrica na fabricação, possibilidade de prototipagem rápida e versatilidade que contribui para possibilidade de fabricação de geometrias complexas (VOLPATO, 2017).

A precisão dimensional, incluindo a retilineidade das peças impressas, é importante para muitas aplicações, especialmente quando se trata de peças que exigem precisão em suas arestas e superfícies. A retilineidade é um parâmetro geométrico que envolve a avaliação de quão próximas as linhas e superfícies de uma peça estão da forma ideal especificada no modelo digital. Durante a impressão, podem ocorrer variações devido à precisão do equipamento, qualidade do material ou ao movimento do robô.

A classificação da MA pode ser feita de acordo com o princípio de processamento das camadas e, com isso, destacam-se: fotopolimerização em cuba, extrusão de material, jateamento de material, jateamento de aglutinante, fusão de leito de pó, adição de lâminas e deposição com energia selecionada (VOLPATO, 2017). Dentro dessas classificações, existem diversas tecnologias aplicadas à impressão 3D, sendo as principais a Modelagem por Fusão e Deposição (do inglês *fused filament fabrication* - FFF) que forma o objeto final a partir da extrusão de

um polímero, a Estereolitografia (do inglês *stereolithography* - SLA) que, por meio de luz ultravioleta, solidifica uma resina líquida e dá origem à peça e Sinterização Seletiva a Laser (do inglês *selective laser sintering* - SLS) que utiliza um laser de alta potência para sinterizar o pó e aglutinar as camadas do material formando o objeto final.

A necessidade de produção de peças em maiores escalas fez surgir novas tecnologias capazes de suprir essa demanda. Shah (2019) estudou o processo de impressão 3D de polímeros e projetou um protótipo de impressora 3D para produtos de dimensões elevadas. Foi possível obter resultados que indicam possibilidade de melhoria desse processo. Porém, o bocal de extrusão deu origem a problemas não vistos nas impressoras 3D de menores escalas, uma vez que o diâmetro do bico aumentou.

O avanço da tecnologia permitiu a implementação da chamada manufatura aditiva robotizada. Ela consiste na utilização de um braço robótico com uma extrusora na extremidade dele. A principal vantagem da utilização dessa técnica em relação a manufatura aditiva tradicional é que ela possibilita a produção de materiais de maiores dimensões, uma vez que o braço robótico, por ter vários graus de liberdade, permite que o material seja depositado em diferentes direções e planos (ZUTIN et al., 2024).

Wang (2016) aperfeiçoou a utilização da manufatura aditiva robotizada ao desenvolver o método *fused pellet modeling* (FPM). Essa tecnologia consiste em utilizar o material em formato de grãos e dispô-lo para o processo de aquecimento na extrusora com o auxílio de um parafuso de extrusão especialmente projetado. Ele movimenta o material até o bocal com uma geometria otimizada para eliminar os defeitos de deposição do produto final. Com isso, são proporcionadas diversas vantagens, como maior flexibilidade de produção, redução de tempo por conta da maior velocidade de processamento advinda do parafuso na extrusora e capacidade de fabricação de peças com geometrias ainda mais complexas do que as produzidas na MA.

2.2 ALGORITMOS DE VISÃO COMPUTACIONAL PARA ANÁLISE DE IMPRESSÃO 3D

A visão computacional busca desenvolver técnicas e algoritmos para que os computadores possam processar e interpretar informações visuais como imagem e vídeo. Essas informações podem ser geradas por uma ou mais câmeras sem restrição de angulação, a depender da aplicação.

O monitoramento automatizado na manufatura aditiva oferece benefícios como redução de custos, aumento da produtividade, melhoria na qualidade e sustentabilidade. O monitoramento contínuo e a detecção precoce de falhas geométricas ajudam a reduzir o desperdício de material e o tempo de retrabalho, contribuindo para a redução de custos de produção. Além disso, sistemas automatizados permitem ajustes em tempo real, sem a necessidade de intervenção manual, acelerando o processo de fabricação e melhorando a eficiência geral. A análise contínua das peças em produção garante que elas atendam aos requisitos dimensionais, incluindo a

retilineidade e a conformidade das superfícies, aumentando a consistência e a qualidade do produto final. O uso eficiente dos materiais e a redução de falhas durante o processo contribuem para práticas mais sustentáveis, minimizando o desperdício de material.

A análise e comparação dessas informações é feita aplicando a tecnologia dos Gêmeos Digitais. Ela consiste em comparar a imagem do produto sendo manufaturado com uma imagem digital da peça obtida por simulação computacional. Quando combinada ao aprendizado de máquina, essa técnica permite o monitoramento e otimização de processos de fabricação em tempo real, uma vez que essa tecnologia pode gerar uma previsão do que a visão computacional irá analisar (MORETTI; ROSSI; SENIN, 2021).

O aprendizado de máquina consiste em um sistema capaz de modificar seu comportamento autonomamente de acordo com sua própria experiência visando melhorar seu desempenho. A demanda por essa tecnologia tem aumentado com a abundância de dados disponíveis, que possibilitam a extração de informações relevantes (MAHESH, 2020).

Nos últimos anos, essa técnica integrada à tecnologia dos gêmeos digitais tem sido cada vez mais implementada no monitoramento do processo de impressão 3D por meio de captura de imagens. Nesse contexto, Moretti, Rossi e Senin (2021) desenvolveram um algoritmo de rede neural convolucional que possibilita a identificação de defeitos nas camadas da impressão 3D durante o processo de fabricação. Esse algoritmo foi implementado em MATLAB e é capaz de extrair informações do código de fabricação (G-Code - linguagem de programação para máquinas-ferramentas) de cada camada da peça para reconstruir, digitalmente, a geometria dos limites de cada camada para, por fim, alinhar o processo de extrusão ao caminho gerado em vez do disponibilizado pela extrusora. Para isso, utilizaram uma câmera que gera as informações visuais para o computador que, por sua vez, faz o processamento das capturas e compara os gêmeos digitais, detectando os defeitos geométricos do objeto físico emitindo um alerta ou encerrando o processo para dar início às correções necessárias. Apesar do tempo de processamento ser longo, o modelo foi capaz de identificar defeitos locais e globais, como sobre-extrusão e deslocamento da peça, respectivamente.

Seguindo o caminho de parada do processo de fabricação para correção, Langeland (2020), em sua tese de mestrado, desenvolveu um sistema com diversos algoritmos que detectam erros na impressão 3D por meio da visão computacional com o objetivo de classificar os maiores desafios na modelagem por fusão e deposição em tempo real. Partindo da imagem 3D da peça, o autor utiliza uma técnica de slicing(fatiamento) para dividir o modelo tridimensional em várias fatias de malha triangular através da interface Prusa Slicer para, com isso, obter um código de fabricação que, nesse caso, é o G-Code. Desse modo, o projeto focou na verificação da primeira camada de deposição e na análise do bocal de extrusão que garante o controle do restante da impressão pela detecção da falta de material a partir da distância entre o bocal e a primeira camada. Caso algum problema seja detectado, a impressão é automaticamente pausada pelo sistema de monitoramento para evitar desperdício de material e tempo. O modelo foi capaz de

detectar erros comuns à impressão 3D como deslocamento de camadas e movimentação da peça, porém se mostrou sensível às mudanças de iluminação e condições do ambiente.

Jin, Zhang e GU (2019) também fazem uso dessa técnica. O algoritmo de machine learning desenvolvido analisa a qualidade da extrusão a cada camada impressa por uma impressora 3D e faz a auto-correção da impressão. As informações a serem processadas são geradas por uma câmera próxima à extrusora. Para o processamento, existem três classificações de referência, sendo elas: *under-extrusion*, *good-quality*, *over-extrusion*. Para cada classificação é feita uma correção dos parâmetros de velocidade, pressão e altura da impressão em tempo real. O algoritmo desenvolvido apresentou mais de 98% de precisão para previsão da qualidade da impressão e demonstrou que a taxa de resposta do modelo atinge ou supera a reação humana.

Paraskevoudis, Karayannis e Koumoulos (2020) monitoram a qualidade da impressão 3D em tempo real por meio de visão computacional baseada em inteligência artificial. A imagem capturada pela câmera é processada por um algoritmo de rede neural capaz de detectar os chamados stringing defects na geometria da peça, que são fios formados entre dois pontos da peça. Com isso, foi possível fazer previsões a partir desse processamento com uma precisão de 44%.

Petsik e Pearce (2020) desenvolveram uma estrutura de hardware em conjunto com um algoritmo em código aberto que verifica o processo de impressão 3D monitorando erros associados a forma externa da peça e a qualidade interna das às camadas de deposição. Para isso, esse sistema dispõe de uma câmera externa à área de impressão que captura as imagens e transmite essas informações para o algoritmo que compara com diversos tipos de falhas catalogadas e gera algumas ações de autocorreção da impressora. Vale ressaltar que o projeto foi realizado mais com o intuito de economizar tempo e material do que desenvolver um algoritmo completo de correção da impressão.

Charalampous (2021) aprimoraram ainda mais a técnica de visão computacional para o controle da impressão 3D. A partir do uso de diversos algoritmos de aprendizado de máquina e do uso de diversos sensores, o processamento das informações capturadas pelas câmeras é feito comparando-se os dados obtidos em tempo real com um modelo digital 3D contendo a chamada nuvem de pontos. Essa "nuvem" compreende informações da estrutura interna da peça e, desse modo, permite uma análise mais eficaz dos defeitos do processo de fabricação. Esse projeto foi validado e possibilitou o controle da qualidade do processo de peças com geometrias mais complexas.

2.3 APLICAÇÕES DA VISÃO COMPUTACIONAL

A visão computacional é uma técnica que pode ser aplicada em diferentes escopos para controle da qualidade de um processo. Uma aplicação que tem ganhado maior notoriedade é a de monitoramento da qualidade de pintura. Nesse sentido, Xu (2020) utilizaram essa técnica alinhada

a um algoritmo de inteligência artificial para detecção de possíveis defeitos de arestas na pintura de componentes de um veículo. Os resultados desse trabalho mostraram excelente efetividade e acurácia, apesar de algumas limitações, como tempo de processamento e impossibilidade de analisar mais de um defeito em uma mesma captura de imagem.

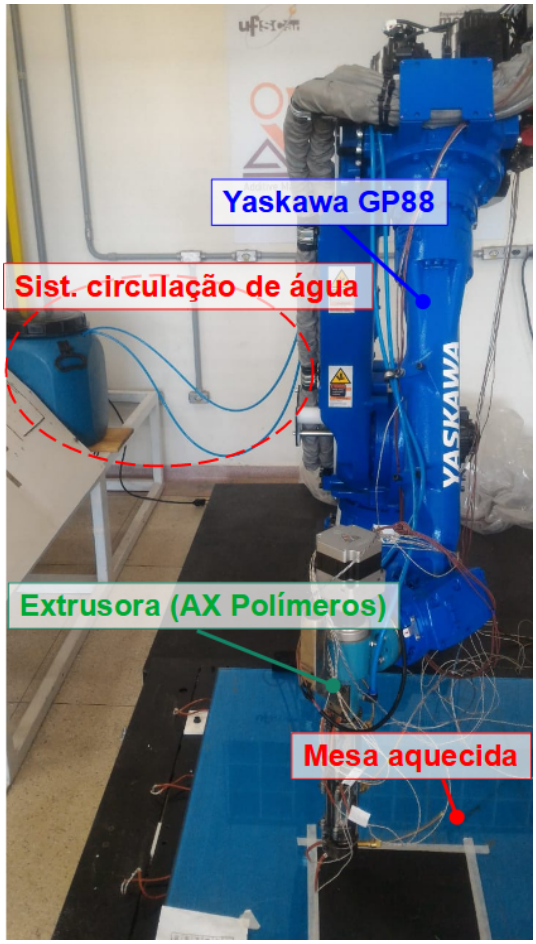
Em processos de fabricação tradicionais, a visão computacional pode ser utilizada para detectar possíveis falhas da máquina. Kurada e Bradley (1997) utilizaram o processamento de imagens para desenvolver sensores visuais capazes de estimar a condição de uma ferramenta de processo de fabricação de uma máquina CNC (controle numérico por computador). A captura de imagens possibilitou a obtenção de alguns parâmetros que os sensores convencionais não foram capazes de identificar, que descrevem as características do perfil de desgaste, como perímetro e área de desgaste. O método se mostrou rápido e eficaz, além de apresentar algumas vantagens como a não necessidade de contato para avaliação da condição da ferramenta e insensibilidade à variação de luz e vibração.

Ainda sobre processos tradicionais, Karthikeyan (2022) desenvolveram um algoritmo de inteligência artificial que utiliza a visão computacional para prever defeitos de rugosidade superficial do processo de usinagem de um eixo cilíndrico de alumínio. A técnica aplicada apresentou alta precisão, cerca de 96%, ao passo que apresentou um erro baixo em todos os experimentos realizados.

3 MATERIAIS E MÉTODOS

Nesta seção, são descritos os materiais e métodos utilizados para o monitoramento das dimensões e da retilineidade de peças fabricadas a partir de uma célula de manufatura aditiva robotizada (CMAR). A célula foi composta por um manipulador robótico Yaskawa GP88 com capacidade de 88 kg e 6 eixos de movimentação para o manejo de uma extrusora mono-rosca da fabricante AX Polímeros (rotação de até cerca de 60 RPM), alimentada com pellets poliméricos. A CMAR contou ainda com uma mesa aquecida composta por três chapas de aquecimento com potência de 2 kW, usada para melhorar a aderência e homogeneizar a temperatura da peça a ser impressa. Por último, o sistema também incluiu um sistema de circulação de água com bomba e reservatório de 50 L para resfriamento da extrusora na região de alimentação da máquina.

Figura 1 – Subsistemas da célula de manufatura aditiva robotizada.



Fonte: Elaborada pelo próprio autor.

Para efetuar a fabricação de peças na CMAR, foi empregado o software CAD SIEMENS Nx, que gerou a geometria tridimensional das peças. O arquivo CAD foi então fatiado utilizando o software Ultimaker Cura, que configurou a altura das camadas a serem impressas, a velocidade

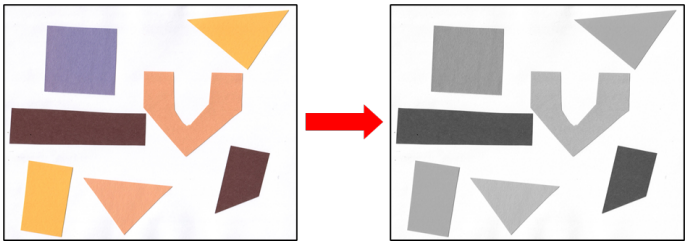
de impressão, entre outras características. O software permitiu gerar um código G (G-Code) comumente empregado para impressão 3D em máquinas tradicionais. No caso específico deste projeto, esse código foi processado pela ferramenta RoboDK, gerando o código para a programação offline do manipulador robótico, que executou os movimentos necessários para a fabricação da peça.

No presente projeto de pesquisa, um sistema de visão computacional foi desenvolvido para monitorar as dimensões e a retilineidade das peças durante o processo de impressão. O sistema foi composto pelos seguintes componentes:

- Câmera Raspberry Pi 5MP (sensor OV5647, abertura (F): 1.8);
- 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 posicionada em uma base fixa próxima à mesa aquecida ilustrada na Figura 1. Ela foi direcionada para a peça e capturou vídeos, que foram processados para obter imagens da peça ao longo de sua fabricação. No computador Raspberry Pi 3, foi implementado um código escrito em Python, utilizando as bibliotecas OpenCV e NumPy para captura e processamento das imagens. O algoritmo foi responsável por capturar continuamente frames da peça sendo impressa. Para a análise, as imagens coloridas em formato RGB foram convertidas para escala de cinza, formando uma matriz cujos pixels foram representados por valores reais de 0 até 1 (Figura 2). Para simplificar a análise, a imagem em escala de cinza foi convertida para uma versão em preto-e-branco usando um algoritmo simples de thresholding.

Figura 2 – Conversão.



Fonte: Elaborada pelo próprio autor.

3.1 PROCESSAMENTO DE IMAGENS E ANÁLISE DIMENSIONAL

O algoritmo desenvolvido foi projetado para realizar duas análises principais: dimensional e de retilineidade. Para a análise dimensional, a imagem binarizada foi utilizada para contar os pixels brancos ou pretos em uma linha ou coluna específica da região de interesse (RI). Essa contagem foi convertida em medidas reais (milímetros) com base em uma referência conhecida,

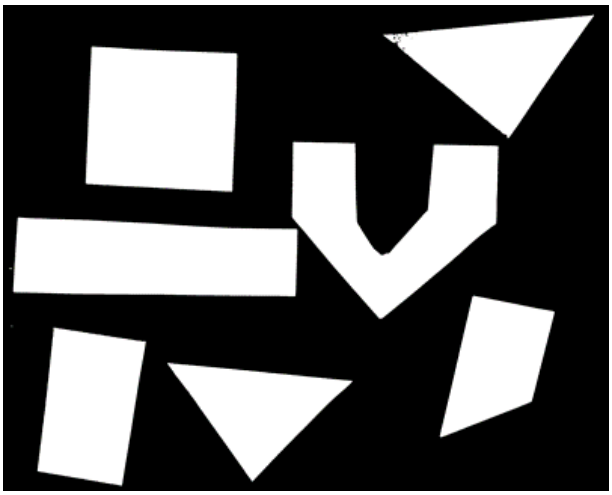
como um objeto de dimensões calibradas presente na imagem. A relação pixels/mm foi calculada e aplicada para estimar as dimensões da peça, como altura, largura e espessura.

3.2 VERIFICAÇÃO DE RETILINEIDADE

Além da análise dimensional, o algoritmo também foi capaz de verificar a retilineidade das peças. Para isso, a RI foi processada para detectar bordas utilizando o algoritmo de Canny, que identifica transições bruscas de intensidade na imagem (RONG et al., 2014). Em seguida, a detecção de linhas foi realizada utilizando a Transformada de Hough Probabilística, que identifica segmentos de linha com base em parâmetros como comprimento mínimo e distância máxima entre pontos. O número de linhas detectadas e sua orientação foram analisados para verificar se a peça atendia aos critérios de retilineidade. Linhas desalinhadas ou ausência de linhas indicaram possíveis defeitos na fabricação, como deformações ou irregularidades (KIRYATI; ELDAR; BRUCKSTEIN, 1991).

Com isso, é gerado um histograma do nível de cinza da imagem, em que os valores vão de 0 a 1. A partir da análise desse histograma, define-se o valor do limiar que separa os pixels da imagem, dando possibilidade à obtenção da imagem em versão preto-e-branco. Essa imagem é ilustrada na Figura 3.

Figura 3 – Captura em preto-e-branco.

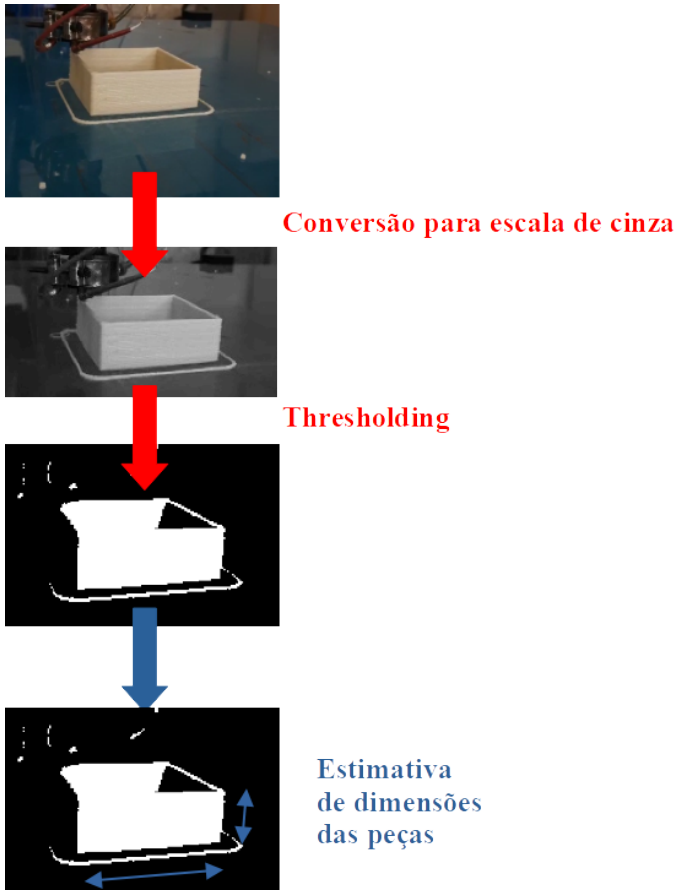


Fonte: Elaborada pelo próprio autor.

A partir dessa representação, as dimensões da peça são estimadas em pixels, sendo que essa unidade foi calibrada com auxílio de medidas de referência na imagem capturada de modo a conhecer a relação pixel/mm. Uma dificuldade do uso dessa metodologia é a definição do valor mais adequado para a limiarização. O sucesso dela depende do quão bem definidas estão as massas de pixels no histograma da imagem capturada. Outra dificuldade é que essa técnica é afetada quando há incidência de luz em uma determinada região de análise da peça e, desse modo, deve-se adotar uma iluminação que não afete a sensibilidade dela.

Em suma, a Figura 4 ilustra de forma resumida essa metodologia de processamento das imagens.

Figura 4 – Resumo da metodologia de processamento de imagens.



Fonte: Elaborada pelo próprio autor.

Para validar o algoritmo de medição, as dimensões estimadas a partir do processamento de imagens foram confrontadas com as obtidas a partir de metrologia dimensional feita com um paquímetro simples. Além disso, a retilineidade foi avaliada visualmente e comparada com os resultados gerados pelo algoritmo.

Para os testes da metodologia, a maioria dos parâmetros foram fixados e configurados para processamento de pellets de polímero PLA. Foi fabricada uma peça oca com dimensões 100 x 100 x 50 milímetros com as configurações resumidas na Tabela 1.

A velocidade de impressão a ser configurada no fatiador foi fixada em 10 mm/s, assim como a velocidade do manipulador robótico. As diferenças de temperaturas nas zonas de aquecimento foram usadas para simular defeitos dimensionais induzidos nas peças impressas.

Todos os parâmetros foram definidos seguindo o estudo de Pulquerio, Barbosa e Shiki (2024), que explora a integração de um extrusor de parafuso único a um braço robótico antropomórfico no contexto da manufatura aditiva robotizada. O estudo apresenta uma análise de regressão para estabelecer a relação entre os parâmetros do processo e a geometria da camada

Tabela 1 – Resumo de parâmetros a serem utilizados para impressão.

Parâmetro	**Valor**
Material	PLA
Temperaturas nas zonas de aquecimento (distintas)	T = 190°C (zona 1), T = 180°C (zona 2), T = 175°C (zona 3)
Rotação da rosa	30 RPM
Velocidade do robô	10 mm/s
Velocidade de impressão (configurada no fatiador)	10 mm/s
Altura das camadas	1 mm

Fonte: Elaborada pelo próprio autor.

depositada, permitindo assim otimizar a impressão de objetos 3D de grande escala em polímero PLA.

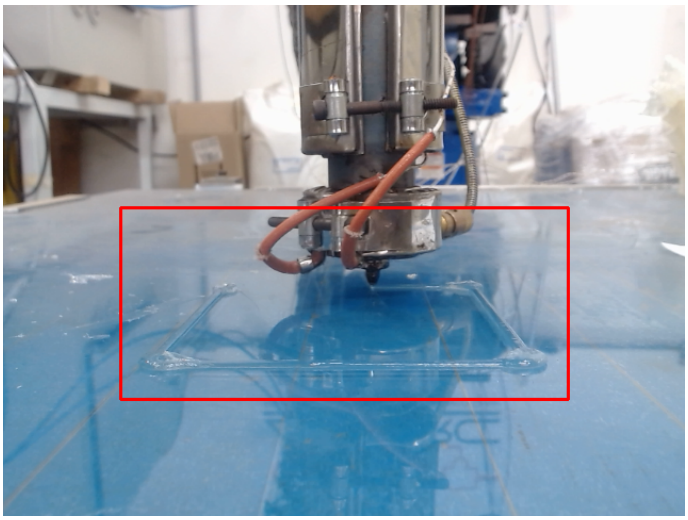
Por fim, é importante frisar que a análise dimensional e de retilineidade foi feita a partir de um vídeo de todo o processo de impressão da peça. Ao fim dos ensaios, o método demonstrou eficiência para monitorar e detectar defeitos na impressão 3D, tanto em termos de dimensões quanto de características geométricas, como a retilineidade.

3.3 DIMENSIONAMENTO E ANÁLISE COMPARATIVA

O dimensionamento das peças foi realizado com base na análise de imagens capturadas durante o processo de fabricação. O algoritmo processou as imagens para extrair as dimensões das peças em milímetros, utilizando uma referência conhecida (5 mm) para calibrar a relação entre pixels e milímetros.

O algoritmo processou as imagens capturadas pela câmera Raspberry Pi, aplicando técnicas de pré-processamento, como conversão para escala de cinza e binarização. A região de interesse (RI) foi definida com base nas dimensões da imagem e em parâmetros pré-configurados, como largura, altura e deslocamento. A seguir, a Figura 5 ilustra a RI destacada na imagem original.

Figura 5 – Região de interesse para análise dimensional.



Fonte: Elaborada pelo próprio autor.

Após a definição da RI, a imagem foi binarizada para facilitar a análise dimensional. A binarização foi realizada utilizando um limiar dinâmico, que separa os pixels em duas categorias: pretos (intensidade 0) e brancos (intensidade 255). A Figura 6 mostra um exemplo de imagem binarizada da RI do sistema, onde os pixels pretos representam a peça e os brancos representam o fundo.

A contagem de pixels brancos na coluna central da RI foi utilizada para calcular as dimensões da peça. A relação pixels/mm foi aplicada para converter a contagem de pixels em medidas reais. A Tabela 2 apresenta as dimensões calculadas para uma amostra de imagens de uma mesma peça sendo impressa. Cada imagem reflete um período da impressão, sendo que a Imagem 1 refere-se ao início e a 6 ao final do processo.

Para validar os resultados do algoritmo, as dimensões calculadas foram comparadas com medidas reais obtidas com um paquímetro. A Figura 7 mostra a comparação entre as dimensões calculadas e as medidas reais para uma amostra de peças.

Figura 6 – Imagem binarizada.

Fonte: Elaborada pelo próprio autor.

Tabela 2 – Resumo do dimensionamento.

Imagem	**Altura [mm]**	**Largura [mm]**
Imagem 1	0,56	107,89
Imagem 2	8,33	103,33
Imagem 3	17,22	102,57
Imagem 4	25,0	102,33
Imagem 5	34,44	102,78
Imagem 6	50,0	101,67

Fonte: Elaborada pelo próprio autor.

A análise comparativa demonstrou que o algoritmo foi capaz de estimar as dimensões de altura da peça com precisão, apresentando um erro de precisão médio de ±0,4 mm. Já com relação ao dimensionamento da largura, um erro de precisão foi observado, resultando em um desvio de ±3,4 mm. Esse erro se deve, principalmente, à primeira fase da impressão, em que a camada inferior é mais larga para garantir que a peça fique firme na mesa. Esse comportamento é comum em processos de manufatura aditiva, onde a aderência à base de impressão é crítica para a qualidade final da peça.

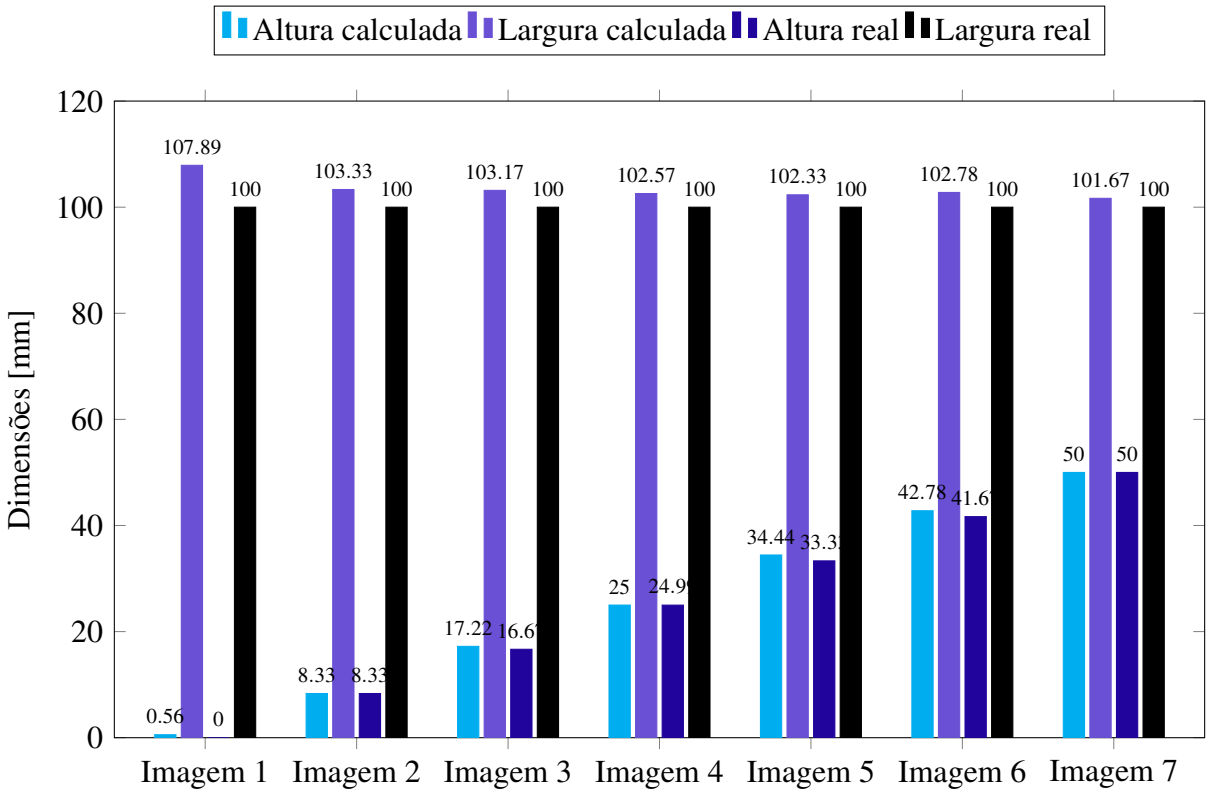
—

3.4 ANÁLISE DE RETILINEIDADE

A análise de retilineidade das peças revelou resultados significativos. O algoritmo foi capaz de identificar um número variável de linhas retas em cada imagem, o que pode indicar a precisão geométrica das peças analisadas. A Tabela 3 apresenta os números de linhas detectadas em cada imagem, destacando a capacidade do algoritmo de detectar defeitos geométricos.

Para a peça completa, foram impressas 30 camadas de filamento, e, portanto, esperava-se a detecção de 30 linhas retas para cada lado da peça. No entanto, os resultados revelaram que, na maioria das imagens analisadas, o número de linhas retas detectadas foi abaixo do esperado. Este erro de precisão pode ser explicado por flutuações de processo, como variações de temperatura ou inconsistências na deposição do material.

Figura 7 – Comparativo entre dimensões calculadas e medidas reais.



Fonte: Elaborada pelo próprio autor.

Tabela 3 – Resultados da análise de retilineidade.

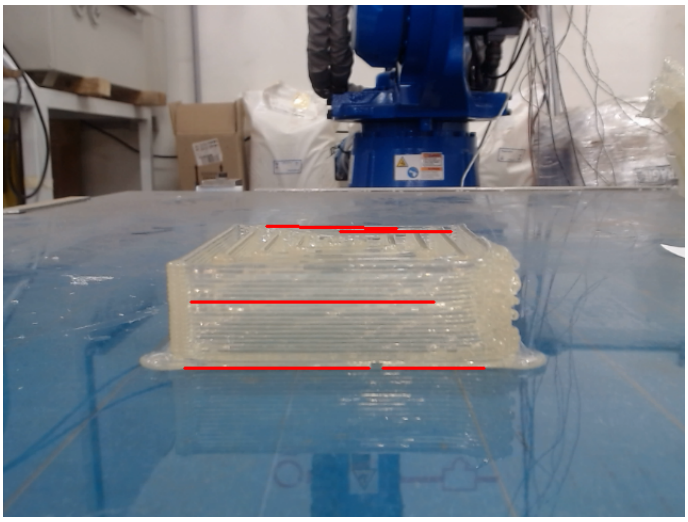
Imagem	**Linhas retas detectadas**
Imagem 1	0
Imagem 2	1
Imagem 3	2
Imagem 4	4
Imagem 5	6
Imagem 6	5

Fonte: Elaborada pelo próprio autor.

A Figura 8 ilustra a aplicação da técnica de análise de retilineidade em uma das peças, demonstrando visualmente as irregularidades detectadas. A variação nos resultados pode ser atribuída a flutuações no processo de impressão, como o erro de precisão devido a variações de temperatura ou inconsistências na deposição do material.

Em conclusão, os resultados indicam que a peça impressa não atingiu os padrões esperados de retilineidade. A análise mostrou uma detecção inferior às 30 linhas retas esperadas para cada lado da peça, o que sugere problemas no processo de impressão. A melhoria na consistência do processo de impressão, incluindo controle mais rigoroso da temperatura e da deposição do material, pode ser crucial para alcançar um desempenho mais preciso e atender às especificações

Figura 8 – Retilineidade final.



Fonte: Elaborada pelo próprio autor.

de projeto, especialmente em aplicações onde a precisão geométrica é crítica.

4 CONCLUSÃO

Com base nos resultados apresentados neste trabalho, conclui-se que o algoritmo desenvolvido demonstrou eficiência e precisão na análise dimensional e verificação de retilineidade de peças fabricadas por manufatura aditiva robotizada. A combinação de técnicas avançadas de processamento de imagens e métodos de análise geométrica permitiu a extração de informações detalhadas e precisas sobre as dimensões das peças, evidenciando a capacidade de avaliar características críticas, tais como a retilineidade, que são essenciais para garantir a qualidade do produto final. A flexibilidade do algoritmo permitiu adaptações para diferentes condições e necessidades, reforçando seu potencial como uma ferramenta eficaz de controle de qualidade em ambientes de manufatura variáveis. Essa capacidade de personalização viabiliza seu uso em uma ampla gama de cenários, promovendo melhorias contínuas nos processos de fabricação e contribuindo para a otimização dos custos e aumento da eficiência na produção de peças complexas.

No que tange à análise dimensional, o algoritmo demonstrou alta precisão, especialmente no cálculo das alturas das peças, apresentando um erro médio relativamente baixo. No entanto, a avaliação da largura das peças revelou desafios devido a características intrínsecas do processo de impressão, que precisam ser consideradas para melhorias futuras. A capacidade do algoritmo em identificar e sinalizar essas discrepâncias fornece insights valiosos para aprimorar práticas de manufatura aditiva. A análise de retilineidade evidenciou a competência do algoritmo em detectar desalinhamentos e irregularidades potencialmente comprometedores das especificações de projeto. A aplicação da Transformada de Hough Probabilística mostrou-se extremamente útil para reconhecer defeitos geométricos de forma eficaz. Contudo, variações e flutuações no processo de impressão devem ser consideradas, uma vez que podem impactar a precisão da retilineidade, sugerindo áreas para ajustes e otimização nos processos de calibragem e controle térmico.

Apesar das conquistas e capacidades demonstradas, é importante reconhecer as limitações do projeto atual. Destaca-se que para peças com geometrias complexas, o algoritmo enfrenta dificuldades em obter um dimensionamento preciso. Isso se deve à complexidade das formas que podem ultrapassar as capacidades das técnicas de análise empregadas, exigindo abordagens mais sofisticadas ou a integração de tecnologias suplementares para garantir a precisão necessária. Essa limitação aponta para a necessidade de desenvolvimento contínuo e investigação de novas soluções que possam expandir o escopo de aplicação do algoritmo e aumentar sua eficácia na análise de uma variedade mais ampla de geometrias.

Em síntese, o trabalho demonstrou que o algoritmo é uma solução viável e promissora para o monitoramento automatizado de qualidade em processos de manufatura aditiva. Recomenda-se, no entanto, a continuidade no aprimoramento das técnicas de calibração, bem como a consideração das condições de impressão, visando alcançar resultados cada vez mais

precisos e alinhados com as especificações de design. Futuros desenvolvimentos nesses aspectos não apenas endereçarão as limitações citadas, mas também avançarão ainda mais as capacidades de controle e otimização de processos, contribuindo significativamente para a evolução das práticas de fabricação aditiva.

REFERÊNCIAS

CHARALAMPOUS, P. e. a. Vision-based real-time monitoring of extrusion additive manufacturing processes for automatic manufacturing error detection. **The International Journal of Advanced Manufacturing Technology**, v. 115, n. 11, p. 3859–3872, 2021. Citado na página 18.

JIN, Z.; ZHANG, Z.; GU, G. X. Autonomous in-situ correction of fused deposition modeling printers using computer vision and deep learning. **Manufacturing Letters**, v. 22, p. 11–15, 2019. Citado na página 18.

KARTHIKEYAN, S. e. a. Computer vision-based surface roughness measurement using artificial neural network. **Materials Today: Proceedings**, v. 60, p. 1325–1328, 2022. Citado na página 19.

KIRYATI, N.; ELDAR, Y.; BRUCKSTEIN, A. M. A probabilistic hough transform. **Pattern Recognition**, Elsevier, v. 24, n. 4, p. 303–316, 1991. Citado na página 22.

KURADA, S.; BRADLEY, C. A machine vision system for tool wear assessment. **Tribology International**, v. 30, n. 4, p. 295–304, 1997. Citado na página 19.

LANGELAND, S. A. K. **Automatic Error Detection in 3D Printing using Computer Vision**. Dissertação de Mestrado, 2020. Citado na página 17.

MAHESH, B. Machine learning algorithms-a review. **International Journal of Science and Research (IJSR)**, v. 9, p. 381–386, 2020. Citado 2 vezes nas páginas 13 e 17.

MORETTI, M.; ROSSI, A.; SENIN, N. In-process monitoring of part geometry in fused filament fabrication using computer vision and digital twins. **Additive Manufacturing**, v. 37, p. 101609, 2021. Citado 2 vezes nas páginas 14 e 17.

PARASKEVOUDIS, K.; KARAYANNIS, P.; KOUMOULOS, E. P. Real-time 3d printing remote defect detection (stringing) with computer vision and artificial intelligence. **Processes**, v. 8, n. 11, p. 1464, 2020. Citado 2 vezes nas páginas 14 e 18.

PETSIK, A. L.; PEARCE, J. M. Open source computer vision-based layer-wise 3d printing analysis. **Additive Manufacturing**, v. 36, p. 101473, 2020. Citado na página 18.

PULQUERIO, E. C.; BARBOSA, G. F.; SHIKI, S. B. 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**, Springer, p. 1–12, 2024. Citado na página 23.

RONG, W. et al. An improved canny edge detection algorithm. In: **2014 IEEE International Conference on Mechatronics and Automation**. [S.l.]: IEEE, 2014. p. 577–582. Citado na página 22.

SHAH, J. e. a. 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. Citado na página 16.

VOLPATO, N. **Manufatura aditiva: tecnologias e aplicações da impressão 3D**. Brasil: Editora Blucher, 2017. Citado 2 vezes nas páginas 13 e 15.

WANG, Z. e. a. 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. Citado 2 vezes nas páginas 13 e 16.

WONG, K. V.; HERNANDEZ, A. A review of additive manufacturing. **International Scholarly Research Notices**, v. 2012, 2012. Citado na página 15.

XU, J. e. a. An apf-aco algorithm for automatic defect detection on vehicle paint. **Multimedia Tools and Applications**, v. 79, n. 35, p. 25315–25333, 2020. Citado na página 18.

ZUTIN, G. et al. Application of robotic manipulator technology and its relation to additive manufacturing process — a review. **International Journal of Advanced Manufacturing Technology**, v. 133, p. 257–271, 2024. Citado na página 16.

APÊNDICE A – ALGORITMO PARA PROCESSAMENTO E VISUALIZAÇÃO DO PROCESSO

```
import cv2
import numpy as np
import os
import re

referencia_mm = 5 # Referncia em milmetros

def identificar_pixels_extrusora(caminho_imagem):
    """
    Processa uma nica imagem e conta os pixels brancos na linha central da ROI.
    """
    # Carregar a imagem
    imagem = cv2.imread(caminho_imagem)
    if imagem is None:
        print(f"Erro ao carregar imagem: {caminho_imagem}")
        return None

    # Definir parmetros para a ROI
    altura, largura, _ = imagem.shape
    tamanho_quadrado = 40
    deslocamento_x = 20
    deslocamento_y = 20
    x_centro = largura // 2 + deslocamento_x
    y_centro = altura // 2 + deslocamento_y

    x_inicio = x_centro - tamanho_quadrado // 2
    y_inicio = y_centro - tamanho_quadrado // 2
    x_fim = x_centro + tamanho_quadrado // 2
    y_fim = y_centro + tamanho_quadrado // 2

    # Obter a regio de interesse (ROI)
    roi = imagem[y_inicio:y_fim, x_inicio:x_fim]
    cinza = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    _, binarizada = cv2.threshold(cinza, 40, 255, cv2.THRESH_BINARY_INV)
```

```
    # Analisar os pixels na linha um tero do topo da ROI
    linha_y = binarizada.shape[0] // 3
    linha = binarizada[linha_y, :]
    pixels_brancos = cv2.countNonZero(linha)
    print(f"Pixels␣brancos␣na␣linha:␣{pixels_brancos}")

    # Desenhar a linha na imagem binarizada
    cv2.line(binarizada, (0, linha_y), (binarizada.shape[1], linha_y), 127, 1)

    return pixels_brancos

def processar_varias_imagens(diretorio_imagens, pixels_mm, limiares):
    arquivos = [f for f in os.listdir(diretorio_imagens) if f.startswith('original_image_')]
    arquivos_ordenados = sorted(arquivos, key=lambda x: int(re.search(r'\d+', x).group()))
    imagens_para_salvar = {0, 14, 29, 44, 59, 74, 90} # ndices ajustados para 0-indexado

    for i, arquivo in enumerate(arquivos_ordenados):
        caminho_imagem = os.path.join(diretorio_imagens, arquivo)
        imagem = cv2.imread(caminho_imagem)

        if imagem is None:
            print(f"Erro␣ao␣carregar␣imagem:␣{caminho_imagem}")
            continue

        limiar_atual = limiares[i] if i < len(limiares) else 128
        imagem_com_retangulo = imagem.copy()

        altura, largura, _ = imagem.shape
        largura_retangulo = 420
        altura_retangulo = 180
        deslocamento_x = 0
        deslocamento_y = 40
        x_centro = largura // 2 + deslocamento_x
        y_centro = altura // 2 + deslocamento_y

        x_inicio = x_centro - largura_retangulo // 2
        y_inicio = y_centro - altura_retangulo // 2
        x_fim = x_centro + largura_retangulo // 2
        y_fim = y_centro + altura_retangulo // 2
```

```
        if (x_inicio < 0 or y_inicio < 0 or x_fim > largura or y_fim > altura):
            print(f"Erro: Retngulo fora dos limites para a imagem:
                {caminho_imagem}")
            continue

        cv2.rectangle(imagem_com_retangulo, (x_inicio, y_inicio), (x_fim, y_fim), (200, 0,
            0), 2)

        roi = imagem[y_inicio:y_fim, x_inicio:x_fim]
        roi_cinza = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        roi_cinza = cv2.GaussianBlur(roi_cinza, (5, 5), 0)

        _, roi_binaria = cv2.threshold(roi_cinza, limiar_atual, 255,
            cv2.THRESH_BINARY_INV)

        coluna_x = roi_binaria.shape[1] // 2
        coluna = roi_binaria[:, coluna_x]

        total_pixels = coluna.shape[0]
        pixels_brancos = cv2.countNonZero(coluna)
        pixels_pretos = total_pixels - pixels_brancos

        print(f"Imagem: {arquivo} - Pixels pretos na coluna central dentro do
            retngulo: {pixels_pretos}")
        altura_mm = pixels_mm * pixels_pretos
        print(f"Altura calculada: {altura_mm:.2f} mm")

        if i in imagens_para_salvar:
            save_path_color = os.path.join(diretorio_imagens, f"processed_{arquivo}")
            save_path_bin = os.path.join(diretorio_imagens, f"binary_{arquivo}")
            cv2.imwrite(save_path_color, imagem_com_retangulo)
            cv2.imwrite(save_path_bin, roi_binaria)

            print(f"Imagem com retngulo salva em: {save_path_color}")
            print(f"Imagem binarizada salva em: {save_path_bin}")

# Caminhos para execuo
caminho_imagem_unica = '/home/thiagosilva/Desktop/TCC/pictures/original_image_1.png'
```

```
diretorio_imagens = '/home/thiagosilva/Desktop/TCC/asa'

# Executar funo para uma nica imagem e obter pixels_brancos
pixels_brancos = identificar_pixels_extrusora(caminho_imagem_unica)
if pixels_brancos is not None:
    pixels_mm = referencia_mm / pixels_brancos
    print(f"Relao pixels/mm: {pixels_mm:.4f}")

    # Defina os limiares desejados para cada imagem
    limiares = [155,185,155,169,195,140,130,160,195,133,165,155,160,155,155,
                100,153,151,160,145,164,131,175,165,183,152,167,137,150,146,
                154,165,155,164,135,167,147,167,164,160,155,174,160,132,135,
                165,151,164,161,128,172,166,171,158,167,140,168,159,150,154,
                148,134,146,139,158,162,160,164,138,165,149,171,166,159,148,
                169,148,139,144,155,144,154,159,140,161,154,156,150,153,151,
                147
    ]

    processar_varias_imagens(diretorio_imagens, pixels_mm, limiares)
else:
    print("Erro ao processar a imagem nica. Verifique o caminho da imagem.")


def verificar_retilineidade_na_roi(imagem, x_inicio, y_inicio, x_fim, y_fim):
    """
    Verifica a retilineidade dentro de uma ROI especfica da imagem.
    """
    # Extrair a ROI
    roi = imagem[y_inicio:y_fim, x_inicio:x_fim]

    # Converter a ROI para escala de cinza
    roi_cinza = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

    # Detectar bordas
    bordas = cv2.Canny(roi_cinza, 50, 150)

    # Detectar linhas com a Transformada de Hough
    linhas = cv2.HoughLinesP(bordas, 1, np.pi / 180, threshold=100, minLineLength=50,
        maxLineGap=10)
```

```
    # Desenhar as linhas detectadas na ROI
    if linhas is not None:
        for linha in linhas:
            x1, y1, x2, y2 = linha[0]
            cv2.line(roi, (x1, y1), (x2, y2), (0, 255, 0), 2)
        print(f"Linhas␣detectadas␣na␣ROI:␣{len(linhas)}")
    else:
        print("Nenhuma␣linha␣detectada␣na␣ROI.")

    # # Exibir a ROI com as linhas detectadas
    # cv2.imshow('Retilineidade na ROI', roi)
    # cv2.waitKey(0)
    # cv2.destroyAllWindows()


def verificar_retilineidade_na_roi(imagem, x_inicio, y_inicio, x_fim, y_fim):
    roi = imagem[y_inicio:y_fim, x_inicio:x_fim]
    roi_cinza = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    bordas = cv2.Canny(roi_cinza, 50, 150)
    linhas = cv2.HoughLinesP(bordas, 1, np.pi / 180, threshold=100, minLineLength=50,
        maxLineGap=10)

    if linhas is not None:
        for linha in linhas:
            x1, y1, x2, y2 = linha[0]
            cv2.line(roi, (x1, y1), (x2, y2), (0, 255, 0), 2)
        print(f"Linhas␣detectadas␣na␣ROI:␣{len(linhas)}")
    else:
        print("Nenhuma␣linha␣detectada␣na␣ROI.")

def processar_varias_imagens_com_retilineidade(diretorio_imagens, pixels_mm, limiares):
    arquivos = [f for f in os.listdir(diretorio_imagens) if f.startswith('original_image_')]
    arquivos_ordenados = sorted(arquivos, key=lambda x: int(re.search(r'\d+', x).group()))

    imagens_para_salvar = {0, 14, 29, 44, 59, 74, 90} # ndices ajustados para 0-indexado

    for i, arquivo in enumerate(arquivos_ordenados):
        caminho_imagem = os.path.join(diretorio_imagens, arquivo)
```

```
        imagem = cv2.imread(caminho_imagem)
        if imagem is None:
            print(f"Erro ao carregar imagem: {caminho_imagem}")
            continue

        altura, largura, _ = imagem.shape
        largura_retangulo = 420
        altura_retangulo = 180
        deslocamento_x = 0
        deslocamento_y = 40
        x_centro = largura // 2 + deslocamento_x
        y_centro = altura // 2 + deslocamento_y

        x_inicio = x_centro - largura_retangulo // 2
        y_inicio = y_centro - altura_retangulo // 2
        x_fim = x_centro + largura_retangulo // 2
        y_fim = y_centro + altura_retangulo // 2

        # Verificar retilineidade
        verificar_retilineidade_na_roi(imagem, x_inicio, y_inicio, x_fim, y_fim)

        # Aplicar limiarizao na ROI
        roi = imagem[y_inicio:y_fim, x_inicio:x_fim]
        roi_cinza = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        limiar_atual = limiares[i] if i < len(limiares) else 128
        _, roi_binaria = cv2.threshold(roi_cinza, limiar_atual, 255, cv2.THRESH_BINARY)

        # Salvar as imagens processadas se estiver nos ndices especificados
        if i in imagens_para_salvar:
            save_path_color = os.path.join(diretorio_imagens, f"retilineidade_{arquivo}")
            save_path_bin = os.path.join(diretorio_imagens,
                f"retilineidade_binary_{arquivo}")
            cv2.imwrite(save_path_color, imagem)
            cv2.imwrite(save_path_bin, roi_binaria)
            print(f"Imagem com retilineidade salva em: {save_path_color}")
            print(f"Imagem binarizada para retilineidade salva em: {save_path_bin}")

# Executar a funo com a lgica de limiares adaptada
processar_varias_imagens(diretorio_imagens, pixels_mm, limiares)
```

```
processar_varias_imagens_com_retilineidade(diretorio_imagens, pixels_mm, limiares)
```

APÊNDICE B – ALGORITMO PARA PROCESSAMENTO E GERAÇÃO DE TABELA DE RESULTADOS

```
import cv2
import numpy as np
import os
import re
import csv

referencia_mm = 5 # Referncia em milmetros

def identificar_pixels_extrusora(caminho_imagem):
    """
    Processa uma nica imagem e conta os pixels brancos na linha central da ROI.
    """
    # Carregar a imagem
    imagem = cv2.imread(caminho_imagem)
    if imagem is None:
        print(f"Erro ao carregar imagem: {caminho_imagem}")
        return None

    # Definir parmetros para a ROI
    altura, largura, _ = imagem.shape
    tamanho_quadrado = 40
    deslocamento_x = 20
    deslocamento_y = 20
    x_centro = largura // 2 + deslocamento_x
    y_centro = altura // 2 + deslocamento_y

    x_inicio = x_centro - tamanho_quadrado // 2
    y_inicio = y_centro - tamanho_quadrado // 2
    x_fim = x_centro + tamanho_quadrado // 2
    y_fim = y_centro + tamanho_quadrado // 2

    # Obter a regio de interesse (ROI)
    roi = imagem[y_inicio:y_fim, x_inicio:x_fim]
    cinza = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    _, binarizada = cv2.threshold(cinza, 40, 255, cv2.THRESH_BINARY_INV)
```

```
    # Analisar os pixels na linha um tero do topo da ROI
    linha_y = binarizada.shape[0] // 3
    linha = binarizada[linha_y, :]
    pixels_brancos = cv2.countNonZero(linha)
    print(f"Pixels␣brancos␣na␣linha:␣{pixels_brancos}")

    # Desenhar a linha na imagem binarizada
    cv2.line(binarizada, (0, linha_y), (binarizada.shape[1], linha_y), 127, 1)

    return pixels_brancos

def processar_varias_imagens_tabela(diretorio_imagens, pixels_mm, limiares, arquivo_saida):
    """
    Processa todas as imagens em um diretrio especfico e gera uma tabela com:
    nome da imagem, altura, largura, linhas detectadas na ROI.
    """
    import os
    import re
    import cv2
    import numpy as np
    import csv

    arquivos = [f for f in os.listdir(diretorio_imagens) if f.startswith('original_image_')]
    arquivos_ordenados = sorted(arquivos, key=lambda x: int(re.search(r'\d+', x).group()))
    dados_tabela = []

    for i, arquivo in enumerate(arquivos_ordenados):
        caminho_imagem = os.path.join(diretorio_imagens, arquivo)
        imagem = cv2.imread(caminho_imagem)

        if imagem is None:
            continue

        limiar_atual = limiares[i] if i < len(limiares) else 128
        altura, largura, _ = imagem.shape
        largura_retangulo = 420
        altura_retangulo = 180
        deslocamento_x = 0
```

```
deslocamento_y = 40
x_centro = largura // 2 + deslocamento_x
y_centro = altura // 2 + deslocamento_y
x_inicio = x_centro − largura_retangulo // 2
y_inicio = y_centro − altura_retangulo // 2
x_fim = x_centro + largura_retangulo // 2
y_fim = y_centro + altura_retangulo // 2

if (x_inicio < 0 or y_inicio < 0 or x_fim > largura or y_fim > altura):
    continue

roi = imagem[y_inicio:y_fim, x_inicio:x_fim]
roi_cinza = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
roi_cinza = cv2.GaussianBlur(roi_cinza, (5, 5), 0)
_, roi_binaria = cv2.threshold(roi_cinza, limiar_atual, 255,
    cv2.THRESH_BINARY_INV)
coluna_x = roi_binaria.shape[1] // 2
coluna = roi_binaria[:, coluna_x]
pixels_pretos_altura = coluna.shape[0] − cv2.countNonZero(coluna)
altura_mm = pixels_mm ∗ pixels_pretos_altura
linha_y = roi_binaria.shape[0] // 2 + 35
linha = roi_binaria[linha_y, :]
pixels_pretos_largura = linha.shape[0] − cv2.countNonZero(linha)
largura_mm = pixels_mm ∗ pixels_pretos_largura
bordas = cv2.Canny(roi_cinza, 50, 150)
linhas = cv2.HoughLinesP(bordas, 1, np.pi / 180, threshold=100,
    minLineLength=50, maxLineGap=10)
num_linhas = len(linhas) if linhas is not None else 0

# Desenhar a linha ‘linha_y‘ na ROI binria para visualizao
cv2.line(roi_binaria, (0, linha_y), (roi_binaria.shape[1] − 1, linha_y), (255, 0, 0), 2)

# Mostrar imagem com a linha em ‘linha_y‘
cv2.imshow('Imagem␣com␣Linha', roi_binaria)
cv2.waitKey(0)
cv2.destroyAllWindows()

dados_tabela.append({
    "Nome␣da␣Imagem": arquivo,
```

```
            "Altura␣(mm)": altura_mm,
            "Largura␣(mm)": largura_mm,
            "Linhas␣Detectadas": num_linhas
        })

    with open(arquivo_saida, mode='w', newline='') as arquivo_csv:
        campos = ["Nome␣da␣Imagem", "Altura␣(mm)", "Largura␣(mm)", "Linhas␣
            Detectadas"]
        escritor = csv.DictWriter(arquivo_csv, fieldnames=campos)
        escritor.writeheader()
        for linha in dados_tabela:
            escritor.writerow(linha)

# Caminhos para execuo
caminho_imagem_unica = '/home/thiagosilva/Desktop/TCC/pictures/original_image_1.png'
diretorio_imagens = '/home/thiagosilva/Desktop/TCC/asa'
arquivo_saida = '/home/thiagosilva/Desktop/TCC/tabela_resultados.csv'

# Executar funo para uma nica imagem e obter pixels_brancos
pixels_brancos = identificar_pixels_extrusora(caminho_imagem_unica)
if pixels_brancos is not None:
    pixels_mm = referencia_mm / pixels_brancos

    # Defina os limiares desejados para cada imagem
    limiares = [155,185,155,169,195,140,130,160,195,133,165,155,160,155,155,
                100,153,151,160,145,164,131,175,165,183,152,167,137,150,146,
                154,165,155,164,135,167,147,167,164,160,155,174,160,132,135,
                165,151,164,161,128,172,166,171,158,167,140,168,159,150,154,
                148,134,146,139,158,162,160,164,138,165,149,171,166,159,148,
                169,148,139,144,155,144,154,159,140,161,154,156,150,153,151,
                147
    ]

    # Processar imagens e gerar tabela
    processar_varias_imagens_tabela(diretorio_imagens, pixels_mm, limiares, arquivo_saida)
```