UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Otávio Mignoso e Silva

Estudo da variação de cor em fitas adesivas holográficas e estimativa do ângulo de observação por meio de redes neurais convolucionais

São Carlos

2025

Otávio Mignoso e Silva

Estudo da variação de cor em fitas adesivas holográficas e estimativa do ângulo de observação por meio de redes neurais convolucionais

Monografia apresentada ao Curso de Engenharia Mecânica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Mecânico.

Orientador: Prof. Dr. Rodrigo Nicoletti

São Carlos

2025

Ficha catalográfica elaborada pela Biblioteca Prof. Sérgio Rodrigues Fontes
e pelo Serviço de Comunicação e Marketing da EESC-USP,
com dados inseridos pelo(a) autor(a).

S586e

Silva, Otávio Mignoso e

Estudo da variação de cor em fitas adesivas holográficas e estimativa do ângulo de observação por meio de redes neurais convolucionais / Otávio Mignoso e Silva ; orientador Rodrigo Nicoletti. -- São Carlos, 2025.
63 p.

Monografia - Graduação em Engenharia Mecânica -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2025.

1. Fita adesiva holográfica. 2. Variação angular. 3. Processamento de imagens. 4. Redes Neurais Convolucionais. I. Nicoletti, Rodrigo, orient. II. Título.

Responsáveis pela estrutura de catalogação da publicação segundo a AACR2: Bibliotecários da EESC/USP.

FOLHA DE AVALIAÇÃO

Candidato: Otávio Mignoso e Silva

Título: Estudo da variação de cor em fitas adesivas holográficas e estimativa do ângulo de observação por meio de redes neurais convolucionais

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo Curso de Engenharia Mecânica

BANCA EXAMINADORA

Professor Dr. Rodrigo Nicoletti
(orientador)

Nota atribuída: 10,0 (dez)

Rodrigo Nicoletti
(assinatura)

Professor Dr. Leopoldo Pisanelli Rodrigues de Oliveira

Nota atribuída: 10,0 (dez)

(assinatura)

Eng. Ms. Aline de Almeida Soares

Nota atribuída: 10,0 (dez)

(assinatura)

Média: 10,0 (dez)

Resultado: APROVADO

Data: 14/11/2025.

Este trabalho tem condições de ser hospedado no Portal Digital da Biblioteca da EESC

SIM ☒ NÃO ☐ Visto do orientador Rodrigo Nicoletti

Dedico este trabalho à minha família, pelo apoio incondicional, aos amigos, pela parceria durante essa jornada, aos professores, pelos ensinamentos valiosos, e a todas as pessoas que, de alguma forma, ajudaram-me a chegar até aqui.

RESUMO

SILVA, O. **Estudo da variação de cor em fitas adesivas holográficas e estimativa do ângulo de observação por meio de redes neurais convolucionais**. 2025. 63 p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2025.

Este trabalho avaliou a viabilidade da utilização de fitas adesivas holográficas como ferramenta para estimativa do ângulo de superfícies, explorando o fenômeno da iridescência. Para isso, conduziu-se um primeiro experimento para identificar padrões na variação de cor da fita em função do ângulo de observação, através de técnicas de processamento digital de imagens. Em seguida, desenvolveu-se um modelo de rede neural convolucional que foi treinado com imagens da fita observada de diferentes posições, para estimar o ângulo de observação através de novas imagens. Diante disso, demonstrou-se uma forte correlação entre os dados reais e os previstos, com erro absoluto médio de 1.11 ($\pm$ 0.30) e coeficiente de determinação (R^2) de 0.9947, evidenciando a eficácia da abordagem. Portanto, concluiu-se que a fita adesiva holográfica apresenta potencial como sensor de baixo custo para aplicações em engenharia, com a possibilidade de estender seu uso para medições em múltiplos eixos através de aperfeiçoamentos experimentais.

Palavras-chave: Fita adesiva holográfica; Variação angular; Processamento de imagens; Redes Neurais Convolucionais.

ABSTRACT

SILVA, O. **Study of color variation in holographic adhesive tapes and estimation of the observation angle using convolutional neural networks**. 2025. 63 p. Monograph (Conclusion Course Paper) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2025.

This study evaluated the feasibility of using holographic adhesive tapes as a tool for estimating surface angles by exploring the phenomenon of iridescence. Initially, an experiment was conducted to identify patterns in the color variation of the tape as a function of the observation angle, using digital image processing techniques. Subsequently, a convolutional neural network model was developed and trained with images of the tape observed from different positions, enabling the estimation of the observation angle from new images. The results showed a strong correlation between the actual and predicted values, with a mean absolute error of 1.11 ($\pm$ 0.30) and a coefficient of determination (R^2) of 0.9947, demonstrating the effectiveness of the approach. Therefore, it was concluded that holographic adhesive tape has potential as a low-cost sensor for engineering applications, with the possibility of extending its use to multi-axis measurements through further experimental improvements.

Keywords: Holographic adhesive tape; Angular variation; Image processing; Convolutional Neural Networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CNN	Rede Neural Convolucional (*Convolutional Neural Network*)
RGB	*Red-Green-Blue*
DIP	Processamento Digital de Imagens (*Digital Image Processing*)
ReLU	*Rectified Linear Unit*
API	Interface de Programação de Aplicações (*Application Programming Interface*)
MSE	Erro Quadrático Médio (*Mean Squared Error*)
MAE	Erro Absoluto Médio (*Mean Absolute Error*)

LISTA DE SÍMBOLOS

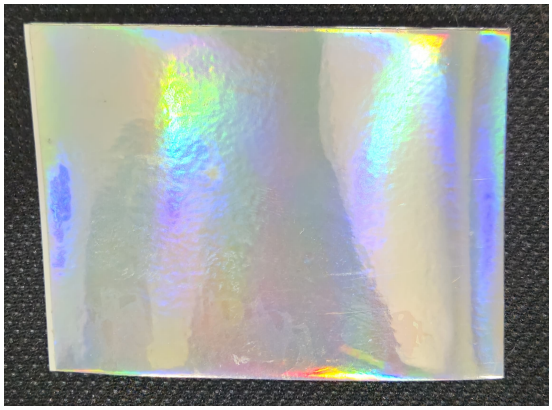
f	Função
R	Componente vermelha do modelo de cor RGB
G	Componente verde do modelo de cor RGB
B	Componente azul do modelo de cor RGB
n	Número de amostras
Y_i	Valor real na amostra i
$\hat{Y}_i$	Valor médio na amostra i
σ	Desvio padrão
R^2	Coeficiente de determinação

SUMÁRIO

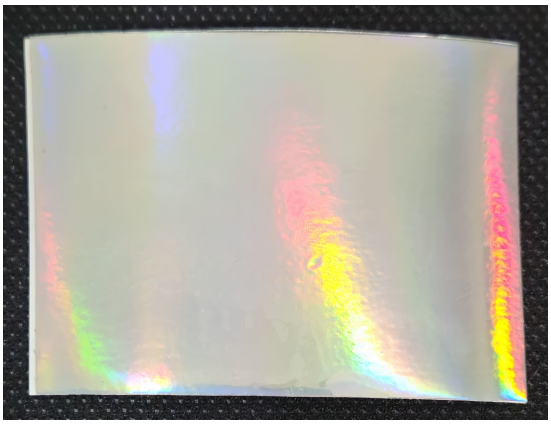
1 INTRODUÇÃO

A fita adesiva holográfica é um item frequentemente utilizado em artesanatos, decorações, artigos de festas e acabamentos em geral. Ela apresenta como principal característica a iridescência, ou seja, a mudança de coloração quando observada a partir de diferentes perspectivas, como explicitado nas Figuras 1a e 1b.

Figura 1 – Fotografias da fita adesiva holográfica em ângulos distintos.



(a) Fotografia da fita adesiva holográfica observada de cima.



(b) Fotografia da fita adesiva holográfica observada de baixo.

Fonte: Elaborada pelo autor.

Ainda que essa caraterística seja majoritariamente aplicada para fins de decoração, essa relação entre o ângulo de observação e a cor da fita possibilita a sua utilização em aplicações tecnológicas. Nesse sentido, como é constatado uma variação com comportamento estável e previsível, essa característica se torna potencialmente explorável como indicador de posição, de maneira semelhante a alguns sistemas de sensoriamento existentes.

De acordo com WANG (2024), o estado da arte já contempla maneiras de realizar a medição de ângulo de maneira visual, baseando-se na diferença de informação presente em uma imagem antes e depois dessa mudança. Além disso, ressalta-se a existência de aplicações que utilizam técnicas de inteligência artificial para esse fim, como modelos baseados em redes neurais convolucionais, do inglês *convolutional neural network* (CNN).

Tendo em vista esse cenário, a fita adesiva holográfica aparece como uma alternativa para investigações que combinam o fenômeno óptico de iridescência com ferramentas de visão computacional e inteligência artificial, viabilizando uma solução inovadora para esse tipo de medição.

1.1 Apresentação do problema

Diante da demanda por sistemas de medições que apresentem alta precisão, amplo intervalo operacional e capacidade de medição em múltiplos graus de liberdade, observa-se um contínuo aperfeiçoamento dos sistemas existentes e a criação de novos sensores baseados em diferentes princípios de funcionamento. Dessa forma, a partir da característica de irisdescência apresentada pela fita adesiva holográfica, levanta-se a possibilidade de explorar essa propriedade óptica como ferramenta em projetos de engenharia para a identificação de ângulos de superfícies através do ângulo de incidência da luz.

1.2 Objetivo

O objetivo deste trabalho é avaliar a viabilidade da medição do ângulo de observação de uma superfície através da variação de cores apresentada em uma fita adesiva holográfica. Para isso, será conduzido um experimento voltado à identificação da presença de padrões na mudança de cores da fita em função da variação do ângulo de observação. Em seguida, será desenvolvida uma CNN, treinada com imagens da fita em diferentes posições, de forma a tornar possível a estimativa desse ângulo.

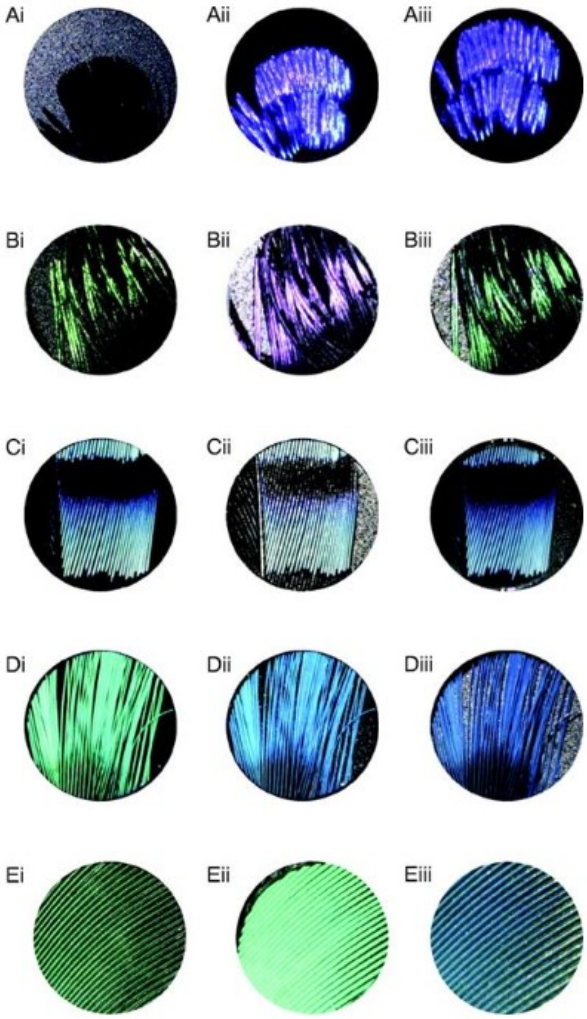
2 DESENVOLVIMENTO

2.1 Revisão bibliográfica

2.1.1 Iridescência

A mudança de coloração que ocorre quando a fita adesiva holográfica é observada a partir de diferentes perspectivas é causada devido ao fenômeno chamado de irisdescência. Um material iridescente é caracterizado pela capacidade de brilhar ou piscar com cores que variam de acordo com a posição em que é vista (SIMPSON; WEINER, 1989). Esse fenômeno pode ser observado tanto em criações artificiais, como em fitas adesivas holográficas e óleos, quanto em materiais naturais, como em plantas e animais, conforme explicitado na Figura 2.

Figura 2 – Fotografias da alteração da cor com a variação da orientação da superfície e iluminação em penas.



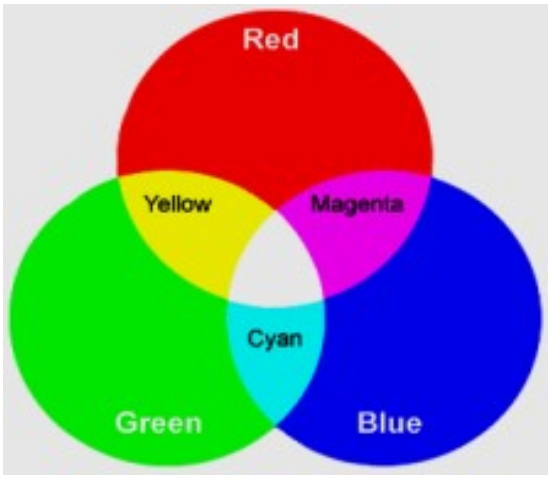
Fonte: OSORIO D.; HAM (2002)

É importante destacar que a cor gerada nesse processo é conhecida como cor estrutural, um tipo de cor que surge quando determinados comprimentos de onda da luz são refletidos de forma específica. Além disso, ressalta-se que apenas as cores estruturais possuem a caraceterística de apresentar diferentes padrões de cor conforme a mudança de posição do observador (BEVERLEY J. G.; HEATHER, 2010).

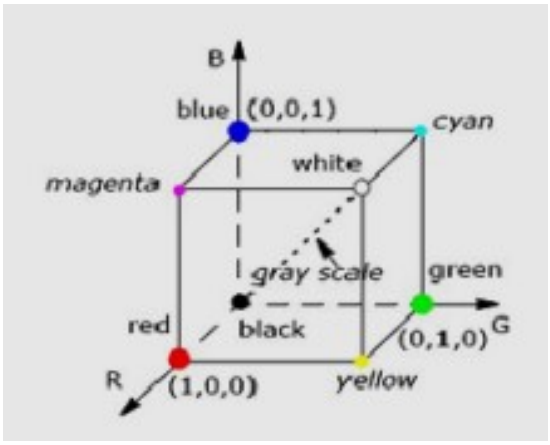
2.1.2 Modelo de cores RGB

O modelo de cores RGB deriva das três cores primárias: vermelho (*red*), verde (*green*) e azul (*blue*), que podem ser combinadas para produzir uma nova gama de cores, conforme explicitado na Figura 3a. Esse espaço de cores RGB pode ser representado como um cubo normalizado, onde os eixos representam às intesidades das cores vermelho, verde e azul, com valores variando de 0 até 1 (IBRAHEEM, 2012). Dentro dessa representação, o preto está situado na origem do sistema de coordenadas (0,0,0) e o branco no vértice oposto (1,1,1), como é destacado na Figura 3b.

Figura 3 – Representações do modelo de cores RGB.



(a) Representação das cores primárias.



(b) Cubo que representa o espaço de cores RGB normalizado.

Fonte: IBRAHEEM (2012)

É importante ressaltar que esse modelo serve como base para a maior parte das aplicações de imagem digitais, como os formatos JPEG e MPEJ, onde os píxeis armazenados geralmente são representados por 8 bits e cada um dos canais apresenta valores entre 0 e 255 (Plataniotis; Venetsanopoulos, 2000).

2.1.3 Modelo *Grayscale*

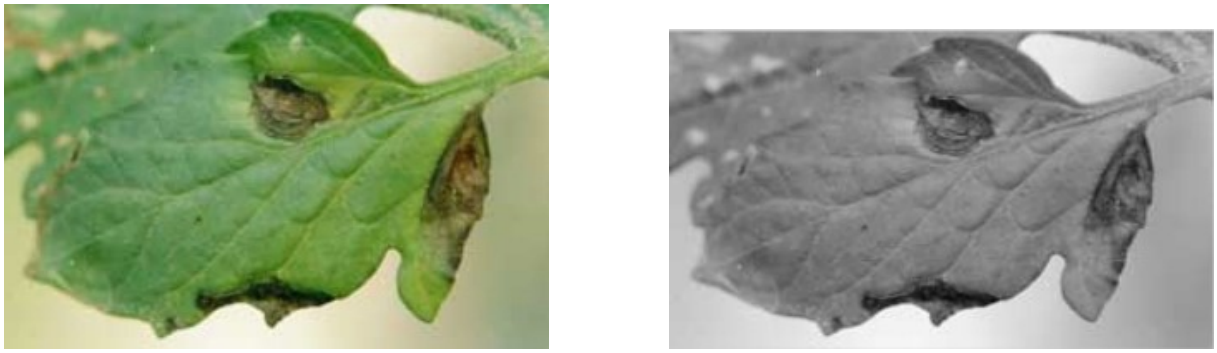
As imagens capturadas, baseadas no modelo RGB, em alguns casos podem ser convertidas para um modelo de cores que utiliza apenas uma componente, a fim de otimizar o pré-processamento. No modelo *Grayscale*, a imagem é monocromática e cada pixel contém

apenas a informação referente à quantidade de luz presente (Padmavathi; Thangadurai, 2016). Embora existam diversos métodos de conversão, o procedimento usual consiste em obter os valores das três cores primárias do modelo RGB e realizar uma soma ponderada dessas componentes, como exemplificado pela Equação 1.

$$f(R, G, B) = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B \tag{1}$$

Assim como no modelo RGB, o modelo mais comum de armazenamento é o de 8 bits, ou seja, a intensidade de cada pixel pode ter valores entre 0 e 255, sendo 0 referente a cor preta e 255 a branca (Kumar; Verma, 2010). Para melhor compreender essa transformação, observe as Figuras 4a e 4b.

Figura 4 – Conversão de imagem do modelo RGB para o modelo *Grayscale*.



(a) Imagem original no modelo RGB. (b) Imagem convertida para *Grayscale*.

Fonte: Padmavathi e Thangadurai (2016)

2.1.4 Processamento digital de imagens

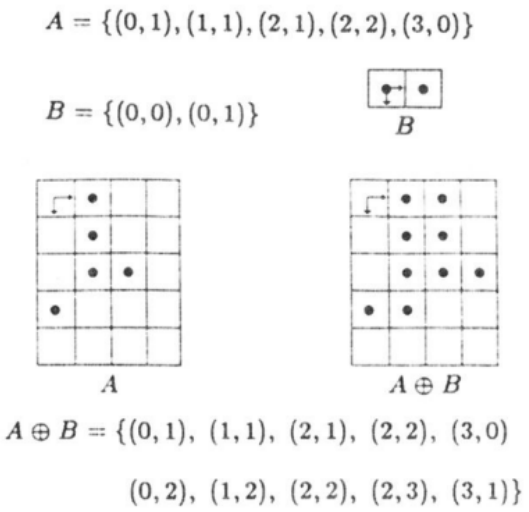
A tecnologia de processamento digital de imagem, do inglês *digital image processing* (DIP), é responsável pela transformação de imagens em sinais digitais, possibilitando a utilização de algoritmos de computadores para realizar modificações, como aprimoramentos de qualidade, análises e até mesmo reconstruções de imagens. (HUANG, 2022).

Segundo DONGUR (2022), essa tecnologia começou a ser bastante desenvolvida por volta de 1960 em lugares como o *Massachusetts Institute Of Technology*, *University of Maryland* e outros centros de pesquisas, sendo um pouco restringida devido aos altos custos dos equipamentos naquele momento. Entretanto, com o rápido avanço da computação, que permitiu a construção de *hardwares* mais poderosos e baratos, as tecnologias de DIP apresentaram grande desenvolvimento e assumiram um papel importante em aplicações reais.

A utilização de filtros morfológicos, que são baseados em formas geométricas de uma maneira matemática, é uma das aplicações mais conhecidas para realização desse tipo de processamento de forma não linear. Dessa maneira, as operações tendem a simplificar

os dados da imagem, preservando as caraterísticas essênciais de suas formas e eliminando apenas os ruídos (HARALICK Robert M.; STERNBERG, 1987). Para melhor exemplificar, uma operação de dilatação está representada na Figura 5, onde são apresentadas todas as possíveis somas vetoriais considerando um elemento vindo de A e um de B

Figura 5 – Dilatação de A por B



Fonte: HARALICK Robert M.; STERNBERG (1987)

Vale destacar que a utilização de modelos de aprendizado profundo está trazendo resultados animadores no campo de DIP, principalmente na área de classificação de imagens. Estudos recentes evidenciaram que abordagens desse tipo, principalmente quando combinadas com a utilização de CNNs, trouxeram melhorias significativas de performance, quando comparado aos métodos convencionais (MAHMOOD, 2024).

2.1.5 Redes neurais convolucionais (CNNs)

Devido à notável capacidade de identificação de padrões visuais, as CNNs se tornaram amplamente empregadas em atividades com foco em processamento de imagens (Taye, 2023). Esse tipo de redes neurais têm desempenhado papel central no avanço da visão computacional, sendo fundamentais em aplicações práticas, como no campo da radiologia, onde são aplicadas para tarefas de classificação, segmentação e até detecção de lesões (YAMASHITA, 2018), o que demonstra sua eficácia em diversos contextos.

Vale ainda ressaltar que as CNNs são um subgrupo dos modelos de aprendizado profundo, que por sua vez representam um subgrupo dos modelos de aprendizado de máquina, que fazem parte do grande campo da inteligência artificial, como evidenciado no diagrama apresentado na Figura 6.

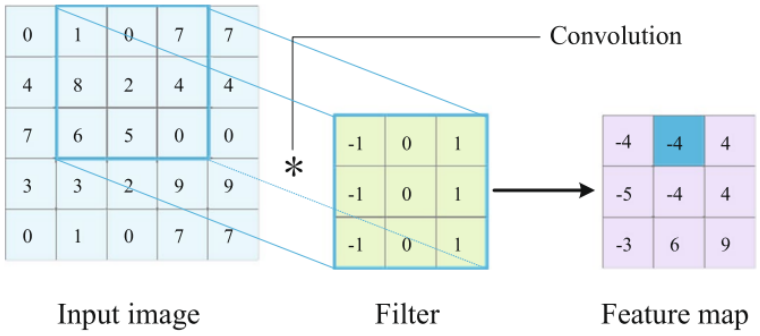
Figura 6 – Contextualização das CNNs no campo da inteligência artificial.

Fonte: Elaborada pelo autor.

Segundo Keiron e Ryan (2015), a estrutura das CNNs é tipicamente composta por 3 tipos de camadas:

- **Camada de convolução:** responsável por extrair características e padrões através de um conjunto de filtros, ou *kernels*, que podem ter diferentes dimensões e que são utilizados em conjunto com o dado de entrada, produzindo uma nova imagem, como demonstrado na Figura 7.

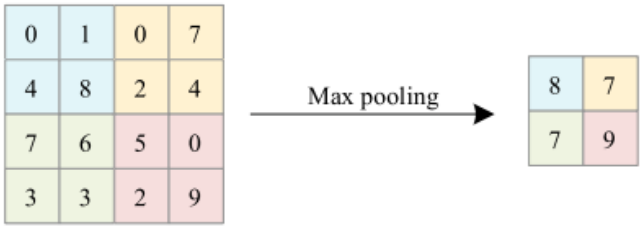
Figura 7 – Representação do processo de convolução.



Fonte: ZHAO (2024)

- **Camada de *pooling*:** responsável por reduzir a dimensão das saídas da camada de convolução, enquanto mantém os dados mais importantes, realizando um agrupamento através de filtros que consideram a média, mínimo ou máximo, como demonstrado na Figura 8.

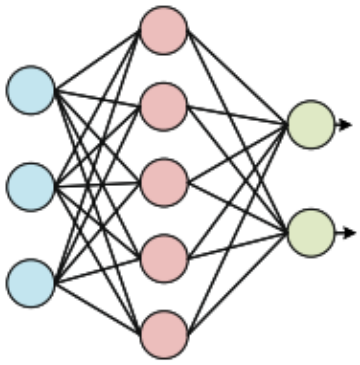
Figura 8 – Representação do processo de *pooling*.



Fonte: ZHAO (2024)

- **Camada totalmente conectada:** responsável por combinar cada uma das informações extraídas nas camadas anteriores e aprender como elas se relacionam, armazenando os padrões específicos no que são chamados de neurônios. Por isso, é comumente empregado no final da rede, para produzir as previsões finais, como mostrado na Figura 9.

Figura 9 – Representação da camada totalmente conectada.



Fonte: ZHAO (2024)

Além disso, a existência de funções de ativação, como a *Rectified Linear Unit* (ReLU), é bastante comum nesse tipo de arquitetura, sendo empregadas para alterar ou até mesmo excluir as saídas, de forma a introduzir não linearidade nas camadas e melhorar a capacidade de aprendizado e representativdade dos resultados (ZHAO, 2024).

Por fim, vale destacar que, apesar da capacidade de extração de características sem que seja necessária a supervisão humana, o que contribuiu para a ampla aplicabilidade das CNNs, a utilização desse tipo de modelo ainda apresenta desafios significativos, como a necessidade de conjuntos de dados extensos e devidamente rotulados para treinamento adequado. Além disso, problemas como *overfitting* são comuns, devido à grande quantidade de parâmetros envolvidos que se correlacionam de maneiras complexas, o que pode reduzir a perfomance do modelo com os dados de teste (ALZUBAIDI, 2021).

2.2 Materiais e métodos

2.2.1 Materiais utilizados

Para identificar possíveis padrões na mudança de cor da fita adesiva holográfica durante a alteração do ângulo de observação, torna-se necessário realizar um experimento, no qual são utilizados os seguintes materiais:

- **Fita adesiva holográfica:** item central do experimento, cujo comportamento de mudança de cor será analisado;
- **Papel sulfite branco:** utilizado como fundo para fixação da fita adesiva holográfica, a fim de facilitar o pré-processamento das imagens;
- **Luminária genérica com lâmpada 12W 6500K:** utilizada para garantir o correto posicionamento da iluminação no experimento;
- **Celular Galaxy S23:** utilizado como câmera para captação das imagens, com um zoom de 3x;
- **Transferidor 360 graus:** utilizado para controlar a variação do ângulo da fita adesiva holográfica e o posicionamento dela em relação à luz e câmera;
- **Tripés:** utilizados para garantir o correto posicionamento da câmera e da fita adesiva holográfica;
- **Esquadros de acrílico:** utilizados para garantir o correto posicionamento da câmera, fita adesiva holográfica e luminária;
- **Trena métrica de aço:** utilizada para realizar medições de distâncias entre os objetos;
- **Fita crepe:** utilizada para marcações e fixações, a fim de garantir que o posicionamento dos objetos seja mantido durante a condução do experimento;
- **Computador:** utilizado para o processamento das imagens.

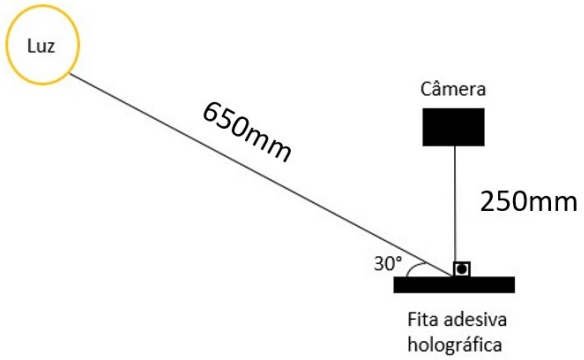
2.2.2 Método de montagem

A montagem do experimento, apresentada na Figura 10, foi executada de maneira a garantir que o posicionamento dos materiais seguisse os esquemas representados nas Figuras 11 e 12.

Figura 10 – Montagem real.

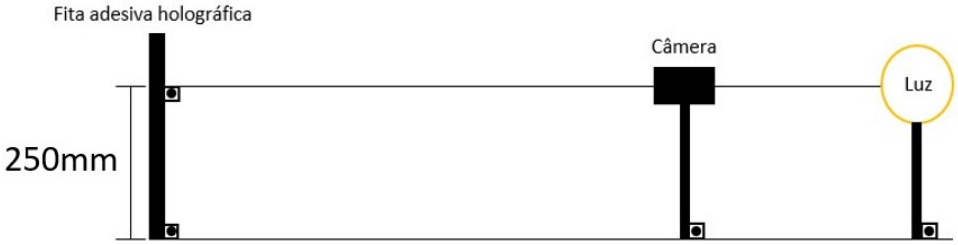
Fonte: Elaborada pelo autor.

Figura 11 – Vista superior da montagem.



Fonte: Elaborada pelo autor.

Figura 12 – Vista lateral da montagem.

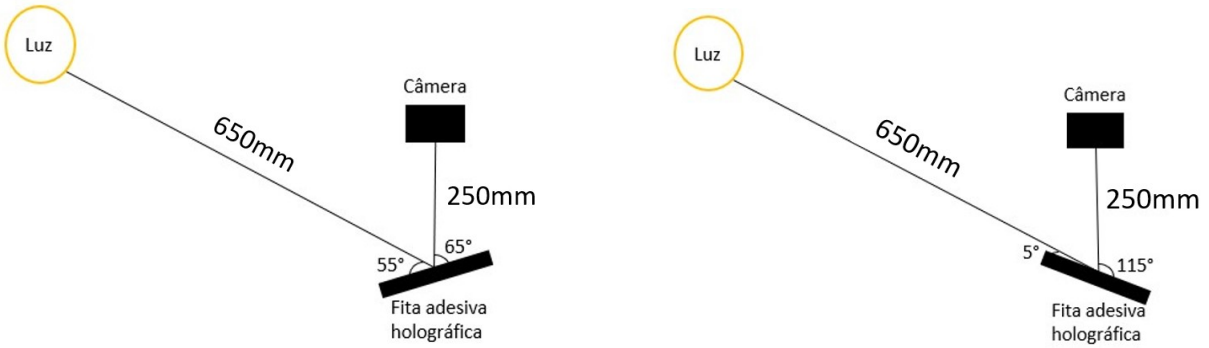


Fonte: Elaborada pelo autor.

É importante destacar que, para observar o comportamento de mudança de cores durante o experimento, foi necessário alterar a configuração da montagem, variando o ângulo referente à fita adesiva holográfica em passos de 1 grau, até atingir 25 graus de variação para cada um dos lados. Portanto, estabeleceu-se que, quando a fita está a 25

graus negativos ela se encontra na posição ilustrada na Figura 13a, e quando está a 25 graus positivos se encontra na posição ilustrada na Figura 13b.

Figura 13 – Vistas superiores da montagem após alterar o ângulo em ±25 graus.



(a) Vista superior da montagem após alterar o ângulo em 25 graus negativos.

(b) Vista superior da montagem após alterar o ângulo em 25 graus positivos.

Fonte: Elaborada pelo autor.

2.3 Técnicas utilizadas

2.3.1 Software

A linguagem de programação escolhida para a análise das imagens capturadas foi Python, devido a sua sintaxe simples e ampla disponibilidade de bibliotecas especializadas. Além disso, sua grande comunidade de desenvolvedores facilita a busca por soluções semelhantes já disponíveis e contribui na implementação de recursos adicionais.

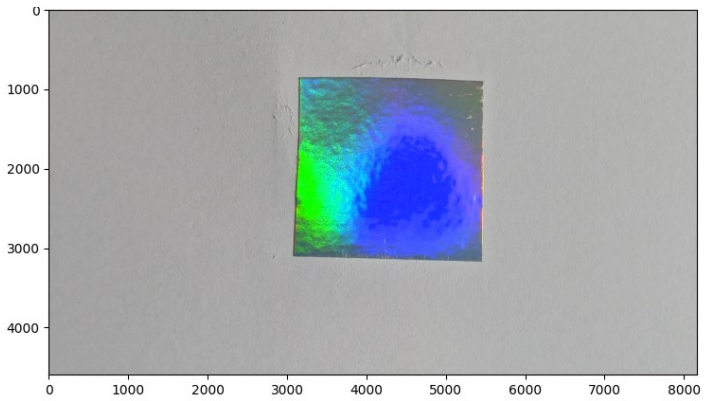
Para realizar o processamento das imagens, a principal biblioteca utilizada foi a *OpenCV*, que conta com mais de 2500 algoritmos otimizados e é amplamente empregada no processamento de imagens (Bradski, 2000), possibilitando a identificação da cor no ponto central das fitas. Além disso, utilizou-se a biblioteca *matplotlib*, que permite a visualização de imagens e gráficos de maneira simples (Hunter, 2007), e a *numpy*, que permite estabelecer parâmetros numéricos utilizados em algumas funções (Harris *et al.*, 2020).

Para a implementação da CNN, o principal *framework* utilizado foi o *TensorFlow*, que permite a implementação e treinamento de algoritmos de aprendizado de máquina, sendo amplamente utilizado para conduzir pesquisas relacionadas com modelos de redes profundas em áreas como visão computacional (Abadi *et al.*, 2015). Além disso, utilizou-se a interface de programação de aplicações, do inglês *application programming interface* (API), *Keras* para estrutução do modelo (Chollet *et al.*, 2015), e a biblioteca *scikit-learn* para separar e estruturar o conjunto de dados utilizado (Pedregosa *et al.*, 2011).

2.3.2 Processamento das imagens

Inicialmente, captura-se a imagem original (8160x4592), que, neste exemplo, corresponde à configuração em que o ângulo da fita adesiva holográfica está alterado em dois graus no sentido positivo. A imagem da fita nessa condição está apresentada na Figura 14 e servirá como base para os próximos passos do processamento.

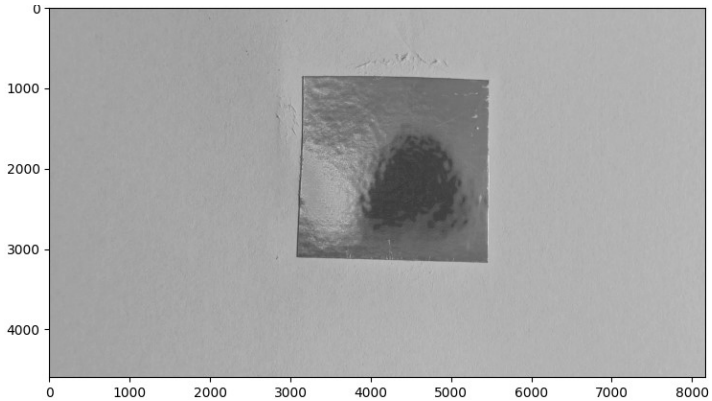
Figura 14 – Imagem original.



Fonte: Elaborada pelo autor.

Em seguida, para prosseguir com as manipulações, torna-se conveniente converter a imagem para o modelo *Grayscale*. A conversão do modelo de cor da imagem é realizada utilizando a função *cvtColor*, que transforma, neste caso, o modelo de cores RGB em *Grayscale*. A imagem resultante dessa conversão pode ser observada na Figura 15.

Figura 15 – Imagem no modelo Grayscale.

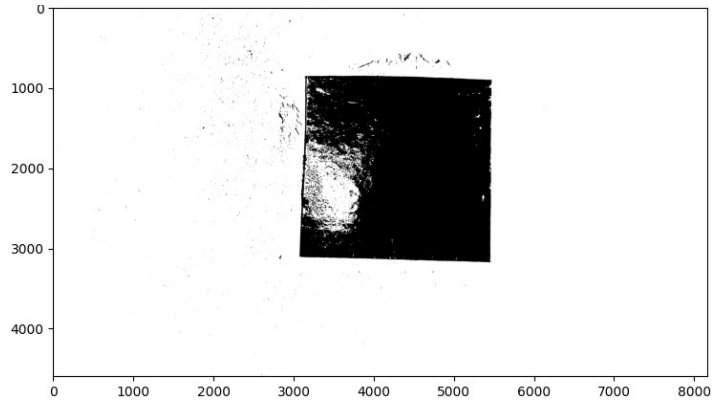


Fonte: Elaborada pelo autor.

Posteriormente, para diferenciar o fundo branco do papel da fita adesiva holográfica, é necessário binarizar a imagem previamente convertida para o modelo *Grayscale*. Para isso, é utilizada a função *threshold*, que aplica um limite de intensidade aos píxeis da imagem, onde os que apresentam intensidade superior ao valor de limiar são convertidos

para branco, enquanto os demais são convertidos para preto. O resultado da binarização está apresentado na Figura 16.

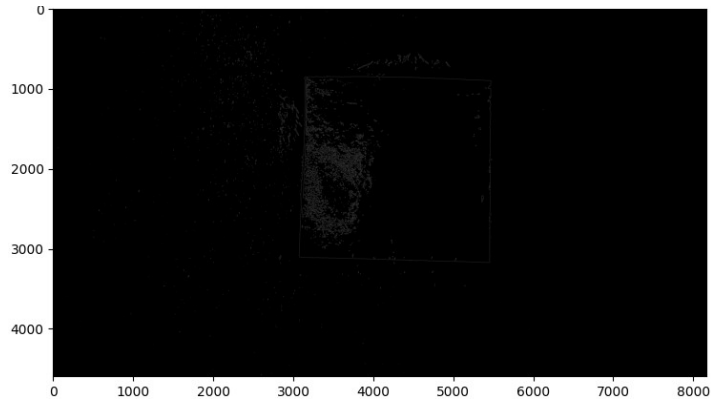
Figura 16 – Imagem binarizada.



Fonte: Elaborada pelo autor.

Após a binarização, torna-se possível detectar as bordas da fita com maior precisão e eficiência. Para isso, é utilizada a função *Canny*, que aplica um algoritmo de detecção de bordas baseado em gradientes de intesidade. Esse algoritmo destaca as transições entre áreas claras e escuras da imagem, permitindo a identificação de contornos e detalhes. O resultado após a detecção de bordas pode ser observado na Figura 17.

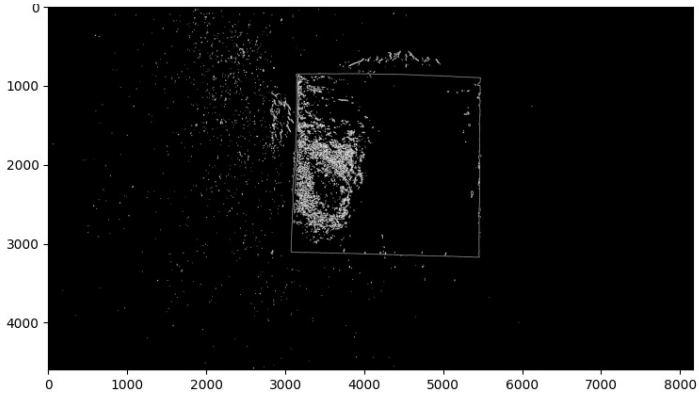
Figura 17 – Imagem após detecção de bordas.



Fonte: Elaborada pelo autor.

Em seguida, para aprimorar a detecção das bordas, utiliza-se a função *dilate*. Essa operação expande as regiões claras da imagem binarizada, realçando as bordas previamente detectadas e facilitando a identificação de contornos. A imagem resultante desse procedimento está mostrada na Figura 18.

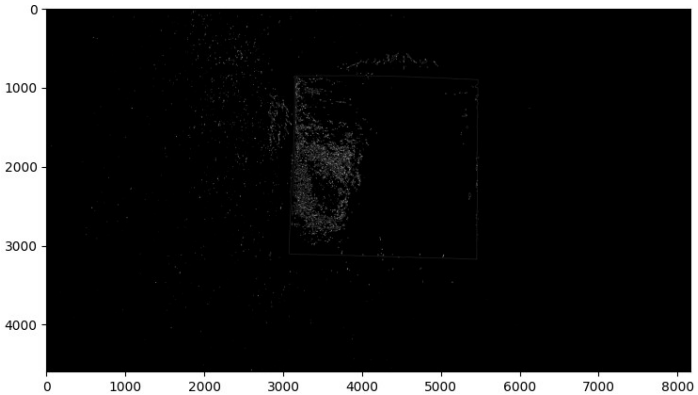
Figura 18 – Imagem após dilatação.



Fonte: Elaborada pelo autor.

Para reduzir os ruídos presentes na detecção das bordas após a utilização do *dilate*, aplica-se a função *erosion*. Essa função atua diminuindo as regiões claras na imagem, suavizando as bordas e eliminando pequenas imperfeições causadas por ruídos, como observado na Figura 19.

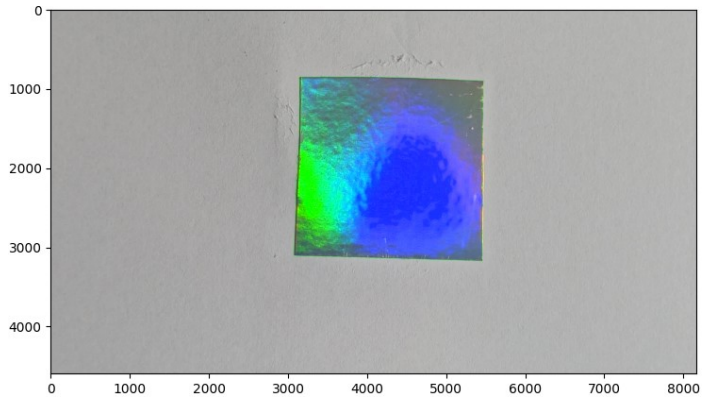
Figura 19 – Imagem após erosão.



Fonte: Elaborada pelo autor.

Após o aprimoramento da detecção das bordas e redução de ruídos, é possível detectar o contorno que corresponde ao da fita com maior precisão. Para isso, utiliza-se a função *findContours*, que identifica os contornos da imagem através da análise de transições de intensidade, localizando as regiões fechadas onde acontecem mudanças significativas de cor e agrupando como contornos. Dessa forma, é possível selecionar o maior contorno, que corresponde ao da fita, conforme ilustrado na Figura 20.

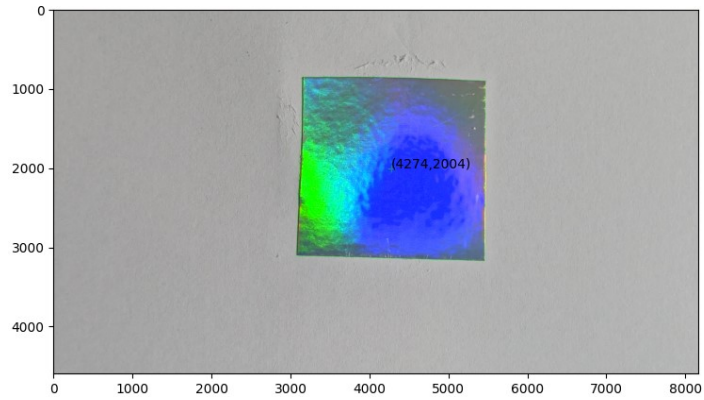
Figura 20 – Imagem original com o contorno destacado.



Fonte: Elaborada pelo autor.

Por fim, é possível determinar as coordenadas do ponto central da fita através da análise do seu contorno. Para isso, utiliza-se a função *boundingRect*, que calcula o retângulo referente ao contorno da fita e retorna suas dimensões, além das coordenadas do canto superior esquerdo. Com essas informações, é possível calcular as coordenadas do centro do retângulo, que correspondem à localização do ponto central da fita, conforme apresentado na Figura 21.

Figura 21 – Imagem original com o contorno e o ponto central destacados.



Fonte: Elaborada pelo autor.

Após o processamento da imagem e definição das coordenadas correspondentes ao ponto central da fita adesiva holográfica, torna-se possível obter os valores das cores primárias do modelo RGB da imagem original nesse ponto, que neste exemplo é: $Red = 20$, $Green = 46$ e $Blue = 255$.

2.3.3 Criação e treinamento do modelo de CNN

Primeiramente, organizou-se o conjunto de dados contendo as imagens da fita posicionada entre 25 graus negativos e 25 graus positivos, com a devida rotulação em

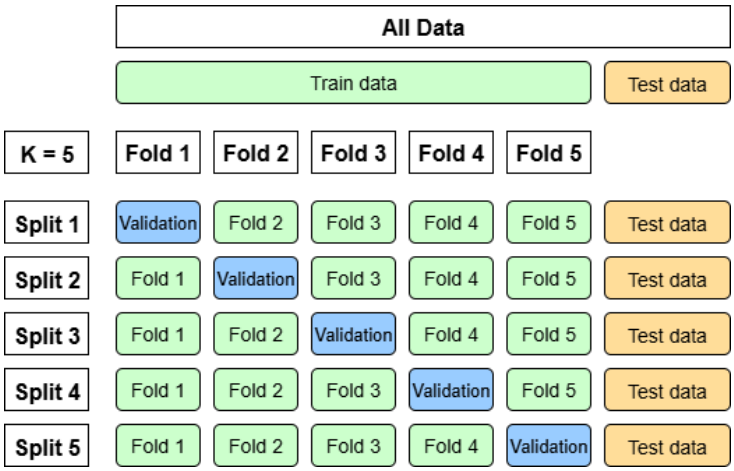
cada uma das imagens e mantendo uma resolução de 128x128 píxeis, para que o modelo pudesse ser treinado de forma adequada.

Posteriormente, através da função *train_test_split*, da biblioteca *scikit-learn*, separou-se uma parcela contendo 80 por cento do conjunto de dados para o treinamento dos modelos, entitulada de *train data*, e o outra contendo 20 por cento para validação final do modelo, entitulada de *test data*.

Ainda a cerca da separação, a fim de evitar o *overfitting* e entender as variações que o modelo poderia sofrer com diferentes separações randômicas do conjunto de dados, utilizou-se a abordagem de *K-fold cross-validation*, implementada através da função *KFold*, da biblioteca *scikit-learn*, que separou o conjunto de treinamento em 5 conjuntos menores, chamados de *folds*.

Através dessa abordagem, o modelo pôde ser treinado 5 vezes, variando os folds utilizados para treinamento e para validação em cada um dos momentos. Vale ressaltar que, após cada um desses treinamentos, os dados de teste foram utilizados para avaliação final do modelo. Toda essa divisão pode ser melhor observada através do esquema mostrado na Figura 22.

Figura 22 – Separação do dataset.



Fonte: Elaborada pelo autor.

Posteriormente, definiu-se que o modelo utilizado para essa aplicação seria o *Sequential*, da API *Keras*, que apresenta uma arquitetura simples e permite empilhar camadas de forma linear, tornando-se ideal para esse cenário, onde o fluxo de dados segue uma única direção. Dessa forma, garante-se que o modelo retorne um único valor final, correspondente à estimativa do ângulo de observação da fita na imagem.

Para definir a arquitetura do modelo, optou-se por iniciar com uma primeira camada de convolução 2D com 16 filtros e dimensão de kernel 3x3, com a função de ativação ReLU introduzindo uma não linearidade na rede, e um *input shape* com as dimensões da imagem

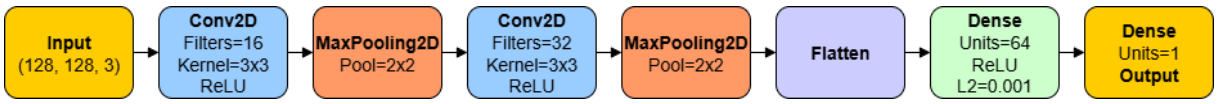
(128x128) e 3 canais, referentes ao modelo de cor RGB.

Posteriormente, adicionou-se uma camada de *polling* 2D considerando o valor máximo, e com *pool size* de 2x2. Em seguida, uma nova camada de convolução 2D com 32 filtros com as demais caraterísticas iguais à primeira camada. Logo após, uma nova camada de *pooling* com caraterísticas iguais à segunda camada.

Prosseguindo para o fim, acrescentou-se uma camada *Flatten*, para transformar a saída das camadas convolucionais e de pooling, que têm formato multidimensional, em um vetor unidimensional. Em seguida, adicionou-se uma camada totalmente conectada, com 64 unidades, também com função de ativação ReLU, e com regularização L2, conhecida como *weight decay*, para evitar *overfitting* e melhorar a generalização do modelo, adicionando um termo extra à função de perda durante o treinamento.

Para finalizar, uma nova camada totalmente conectada, com 1 unidade, para retornar o resultado final. A estrutura pode ser melhor compreendida através da Figura 23.

Figura 23 – Arquitetura do modelo.



Fonte: Elaborada pelo autor.

Para a compilação desse modelo, utilizou-se como função de perda para a otimização o erro quadrático médio, do inglês *mean squared error* (MSE), uma função comumente utilizada nesses tipos de aplicações, pois minimiza grandes desvios e garante um gradiente suave para a otimização, uma vez que penaliza os grandes desvios, elevando os erros ao quadrado, como observado na Equação 2.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2}$$

Além disso, utilizou-se como métrica para avaliação do modelo o erro absoluto médio, do inglês *mean absolute error* (MAE). Isso, pois ela apresenta características complementares ao MSE, uma vez que mede o erro real médio, sem pesos extras aos grandes desvios, sendo mais robusta aos *outliers*, como evidente na Equação 3.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{3}$$

Vale apontar que, para o treinamento desse modelo, estabeleceu-se como limite máximo de 200 épocas, um batch size de 128, além de uma estratégia de redução da taxa

de aprendizado através da função *ReduceLROnPlateau*, e outra de parada antecipada através da função *EarlyStopping*.

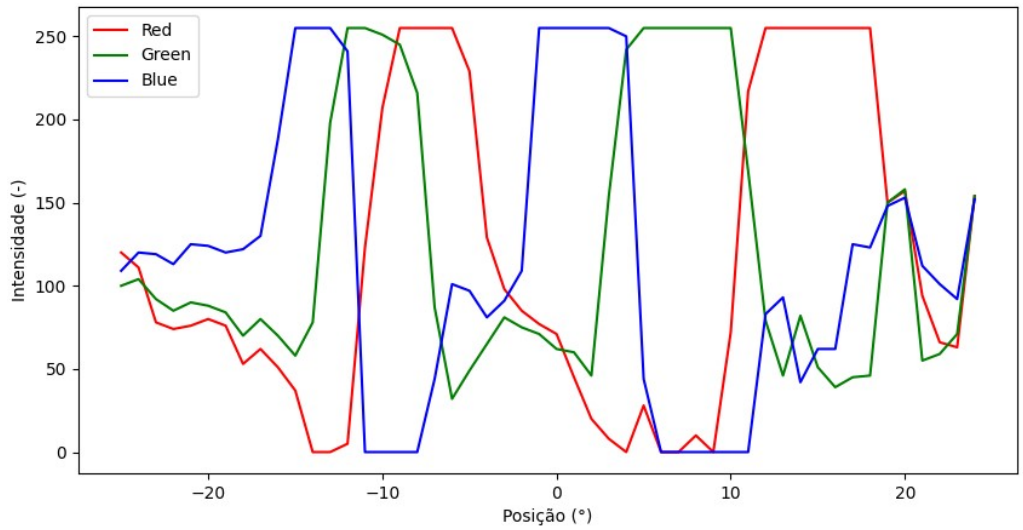
Dessa forma, faz-se com que o aprendizado diminua pela metade quando a função de perda (MSE) não diminui durante 10 épocas consecutivas, e também que o treinamento termine caso a função de perda (MSE) não diminua por 20 épocas consecutivas. Isso, para garantir um melhor comportamento durante o aprendizado e evitar *overfitting*.

2.4 Resultados e discussões

2.4.1 Análise do padrão de mudança de cores da fita

Conforme descrito na Seção 2.2.2, intitulada Método de montagem, foi realizado o experimento inicial com o objetivo de identificar um padrão na mudança de cor da fita adesiva holográfica. Vale salientar que, nesse experimento, a configuração foi alterada variando o ângulo referente à fita em passos de 1 grau, até atingir 25 graus de variação para cada um dos lados. Posteriormente, conforme o procedimento detalhado na Seção 2.3.2, intitulada Processamento das imagens, foram obtidos os dados referentes aos valores das cores primárias do modelo RGB no ponto central da fita em cada uma das 51 posições analisadas. Os resultados estão compilados no gráfico apresentado na Figura 24.

Figura 24 – Valores de R, G e B obtidos no centro da fita pela variação do ângulo.



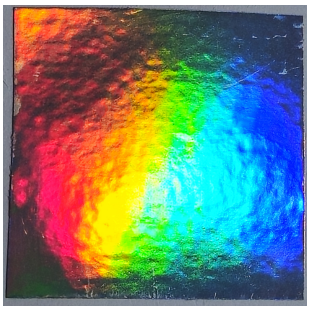
Fonte: Elaborada pelo autor.

Diante do exposto, torna-se evidente que os valores das componentes R (*Red*), G (*Green*) e B (*Blue*) variam conforme o ângulo da fita adesiva holográfica é alterado, ou seja, ela realmente apresenta a propriedade de iridescência.

Além disso, o comportamento geral evidencia uma relação não linear entre a variação angular e os valores de cada uma das componentes, que apresentam picos e quedas abruptas em intervalos angulares pequenos, principalmente quando a variação

do ângulo é negativa. Destaca-se que essa variação está intimamente relacionada com as variações das cores principais, conforme apresentado na Figura 3a e descrito na Seção 2.1.2, intitulada Modelo de cores RGB. Nesse contexto, observa-se uma transição sequencial de cores, iniciando pela cor azul e seguindo para o ciano, verde, amarelo, vermelho e magenta, como ilustrado na Figura 25.

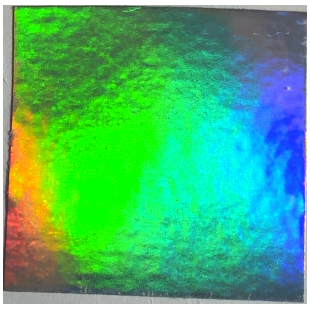
Figura 25 – Imagem original com ângulo alterado em 10 graus negativos.



Fonte: Elaborada pelo autor.

Vale ressaltar que o padrão de comportamento se repete quando acontece a variação positiva do ângulo. Entretanto, a transição sequencial de cores acontece em um intervalo angular maior, indicando uma distribuição mais espaçada das cores nesse cenário. Essa caraterística pode ser observada através da Figura 26.

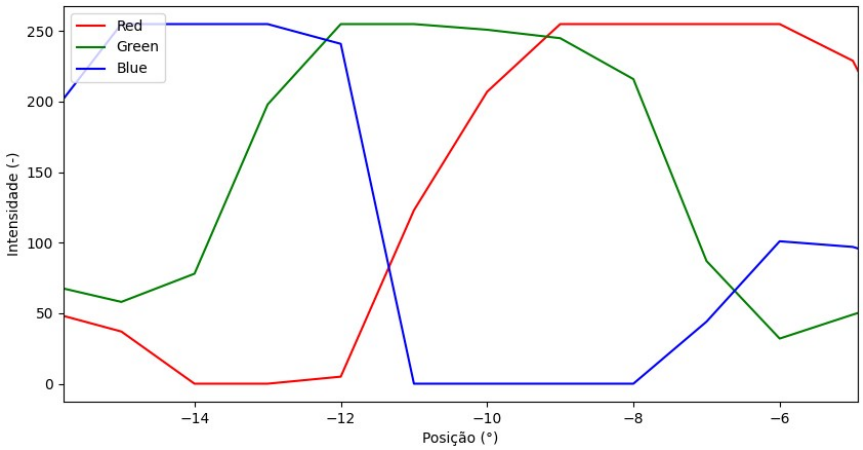
Figura 26 – Imagem original com ângulo alterado em 5 graus positivos.



Fonte: Elaborada pelo autor.

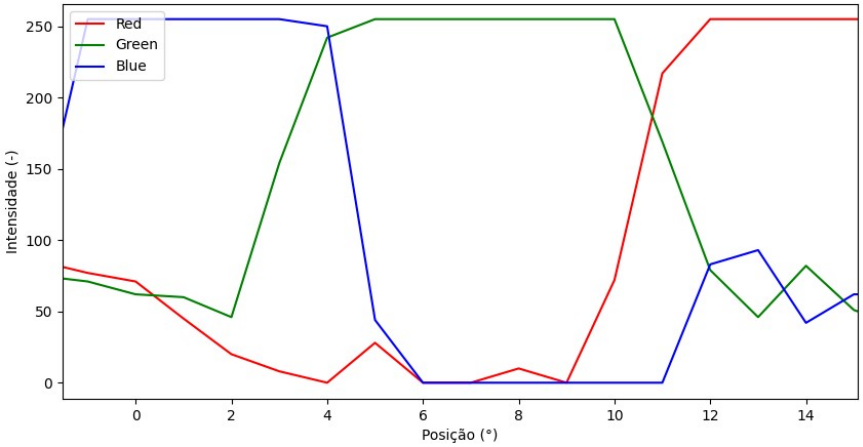
De maneira geral, o padrão identificado na mudança de cor da fita adesiva holográfica em função da variação angular apresenta características que podem ser relevantes para a determinação de posição. Em particular, destaca-se o comportamento de queda abrupta na intensidade da coloração azul, acompanhada do aumento de intensidade da coloração verde, que ocorre nos intervalos de -11 a -8 graus e de +5 a +11 graus, como evidenciado nas Figuras 27 e 28, respectivamente.

Figura 27 – Zoom na área de interesse (intervalo negativo).



Fonte: Elaborada pelo autor.

Figura 28 – Zoom na área de interesse (intervalo positivo).



Fonte: Elaborada pelo autor.

No intervalo negativo, a variação ocorre em um espaço angular menor, de apenas 4 graus, o que torna esse intervalo especialmente útil para cenários de maior precisão, uma vez que a resposta é sensível a pequenos deslocamentos. Nessa perspectiva, por exemplo, torna-se possível identificar com precisão o início, meio e fim de um período de vibração utilizando como parâmetros as intensidades das cores primárias. Dessa forma, possibilitando o cálculo da frequência de vibração de uma estrutura.

2.4.2 Análise dos resultados previstos pelo modelo

Conforme o procedimento detalhado na Seção 2.3.3, intitulada Criação e treinamento do modelo de CNN, realizou-se o treinamento do modelo com cada um dos *splits* e foram obtidos os resultados compilados na Tabela 1.

Tabela 1 – Resultado do treinamento em cada um dos 5 *splits*

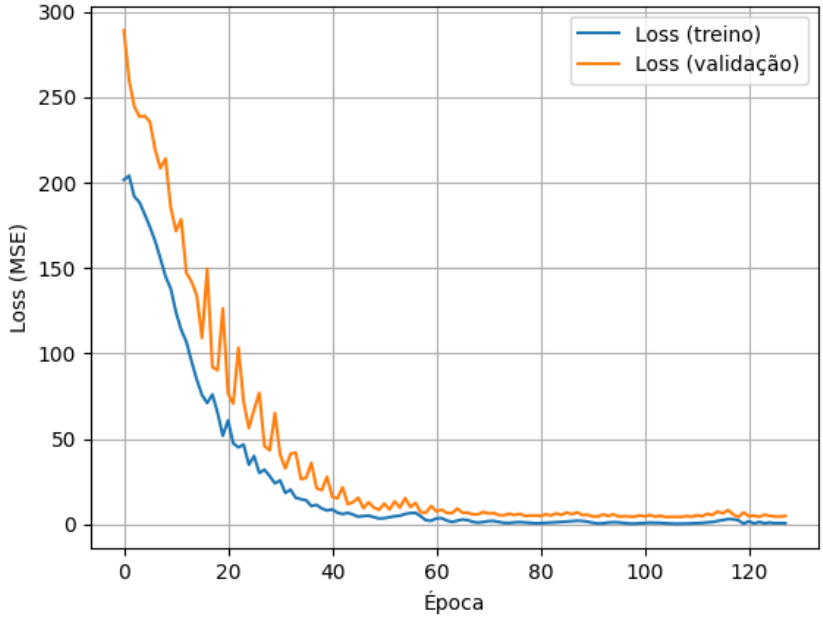
Separação (Split)	**Perda (MSE)**	**Métrica (MAE)**
1	1.1974	0.8504
2	1.9082	1.1827
3	2.0700	1.2008
4	1.2844	0.7991
5	5.4511	1.5350
Média (σ)	2.38 ($\pm$ 1.75)	1.11 ($\pm$ 0.30)

Os resultados das diferentes divisões de treino e teste indicam, de maneira geral, um desempenho satisfatório do modelo. Os valores médios e desvios padrões do MSE e MAE evidenciam que o modelo foi capaz de aprender de forma adequada os padrões presentes no conjunto de dados. Além disso, esse comportamento sugere que a abordagem adotada apresenta boa capacidade de generalização, uma vez que, em grande parte das divisões realizadas, os resultados permaneceram próximos entre si.

Entretanto, observa-se um certo aumento tanto no MSE quanto no MAE durante o *split* 5, o que pode ser atribuído à forma como a divisão do conjunto de dados foi realizada. Como o *dataset* utilizado não apresenta grande quantidade de amostras, é possível que a parcela de dados utilizada para o treinamento nesse cenário específico contenha imagens menos representativas, ou ainda uma concentração dentro de determinado intervalo, ocasionando o aumento dos erros.

Apesar disso, como os demais resultados apresentaram estabilidade e desempenho satisfatório, conclui-se que o impacto da divisão dos dados, embora existente, não compromete de maneira relevante a qualidade global do modelo. Diante do exposto, adota-se como referência para analises posteriores o modelo treinado com o *split* 4, que apresentou o melhor resultado quanto à métrica (MAE). A sua curva de aprendizado está apresentada na Figura 29.

Figura 29 – Curva de aprendizado do modelo treinado com o *split* 4.



Fonte: Elaborada pelo autor.

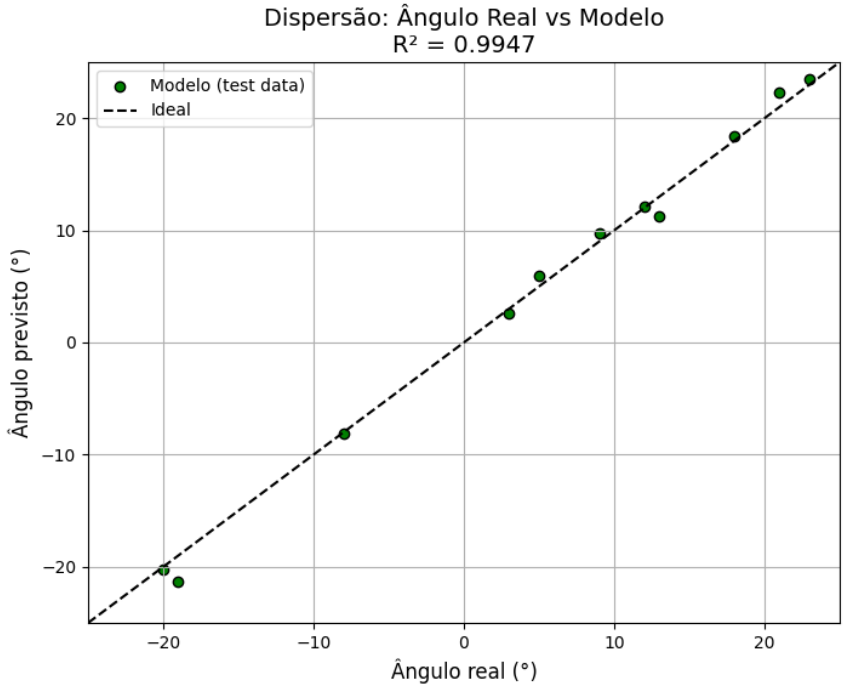
A curva exposta indica um comportamento satisfatório, evidenciando o rápido aprendizado do modelo e a ausência de sinais claros de *overfitting*. Ressalta-se ainda a semelhança na convergência entre a curva de treino e de validação, indicando uma boa generalização do modelo.

Observa-se, contudo, a presença de pequenas oscilações na curva de validação ao longo do treinamento, o que é de certa forma esperado, uma vez que pequenas variações no conjunto de dados podem causar flutuações na função de perda a cada época. Entretanto, alguns mecanismos utilizados, como a redução automática da taxa de aprendizado, *ReduceLROnPlateau*, contribuem para amortecer essas oscilações ao longo das épocas, favorecendo uma convergência mais estável. Assim, embora as flutuações sejam visíveis, o comportamento médio das curvas indica estabilidade e boa capacidade de generalização do modelo, sem fortes sinais de *overfitting*.

Vale ainda pontuar que, por ter sido adotada uma estratégia de parada antecipada, o critério estabelecido permitiu que o treinamento se estendesse até aproximadamente 120 épocas. Dessa forma, o mecanismo se apresentou útil, pois interrompeu o processo antes do limite de 200 épocas, diminuindo o gasto computacional e ainda assim garantindo eficácia no treinamento e convergência do modelo.

Por fim, realizou-se uma comparação entre os ângulos estimados pelo modelo de referência e os ângulos reais das imagens contidas no conjunto de teste, cujo modelo não foi exposto durante o treinamento e validação. O resultado está apresentado na Figura 30.

Figura 30 – Gráfico comparando ângulos reais com os estimados pelo modelo.



Fonte: Elaborada pelo autor.

Como observado através do gráfico, os pontos se distribuem bastante próximos à reta ideal, indicando elevada correlação entre os ângulos reais e os estimados, o que evidencia as boas estimativas do modelo. Esse comportamento é confirmado pelo coeficiente de determinação $R^2 = 0{,}9947$, apontando que praticamente toda a variabilidade dos dados reais foi explicada pelo modelo.

Diante dessa perspectiva de um elevado R^2 combinado com um MAE reduzido, assegura-se que o modelo foi capaz de generalizar adequadamente esse tipo de problema e fornecer previsões consistentes e precisas, demonstrando sua qualidade através das verificações utilizando o conjunto de teste.

3 CONCLUSÃO

O trabalho demonstrou a viabilidade de realizar uma análise da variação de cor de uma fita adesiva holográfica em função do ângulo de observação, através do monitoramento dos canais RGB. Os resultados indicaram uma relação forte entre a variação angular e o comportamento óptico da fita, evidenciando o seu potencial como um sensor simples e de baixo custo.

Embora tenham sido identificadas oscilações e dispersões nos resultados experimentais, grande parte dessas variações está associada a fatores práticos, como o sistema de fixação, iluminação e o controle da variação angular. Dessa forma, fica evidente que, com maior refinamento experimental, torna-se plausível alcançar uma precisão significativamente superior.

A partir dessas melhorias, abre-se inclusive a perspectiva de ampliar a aplicação do método para além da determinação de ângulos em um único eixo. Dada a natureza da reflexão holográfica, é possível que com ajustes adequados no aparato experimental seja possível extrair informações relativas a diferentes eixos de variação angular. Isso ampliaria de maneira considerável o potencial de uso da técnica, fazendo com que ela se aproxime de soluções mais sofisticadas de sensoriamento óptico, mas ainda preservando sua simplicidade e acessibilidade.

Em síntese, o estudo confirmou a viabilidade da abordagem proposta e indicou caminhos promissores para sua evolução, tanto em investigações acadêmicas quanto em possíveis aplicações..

REFERÊNCIAS

ABADI, M. *et al.* **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org.

ALZUBAIDI, L. e. a. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. **Journal of Big Data, v. 8, n. 1**, 2021.

BEVERLEY J. G.; HEATHER, M. W. Structural colour and iridescence in plants: the poorly studied relations of pigment colour. **Annals of Botany**, 2010.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

CHOLLET, F. *et al.* **Keras**. 2015. https://keras.io.

DONGUR, K. e. a. Digital image processing: Its history and application. **International Journal of Advanced Research in Computer and Communication Engineering, v.11, n. 6**, 2022.

HARALICK ROBERT M.; STERNBERG, S. R. Z. X. Image analysis using mathematical morphology. **IEEE Transactions on Pattern Analysis and Machine Intelligence, v. PAMI-9, n. 4**, 1987.

HARRIS, C. R. *et al.* Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, 2020.

HUANG, Y. Overview of research progress of digital image processing technology. **Journal of Physics: Conference Series, v. 2386, n. 1**, 2022.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, 2007.

IBRAHEEM, N. A. e. a. Understanding color models: A review. **ARPN Journal of Science and Technology**, 2012.

KEIRON, O.; RYAN, N. An introduction to convolutional neural networks. **arXiv:1511.08458**, 2015.

KUMAR, T.; VERMA, K. A theory based on conversion of RGB image to gray image. **Int. J. Comput. Appl.**, Foundation of Computer Science, 2010.

MAHMOOD, Z. Digital image processing: Advanced technologies and applications. **Applied Sciences, v. 14(14), n. 6051**, 2024.

OSORIO D.; HAM, A. Spectral reflectance and directional properties of structural coloration in bird plumage. **J Exp Biol**, 2002.

PADMAVATHI, K.; THANGADURAI, K. Implementation of RGB and grayscale images in plant leaves disease detection – comparative study. **Indian J. Sci. Technol.**, Indian Society for Education and Environment, 2016.

PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PLATANIOTIS, K. N.; VENETSANOPOULOS, A. N. **Color Image Processing and Applications**. Berlin, Germany: Springer Berlin Heidelberg, 2000.

SIMPSON, J. A.; WEINER, E. S. C. **The Oxford English dictionary**. 2. ed. New York: Oxford University Press, 1989.

TAYE, M. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. **Computation, v. 11, n. 3**, 2023.

WANG, S. e. a. A review: High-precision angle measurement technologies. **Sensors, v. 24(6), n. 1755**, 2024.

YAMASHITA, R. e. a. Convolutional neural networks: an overview and application in radiology. **Insights into Imaging, v. 9, n. 4**, 2018.

ZHAO, X. e. a. A review of convolutional neural networks in computer vision. **Artificial Intelligence Review, v. 57, n. 99**, 2024.

APÊNDICES

APÊNDICE A – CÓDIGO PARA PROCESSAMENTO DA IMAGEM.

```
#Importando bibliotecas
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

#Definindo tamanho padrão das figuras
figsize_default = (10,5)

#Carregando a imagem utilizada durante o processamento
caminhoImagem = "C:\\Users\\Otavio\\Desktop\\FotosTCC\\182.jpg"
img_bgr_original = cv.imread(caminhoImagem)
#Convertendo imagem para diferentes formatos utilizados
img_rgb = cv.cvtColor(img_bgr_original, cv.COLOR_BGR2RGB)
img_gray = cv.cvtColor(img_bgr_original, cv.COLOR_BGR2GRAY)

#Plotando imagem original no modelo RGB
plt.figure(figsize=figsize_default)
plt.title("Imagem original")
plt.imshow(img_rgb)
plt.show()

#Plotando imagem no modelo GrayScale
plt.figure(figsize=figsize_default)
plt.title("Imagem grayscale")
plt.imshow(img_gray,cmap="gray", vmin=0, vmax=255)
plt.show()

#Processando imagem
#Binarizando imagem
_, thresh = cv.threshold(img_gray, 150, 255, cv.THRESH_BINARY)
#Plotando imagem binarizada
plt.figure(figsize=figsize_default)
plt.title("Imagem binarizada")
plt.imshow(thresh,cmap="gray", vmin=0, vmax=255)
plt.show()
```

```
#Detectando bordas
img_canny = cv.Canny(thresh, 0, 0)
#Plotando imagem com a deteccao de bordas
plt.figure(figsize=figsize_default)
plt.title("Imagem com deteccao de bordas")
plt.imshow(img_canny,cmap="gray", vmin=0, vmax=255)
plt.show()

#Aplicando dilate para melhorar a detecção de contornos
kernel = np.ones((5, 5))
img_dilate = cv.dilate(img_canny, kernel, iterations=1)
#Plotando imagem após dilate
plt.figure(figsize=figsize_default)
plt.title("Imagem após dilate")
plt.imshow(img_dilate,cmap="gray", vmin=0, vmax=255)
plt.show()

#Aplicando erode para diminiur o ruído
img = cv.erode(img_dilate, kernel, iterations=1)
#Plotando imagem após erode
plt.figure(figsize=figsize_default)
plt.title("Imagem após erode")
plt.imshow(img,cmap="gray", vmin=0, vmax=255)
plt.show()

#Detectando maior contorno após a manipulação da imagem
contornos, _ = cv.findContours(img, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
maior_contorno = max(contornos, key=cv.contourArea)

#Criando uma cópia da imagem para visualização
imagem_com_contorno = img_rgb.copy()

#Desenhando o maior contorno na cópia da imagem
cv.drawContours(imagem_com_contorno, [maior_contorno], -1, (0, 255, 0), 2)

#Plotando a imagem com o contorno da fita
plt.figure(figsize=figsize_default)
plt.title("Imagem com o contorno destacado")
```

```
plt.imshow(imagem_com_contorno)
plt.show()

#Encontrando coordenadas centrais do maior contorno
x, y, w, h = cv.boundingRect(maior_contorno)
x_centro = int(x+(w/2))
y_centro = int(y+(h/2))
print(x_centro)
print(y_centro)

#Plotando imagem com o contorno da fita e centro
plt.figure(figsize=figsize_default)
plt.title("Imagem com o contorno e centro destacados")
plt.imshow(imagem_com_contorno)
plt.plot(x_centro, y_centro, "+")
plt.text(x_centro, y_centro, "({},{})".format(x_centro,y_centro))
plt.show()

#Encontrando e exibindo valores de R,G e B do ponto central da fita
(r,g,b) = img_rgb[y_centro, x_centro]
print("Red: "+str(r)+"\nGreen: "+str(g)+"\nBlue: "+str(b))
```

APÊNDICE B – CÓDIGO PARA ANÁLISE DAS IMAGENS.

```
#Importando bibliotecas
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

#Definindo tamanho padrão das figuras
figsize_default = (10,5)

#Iniciando variáveis onde os dados serão armazenados
redList = []
greenList = []
blueList = []

#Garantindo que as variáveis anteriores estão limpas
redList.clear()
blueList.clear()
greenList.clear()

#Repetição para processar todas as imagens e armazenar dados
for i in range(155, 206):
    #Carregando a imagem utilizada durante o processamento
    caminhoImagem = "C:\\Users\\Otavio\\Desktop\\FotosTCC\\"+str(i)+ ".jpg"
    img_bgr_original = cv.imread(caminhoImagem)
    #Convertendo imagem para diferentes formatos utilizados
    img_rgb = cv.cvtColor(img_bgr_original, cv.COLOR_BGR2RGB)
    img_gray = cv.cvtColor(img_bgr_original, cv.COLOR_BGR2GRAY)
    #Processando imagem
    #Binarizando imagem
    if i < 180:
        _, thresh = cv.threshold(img_gray, 90, 255, cv.THRESH_BINARY)
    else:
        _, thresh = cv.threshold(img_gray, 150, 255, cv.THRESH_BINARY)
    #Detectando bordas
    img_canny = cv.Canny(thresh, 0, 0)
    #Dilate e erode para melhorar a detecção de contornos e reduzir ruídos
```

```
    kernel = np.ones((5, 5))
    img_dilate = cv.dilate(img_canny, kernel, iterations=1)
    img = cv.erode(img_dilate, kernel, iterations=1)
    #Detectando maior contorno após a manipulação da imagem
    contornos, _ = cv.findContours(img, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
    maior_contorno = max(contornos, key=cv.contourArea)
    #Encontrando coordenadas centrais do maior contorno
    x, y, w, h = cv.boundingRect(maior_contorno)
    x_centro = int(x+(w/2))
    y_centro = int(y+(h/2))
    #Armazenando valores de R,G e B do ponto central da fita
    (r,g,b) = img_rgb[y_centro, x_centro]
    redList.append(r)
    greenList.append(g)
    blueList.append(b)

#Plotando o grafico com o resultado final
eixoXgraf = np.arange(-25,26,1)

plt.figure(figsize=figsize_default)
plt.title("Valores de R, G e B para diferentes ângulos")
plt.plot(eixoXgraf,redList,color="red",label="Red")
plt.plot(eixoXgraf,greenList,color="green",label="Green")
plt.plot(eixoXgraf,blueList,color="blue",label="Blue")
plt.xlabel("Posição (°)")
plt.ylabel("Intensidade (-)")
plt.legend(loc="upper left")
plt.show()
```

APÊNDICE C – CÓDIGO PARA CRIAÇÃO DO MODELO DA CNN.

```
#Importando bibliotecas
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import KFold
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras import regularizers
from sklearn.model_selection import train_test_split

#Carregando imagens e rotulando
IMG_SIZE = 128
CAMINHO_IMAGENS = 'C:\\Users\\37108839\\Documents\\Documentos\\0 - TCC\\FotosTCC\\'
dados = []
for nome_arquivo in os.listdir(CAMINHO_IMAGENS):
    if nome_arquivo.endswith('.jpg') or nome_arquivo.endswith('.png'):
        angulo_original = int(nome_arquivo.split('.')[0])
        angulo_convertido = angulo_original - 180
        dados.append({'nome': nome_arquivo, 'angulo': angulo_convertido})
df = pd.DataFrame(dados)

#Separando imagens e rotulos
X = []
y = []
for index, row in df.iterrows():
    img_path = os.path.join(CAMINHO_IMAGENS, row['nome'])
    img = cv2.imread(img_path)
    if img is None:
        continue
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    img = img / 255.0
    X.append(img)
    y.append(row['angulo'])
```

```
#Separando em conjunto de treino e teste
X = np.array(X)
y = np.array(y)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=44
)

#Definindo folds
K = 5
kf = KFold(n_splits=K, shuffle=False)
maes = []
best_mae = float('inf')
melhor_modelo = None
melhor_history = None
fold = 1

#Repetição para cada fold
for train_idx, val_idx in kf.split(X_train):
    print(f"\nTreinando fold {fold}")

    X_train_kf, X_val_kf = X_train[train_idx], X_train[val_idx]
    y_train_kf, y_val_kf = y_train[train_idx], y_train[val_idx]

#Definindo modelo
    model = tf.keras.Sequential([
        tf.keras.layers.Conv2D(16, (3,3), activation='relu', =(IMG_SIZE, IMG_SIZE, 3)),
        tf.keras.layers.MaxPooling2D(2,2),
        tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(64, activation='relu', kernel_regularizer=regularizers.l2(
        tf.keras.layers.Dense(1)
    ])

#Definindo Learning Rate
    optimizerLR = tf.keras.optimizers.Adam(learning_rate=0.001)

#Compilando Modelo
    model.compile(optimizer=optimizerLR, loss='mse', metrics=['mae'])
```

```
#Definindo ReduceLrONPlateau e EarlyStopping
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=10,
    min_lr=1e-6, verbose=0)
    early_stop = EarlyStopping(monitor='val_loss', patience=20,
    restore_best_weights=True, verbose=0)

#Treinando o modelo
    history = model.fit(
        X_train_kf, y_train_kf,
        validation_data=(X_val_kf, y_val_kf),
        epochs=200,
        batch_size=128,
        callbacks=[reduce_lr, early_stop],
        verbose=0
    )

#Avaliando modelo
    loss, mae = model.evaluate(X_test, y_test, verbose=1)
    print(f"Fold {fold} - MAE: {mae}")
    maes.append(mae)

#Salvando melhor modelo
    if mae < best_mae:
        best_mae = mae
        melhor_modelo = model
        melhor_history = history

    fold += 1

#Reportando resultado final
print(f"\nMAE médio após {K} folds: {np.mean(maes)} (±{np.std(maes)})")
print(f"Melhor MAE obtido: {best_mae}")

#Salvando melhor modelo
melhor_modelo.save('modelo_kfold_final.h5')
print("\nMelhor modelo salvo como 'modelo_kfold_final.h5'")

#Plotando curva de aprendizado com melhor fold
```

```
plt.plot(melhor_history.history['loss'], label='Loss (treino)')
plt.plot(melhor_history.history['val_loss'], label='Loss (validação)')
plt.xlabel('Época')
plt.ylabel('Loss (MSE)')
plt.title('Curva de aprendizado (melhor fold)')
plt.legend()
plt.grid()
plt.show()
```

APÊNDICE D – CÓDIGO PARA ANÁLISE DO MODELO.

```
#Importando bibliotecas
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold, train_test_split
from tensorflow.keras.models import load_model
from tensorflow.keras.metrics import mse
from sklearn.metrics import r2_score

#Carregando imagens e modelo
CAMINHO_TESTE = 'C:\\Users\\37108839\\Documents\\Documentos\\0 - TCC\\FotosTCC\\'
CAMINHO_MODELO = 'C:\\Users\\37108839\\Documents\\Documentos\\0 - TCC\\ResultadoFin
IMG_SIZE = 128
modelo = load_model(CAMINHO_MODELO, custom_objects={'mse': mse})
X = []
y = []
for nome_arquivo in os.listdir(CAMINHO_TESTE):
    if nome_arquivo.endswith('.jpg') or nome_arquivo.endswith('.png'):
        angulo_original = int(nome_arquivo.split('.')[0])
        angulo_convertido = angulo_original - 180
        caminho_img = os.path.join(CAMINHO_TESTE, nome_arquivo)
        img = cv2.imread(caminho_img)
        if img is None:
            continue
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        img = img / 255.0
        X.append(img)
        y.append(angulo_convertido)

#Separando em conjunto de treino e teste
X = np.array(X)
y = np.array(y)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=44
)
```

```
#Definindo folds
K=5
kf = KFold(n_splits=K, shuffle=False)
maes = []
best_mae = float('inf')
fold = 1
X_train_kf=[]
X_val_kf=[]
y_train_kf=[]
y_val_kf=[]

#Repetição para cada fold
for train_idx, val_idx in kf.split(X_train):
    print(f"\nTreinando fold {fold}")
    X_train_kf, X_val_kf = X_train[train_idx], X_train[val_idx]
    y_train_kf, y_val_kf = y_train[train_idx], y_train[val_idx]
    #Definindo melhor fold (4)
    if fold == 4:
        X_train_quero = X_train_kf
        X_val_quero = X_val_kf
        y_train_quero = y_train_kf
        y_val_quero = y_val_kf
    fold+=1
print(y_val_quero)

#Realizando estimativas
y_pred = modelo.predict(X).flatten()
y_pred_train_quero = modelo.predict(X_train_quero).flatten()
y_pred_val_quero = modelo.predict(X_val_quero).flatten()
y_pred_test = modelo.predict(X_test).flatten()

#Calculando R2
print(y_test)
print(y_pred_test)
r2 = r2_score(y_test, y_pred_test)
print(f"R²: {r2:.4f}")

#Plotando grafico da dispersão
```

```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred_test, color='green', edgecolors='black',
         label='Modelo (test data)')
plt.plot([min(y), max(y)], [min(y), max(y)], 'k--', label="Ideal")
plt.xlim(-25,25)
plt.ylim(-25,25)
plt.xlabel('Ângulo real (°)', fontsize=12)
plt.ylabel('Ângulo previsto (°)', fontsize=12)
plt.title(f'Dispersão: Ângulo Real vs Modelo\nR² = {r2:.4f}', fontsize=14)
plt.legend()
plt.grid(True)
plt.show()
```