UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

VICTOR HUGO BELINELLO DA SILVA

DESENVOLVIMENTO DE ALGORITMO PARA ALINHAMENTO DE FIBRA ÓTICA COM GUIA DE ONDA EM DISPOSITIVO PLANAR

CURITIBA

2022

VICTOR HUGO BELINELLO DA SILVA

DESENVOLVIMENTO DE ALGORITMO PARA ALINHAMENTO DE FIBRA ÓTICA COM GUIA DE ONDA EM DISPOSITIVO PLANAR

Development of an algorithm for alignment of optical fiber and planar waveguide device

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Alexandre de Almeida Prado Pohl

CURITIBA

2022

VICTOR HUGO BELINELLO DA SILVA

DESENVOLVIMENTO DE ALGORITMO PARA ALINHAMENTO DE FIBRA ÓTICA COM GUIA DE ONDA EM DISPOSITIVO PLANAR

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 21/dezembro/2022

Alexandre de Almeida Prado Pohl
Professor Doutor
Universidade Tecnológica Federal do Paraná

Lucia Valéria Ramos de Arruda
Professora Doutora
Universidade Tecnológica Federal do Paraná

Rodrigo Minetto
Professor Doutor
Universidade Tecnológica Federal do Paraná

CURITIBA
2022

Dedico esse trabalho à minha família, por
permitirem que eu chegasse onde cheguei e
pelo apoio e compreensão durante os,
inúmeros, momentos difíceis.

AGRADECIMENTOS

Agradeço à Fibracem Teleinformática LTDA, por disponibilizar os recursos físicos necessários para montar a plataforma.

Aos meus colegas, pelas sugestões técnicas valiosas durante o desenvolvimento do trabalho.

A todos os professores, que foram fundamentais para a minha formação acadêmica.

Por fim, agradeço à minha família, por me apoiarem durante os momentos difíceis.

RESUMO

O trabalho tem como objetivo desenvolver um algoritmo, com emprego de técnicas de processamento de imagens e aprendizado de máquina, para controlar uma plataforma de alinhamento, cuja finalidade é obter um acoplamento de baixa perda entre fibras óticas e guias de onda em circuitos fotônicos planares, com resolução na ordem de nanômetros, em substituição ao alinhamento manual realizado por um profissional humano. O algoritmo de alinhamento pode ser separado em duas partes, um alinhamento inicial (grosseiro) baseado em imagens, seguido de um alinhamento fino, usando medidores de potência. O foco do trabalho está no alinhamento grosseiro, o qual é essencial para reduzir o espaço de busca para a próxima etapa. Sem essa redução o alinhamento fino pode não ser concluído. Essa etapa do alinhamento é um desafio importante principalmente pelo tamanho reduzido das peças e precisão necessária, que impossibilitam métodos mais simples de processamento de imagens de funcionar adequadamente. O algoritmo de alinhamento fino foi desenvolvido em outro projeto e será utilizado com poucas alterações para operar junto com o algoritmo desenvolvido neste trabalho.

Palavras-chave: processamento digital de imagens; redes óticas passivas; inteligência artificial; redes neurais.

ABSTRACT

This work has the goal of developing an algorithm, using image processing and machine learning techniques, to control an alignment plataform in order to obtain a low coupling loss between optical fibers and waveguides of photonics circuits, with resolution in the order of nanometers, replacing the alignment made by human technicians. The alignment algorithm is split in two parts, an initial alignment (rough estimate) based on images, followed by a more precise alignment, using power meters. The focus of the project is in the initial alignment, which is essential to reduce the search space for the next step. Without this reduction the precise alignment may not finish. This step of alignment is an important challenge mainly because of the reduced size of the components and the precision needed, both of which making it impossible for simple methods of image processing to work properly. The algorithm of precise alignment has been created elsewhere and will be used with just small changes to operate within this work.

Keywords: digital image processing; optics; artificial intelligence; neural networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

CNN	Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks*
DL	Aprendizado Profundo, do inglês *Deep Learning*
DOF	Graus de Liberdade, do inglês *Degree of freedom*
GPU	Unidade de processamento gráfico, do inglês *Graphics Processing Unit*
ML	Aprendizado de Máquina, do inglês *Machine Learning*
NA	Abertura Numérica, do inglês *Numerical Aperture*
PID	Proporcional Integral Derivativo
PLC	Circuitos planares de luz, do inglês *Planar Lightwave Circuits*
PON	Redes óticas passivas, do inglês *Passive Optical Networks*
ReLU	Unidade linear retificada, do inglês *Rectfied Linear Unit*
RGB	Vermelho, Verde e Azul, do inglês *Red, Green and Blue*
SMF	Fibras Monomodo, do inglês *Single-mode fiber*
UTFPR	Universidade Tecnológica Federal do Paraná
YOLO	Você só olha uma vez, do inglês *You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

Em um mundo cada vez mais conectado, com o aumento crescente de dispositivos em rede, a demanda por banda e taxas de transmissão mais elevadas na comunicação cresce de modo semelhante. Tal demanda tem sido suprida por diversas tecnologias e meios de propagação, entre os quais se encontra a fibra ótica. Até há pouco tempo os enlaces e redes óticas eram implantados para atender conexões intercontinentais e metropolitanas ponto-a-ponto, ao passo que a conexão com o cliente final era realizada através de cabo metálico. Entretanto, as Redes óticas passivas, do inglês *Passive Optical Networks* (PON), que utilizam fibras óticas, vem se tornando mais comum no ambiente do usuário final (ANATEL, 2021).

Nesse sentido, para que a distribuição do sinal possa ser realizada em maior escala, há necessidade de emprego de dispositivos passivos com razão de distribuição de sinal até o assinante variando entre 1x2 e 1x64, podendo chegar até mesmo a 1x128. Os divisores óticos de sinal, que realizam tal função, são formados por guias de onda gravados em um substrato de material apropriado e têm sido fabricados em tecnologia planar. Tais dispositivos são conhecidos como Circuitos planares de luz, do inglês *Planar Lightwave Circuits* (PLC) ou, simplesmente, circuitos fotônicos.

Ao mesmo tempo, para realizar o acoplamento físico do sinal entre a fibra e os guias de onda em tais circuitos fotônicos é necessário utilizar uma plataforma que executa o alinhamento de forma automática ou semi-automática, de forma a agilizar e fornecer maior precisão na operação, antes da etapa final de fixação (colagem) definitiva da fibra com o dispositivo planar. É nesse contexto que reside a motivação para esse projeto, em razão de se desenvolver um procedimento controlado por software capaz de se realizar o alinhamento com maior rapidez, repetibilidade e confiabilidade.

Este projeto busca automatizar o sistema de alinhamento através do desenvolvimento de um algoritmo de controle dos atuadores lineares e angulares que fazem parte da plataforma e possibilitam sua movimentação através da aplicação de uma tensão controlada por um algoritmo. Com tal controle é possível um aumento de precisão no processo de alinhamento e também uma redução no tempo para realizar a operação. Assim, o desafio deste projeto se concentra no desenvolvimento do próprio algoritmo, equilibrando eficácia no alinhamento, ou seja baixas perdas entre fibra ótica e o guia de onda do dispositivo planar, com o tempo necessário para o alinhamento.

1.1 Objetivos

Nesta seção são apresentados os objetivos geral e específicos do trabalho, de acordo com o exposto anteriormente.

1.1.1 Objetivo geral

O objetivo geral é desenvolver um algoritmo para controle da plataforma de alinhamento ótico com aplicação de técnicas de processamento de imagem e aprendizado de máquina.

1.1.2 Objetivos específicos

- Estudar e compreender a teoria de acoplamento ótico entre os guias de onda;
- Desenvolver um código para controle dos atuadores e leitura dos sensores;
- Desenvolver um algoritmo para alinhamento baseado em processamento de imagens;
- Aplicar um algoritmo para alinhamento fino baseado em leituras de potência e aprendizado de máquina em conjunto com o algoritmo desenvolvido na etapa anterior.

1.2 Estrutura do trabalho

Este documento está dividido em 4 capítulos. Neste primeiro capítulo o trabalho foi introduzido, apresentando-se a motivação e objetivos do projeto.

No segundo capítulo será apresentada a fundamentação teórica, contendo alguns conceitos essenciais sobre ótica e processamento de imagens. Ao final serão apresentados alguns trabalhos realizados na área e usados durante o desenvolvimento.

No terceiro capítulo a metodologia para alcançar os objetivos será apresentada, juntamente com o cronograma previsto.

No quarto capítulo os resultados do projeto serão relatados e discutidos e possibilidades de trabalhos futuros serão apresentados.

2 REFERENCIAL TEÓRICO

As áreas de conhecimento mais utilizadas nesse trabalho são:

- Ótica;
- Processamento Digital de Imagens;
- Redes Neurais Convolucionais;
- Algoritmos de otimização.

Os tópicos específicos de cada área serão apresentados nas respectivas seções abaixo.

2.1 Ótica

Por se tratar de um algoritmo que irá interagir com um sistema físico que exige precisão, alguns conhecimentos básicos sobre o comportamento de tal sistema são necessários para que sejam considerados no desenvolvimento do software como, por exemplo, correções necessárias por algum fenômeno particular.

2.1.1 Introdução

Quando um raio de luz incide de um meio 1 para um meio 2 ele sofre tanto refração quanto reflexão, como apresentado na Figura 1 para uma incidência em uma interface entre ar e água.

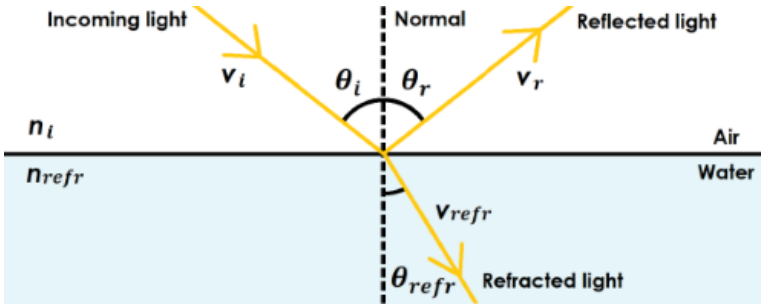


Figura 1 – Refração e reflexão
Fonte: Helen (2022).

O ângulo de refração é expresso pela Lei de Snell-Descartes, expressa pela Equação 1

$$n_1 * \sin\theta_i = n_2 * \sin\theta_r, \tag{1}$$

onde n_1 é o índice de refração do meio 1, n_2 é o índice de refração do meio 2, θ_i é o ângulo de incidência e θ_r é o ângulo de refração.

O índice de refração, por sua vez, é definido como:

$$n = \frac{c}{v}, \tag{2}$$

onde c é a velocidade da luz no vácuo e v é a velocidade no meio.

2.1.2 Reflexão interna total

Quando a luz incide de um meio com índice de refração maior para um meio com índice menor ($n_2 < n_1$) e o ângulo de incidência é suficientemente grande, a Lei de Snell-Descartes (Equação 1) exigiria que o $\sin\theta_r$ fosse maior que 1, o que não é possível. Neste caso particular, o máximo valor que o ângulo de refração pode assumir será 90 graus. Quando esta situação ocorre, o ângulo de incidência é chamado de ângulo crítico e é dado pela Equação 3

$$\theta_{crit} = \arcsin(\frac{n_2}{n_1}). \tag{3}$$

Para qualquer ângulo de incidência maior que o ângulo crítico ocorre o fenômeno de reflexão interna total, onde o raio é totalmente refletido de volta para o meio de incidência. Esse fenômeno é a base para o confinamento de luz em guias de onda dielétricos, como a fibra ótica, e que serão apresentados a seguir.

2.1.3 Guias de onda

Instrumentos óticos convencionais transmitem a luz em forma de feixes, o que exige o uso de lentes e espelhos para focar novamente o feixe, que naturalmente diverge ao se propagar. Contrariamente à propagação no espaço livre, onde ocorre divergência, uma tecnologia foi desenvolvida para transmitir luz por longas distâncias sem a necessidade de lentes, chamada de fibra ótica, que é baseada em ondas guiadas. (SALEH; TEICH, 1991).

De forma geral, um guia de onda ótico é constituído por uma camada ou cilindro de um material dielétrico cercado de outro material dielétrico com um índice de refração menor, conforme ilustrado na Figura 2. Guias de onda com seção transversal retangular são os mais utilizados em circuitos integrados óticos, que são análogos aos circuitos integrados eletrônicos tradicionais. Por outro lado, guias de onda cilíndricos são utilizados para fabricação de fibras óticas, que serão mais detalhadas na subseção 2.1.4.

A propagação de uma onda em um guia é ditada pela geometria e parâmetros elétricos (permissividade elétrica) de meio, sendo que apenas campos eletromagnéticos que satisfazem às condições de contorno são capazes de propagar. Esse campos são chamados de modos do guia de onda e estão ilustrados na Figura 3 para uma guia planar formado por duas pla-

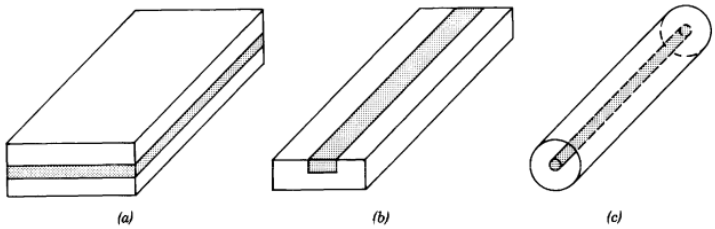


Figura 2 – Guias de onda: a) planar, b) retangular e c) circular enterrado.
Fonte: Saleh e Teich (1991).

cas metálicas paralelas (espelhos), onde podem ser vistos, por exemplo, o comportamento da intensidade do campo elétrico para os modos 1,2,3 e 6.

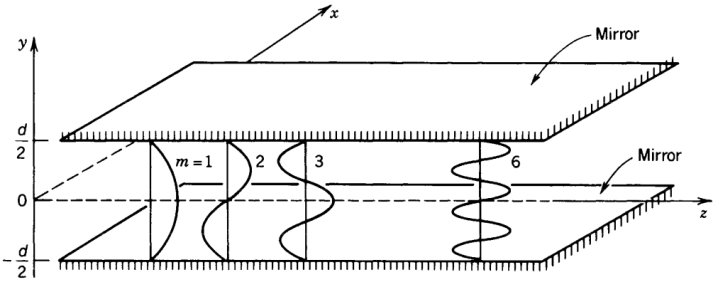


Figura 3 – Modos de um guia de onda planar
Fonte: Saleh e Teich (1991).

Um guia de onda pode ser monomodo, quando apenas um modo se propaga, ou multimodo, que podem ser transmitidos sem se misturar. O número máximo de modos existentes é dado pela Equação 4

$$M = \frac{2d}{\lambda}, \tag{4}$$

onde *d* é a distância de separação entre os espelhos e λ é o comprimento de onda do sinal que se propaga no guia. M assume apenas valores inteiros, então, se $2d/\lambda = 0.9$, M = 0 e se $2d/\lambda = 1.1$, M = 1.

Um modo em particular merece destaque, o modo fundamental, que é o modo com a menor frequência de corte, onde $\lambda_{max} = 2d$. Se $1 < 2d/\lambda \leq 2$, apenas um modo é permitido e o guia de onda é chamado **monomodo**.

Os guias de onda utilizados neste trabalho são guias monomodo do tipo planar, com seção transversal retangular, com o núcleo semelhante à imagem b) da Figura 2

2.1.4 Fibras Óticas

Uma fibra ótica é um guia de onda cilindrico feito de materiais de baixa perda, como o quartzo (SiO2) em alto grau de pureza. A imagem c) na Figura 2 é a representação de uma fibra, em que é possível ver o núcleo envolto por um material (casca) que tem um índice de

refração levemente menor que o núcleo. Na prática, há também uma camada externa, formada por um polímero, envolvendo a estrutura com o objetivo de proteger a fibra. Com as tecnologias atuais, as fibras conseguem transmitir luz com perdas da ordem de 0,2 dB/km (SALEH; TEICH, 1991).

O funcionamento da fibra ótica é semelhante aos guias de onda planares, com a principal diferença sendo sua geometria cilíndrica. Assim como os outros guias, as fibras podem ser monomodo ou multimodo, dependendo do diâmetro do núcleo, da diferença de índice de refração entre o seu núcleo e a casca e do comprimento de onda ultizado para transmitir o sinal.

As fibras multimodo apresentam um problema chamado **dispersão modal** que limita sua capacidade de transmissão. Esse mecanismo pode ser mitigado ao se utilizar uma fibra de perfil de índice de refração gradual, cujo indice de refração varia gradualmente a partir do núcleo até atingir a casca. Os perfis de índice de refração para uma fibra multimodo de perfil degrau e uma fibra com perfil de índice gradual estão ilustrados na Figura 4.

Por outro lado, Fibras Monomodo, do inglês *Single-mode fiber* (SMF), possuem um perfil de índice degrau, mas um diâmetro de núcleo menor. Como no caso de fibras multimodo, o índice de refração é constante em todo o núcleo e muda abruptamente ao atingir a casca. Entretanto, permitem propagar um único modo.

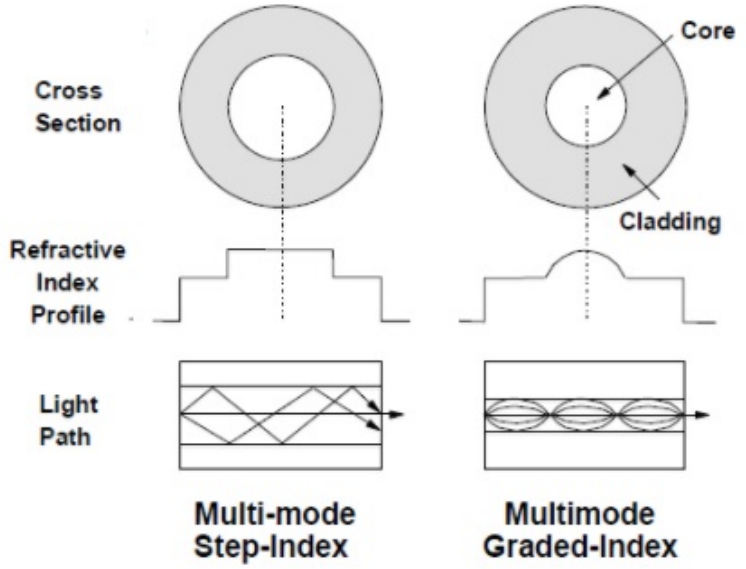


Figura 4 – Perfis de índice de fibras
Fonte: World (2021).

Uma característica importante das fibras é o chamado cone de aceitação de luz. Para que um raio oriundo do ar seja guiado pela fibra ele precisa ter um ângulo de incidência θ_a, na interface entre o ar e a fibra (núcleo), de modo que o ângulo de refração no núcleo, ao se propagar e atingir a interface entre o núcleo e a casca, obedeça à condição do ângulo crítico. Usando a Lei de Snell-Descartes, dada pela (Equação 1), pode-se escrever a Equação 5.

$$\begin{aligned} \theta_a &= \arcsin((n_1^2 - n_2^2)^{1/2}), \\ \theta_a &= \arcsin(NA), \end{aligned} \tag{5}$$

onde o parâmetro NA é conhecido como abertura numérica da fibra. Quanto maior a Abertura Numérica, do inglês *Numerical Aperture* (NA) da fibra, maior sua capacidade de captar e transmitir luz, pois seu cone de aceitação será maior.

A Figura 5 ilustra o cone de aceitação da fibra, a qual mostra que raios de luz dentro do cone de aceitação serão guiados pela fibra.

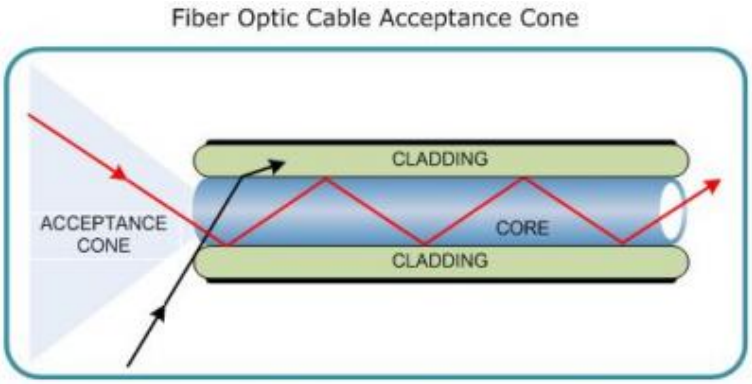


Figura 5 – Cone de aceitação da fibra
Fonte: Optik (2022).

Quando o raio (a) do núcleo da fibra e a NA são suficientemente pequenos, ou o comprimento de onda é suficientemente grande, a fibra é monomodo. A distribuição da intensidade do campo elétrico é descrita por meio das funções de Bessel e funções modificadas de Bessel. A Figura 6 apresenta a distribuição de campo elétrico na seção transversal da fibra para o modo fundamental, que, devido ao seu perfil, pode ser modelado por uma função Gaussiana.

Fibras SMF apresentam menor dispersão, maior velocidade de transmissão, são mais adequadas a aplicações de circuitos integrados óticos, além de não apresentar as interferências e atrasos ocasionados pela interação dos vários modos propagados em uma fibra multimodo. Atualmente, enlaces óticos têm sido instalados apenas com fibras monomodo, onde o problema da dispersão é mais reduzido. Por outro lado, seu tamanho reduzido dificulta o seu alinhamento com guias de onda em dispositivos fotônicos.

A fibra utilizada neste trabalho é monomodo.

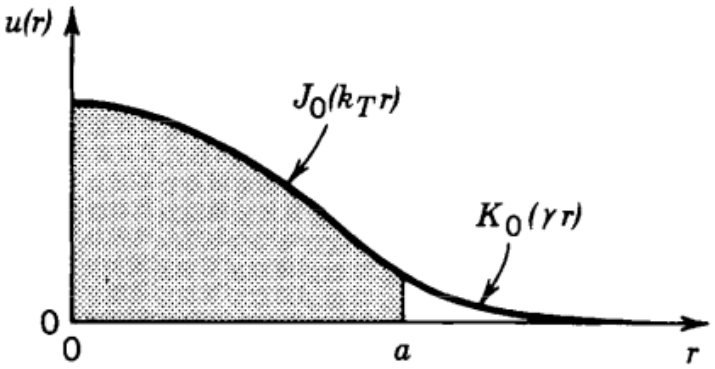


Figura 6 – Distribuição radial do modo fundamental
Fonte: Adaptado de Saleh e Teich (1991).

2.1.5 Alinhamento Ótico

A distribuição da intensidade de campo elétrico à saída de um guia de onda é definida pelo campo modal que depende de sua geometria e dos parâmetros elétricos e de propagação. O campo modal de uma SMF padrão, que segue a recomendação G.652, tem entre 9 e 10 μm, enquanto o campo modal de um guia de onda planar com seção transversal retangular apresenta dimensões que podem variar entre 3 e 9 μm.

Ao se alinhar a fibra e o guia de onda planar existem 6 Graus de Liberdade, do inglês *Degree of freedom* (DOF) a serem considerados, 3 lineares (X,Y,Z) e 3 angulares (α,β,γ), conforme ilustrado na Figura 7.

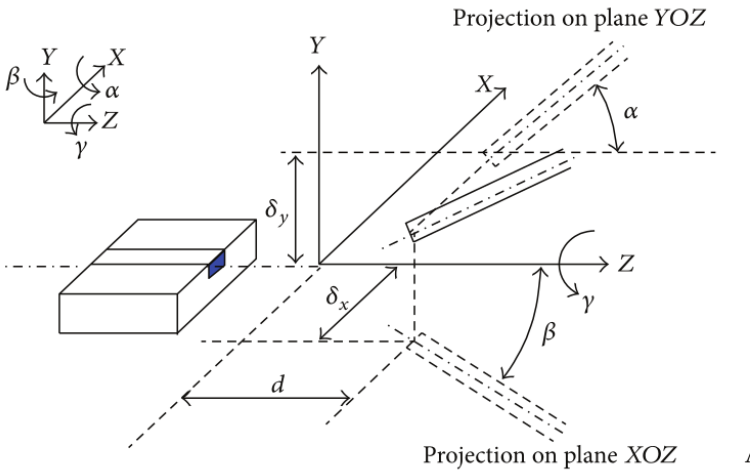


Figura 7 – Alinhamento entre fibra e guia de onda
Fonte: Adaptado de Zheng *et al.* (2013).

A eficiência do acoplamento é afetada ao se deslocar os componentes em qualquer uma das direções e pode ser calculada pela sobreposição dos campos eletromagnéticos (campo modal) da SMF e do guia de onda, conforme descrito pela Equação 6 (ZHENG; DUAN, 2012; ZHENG *et al.*, 2013).

$$\eta = \frac{|\int_A \Psi_f \Psi_w^* dA|^2}{[\int_A \Psi_f \Psi_f^* dA]^2 [\int_A \Psi_w \Psi_w^* dA]^2}, \tag{6}$$

onde A é a área da intersecção dos campos modais do guia de onda e da fibra, Ψ_f a distribuição modal do campo da fibra e Ψ_w a distribuição modal do campo do guia de onda.

Em geral, ambas as distribuições podem ser modeladas como funções Gaussianas. Para a fibra, por ser simétrica, uma curva Gaussiana com o mesmo coeficiente de decaimento pode ser usada para modelar o perfil do campo no eixo horizontal e vertical. Para o guia de onda, no entanto, uma curva Gaussiana distinta para o eixo vertical e duas meias Gaussianas distintas para o eixo vertical precisam ser usadas e combinadas. Tal situação pode ser visualizada na Figura 8.

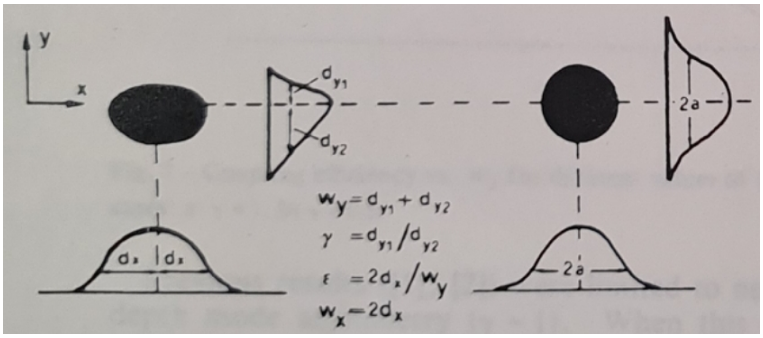


Figura 8 – Perfil modal do guia de onda (esquerda) e fibra monomodo (direita)
Fonte: Adaptado de Irrera (1988).

De acordo com Zheng e Duan (2009), a eficiência de acomplamento é dada pela Equação 7

$$\eta \approx \eta_x * \eta_y, \tag{7}$$

onde

$$\eta_x = k_x \exp\left(-k_x \left[\frac{\delta_x^2}{2}\left(\frac{1}{W_{wa}^2} + \frac{1}{W_f^2}\right) + \frac{\pi^2\beta^2[\omega_{xa}^2(d) + W_f^2]}{2\lambda^2} - \frac{\delta_x \beta d}{W_{wa}^2}\right]\right), \tag{8}$$

$$k_x = \frac{4W_{wa}^2 W_f^2}{(W_{wa}^2 + W_f^2) + \lambda^2 d^2/\pi^2}, \tag{9}$$

$$\omega_{xa}^2(d) = W_{wa}^2\left[1 + \frac{\lambda d}{\pi W_{wa}^2}\right], \tag{10}$$

onde W_f é o raio do campo modal da fibra ótica , W_{wa} e W_{wb} são os raios do campo modal do guia de onda no eixo maior e menor da elipse, respectivamente (na Figura 8, W_x e W_y, respectivamente), α e β são os desvios angulares, d é a distância longitudinal e λ é o comprimento de onda de propagação do sinal. Trocando x por y a fórmula para η_y pode ser obtida. A Figura 9

apresenta um gráfico da eficiência de acoplamento em X. Os valores utilizados foram $d = 0$, $W_{wa} = 0.8\mu m$, $W_f = 8\mu m$. O comportamento é semelhante para a direção Y.

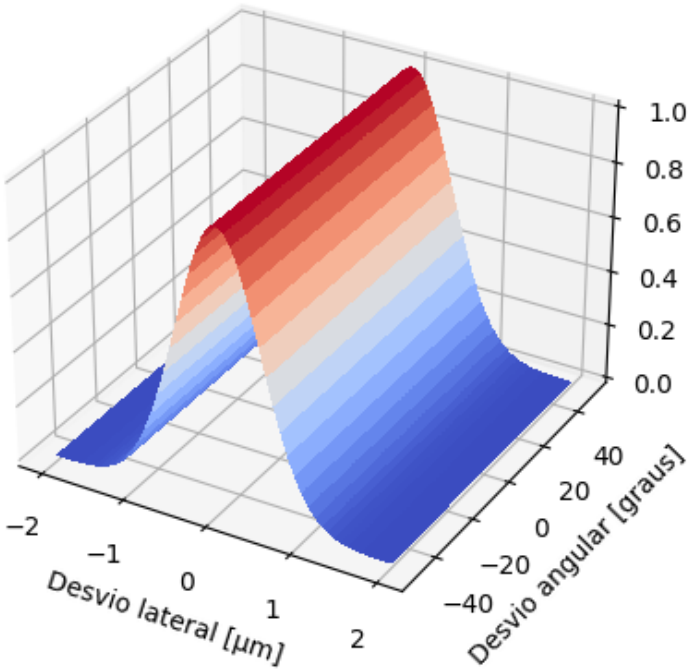


Figura 9 – Gráfico da eficiência de acoplamento em X
Fonte: Autoria própria (2022).

Neste trabalho será realizado o acoplamento entre fibra e divisor ótico. É relevante então esclarecer como esse divisor ótico se apresenta internamente. Na Figura 10 é possível ver um esquemático de tal dispositivo.

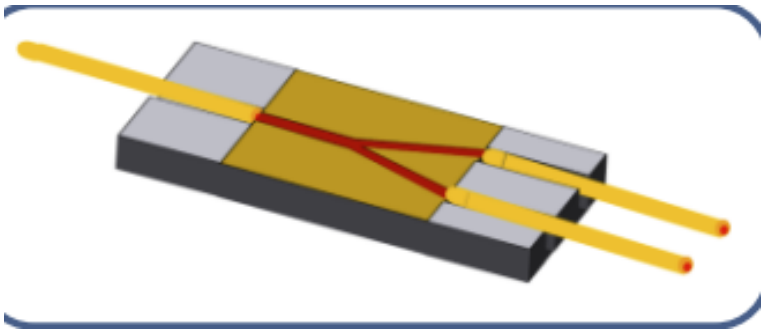


Figura 10 – Esquema de divisor ótico
Fonte: Adaptador de Ryu *et al.* (2011).

2.2 Processamento Digital de Imagens

Como já apresentado, a primeira parte do algoritmo consiste em utilizar imagens para alcançar um primeiro alinhamento, dito grosseiro. Alguns tópicos explorados no desenvolvimento do algoritmo são detalhados a seguir.

2.2.1 Imagens

O primeiro conceito a ser entendido trata do que é uma imagem, na perspectiva computacional. Essencialmente, uma imagem é uma matriz 2D de elementos quadrados chamados pixels (GONZALEZ; WOODS, 2006). Cada pixel é um elemento (x,y) na matriz, semelhante ao plano cartesiano, mas em geral define-se que o eixo Y cresce para baixo em uma imagem.

Em uma imagem em escala de cinza cada pixel tem um valor entre 0 (preto) a 255 (branco). Nas imagens coloridas existem 3 valores que variam nesse mesmo intervalo, o resultado da combinação desses três valores gera a cor final. A cor exata depende do espaço de cores utilizado.

O espaço de cores mais comum é o Vermelho, Verde e Azul, do inglês *Red, Green and Blue* (RGB), onde uma cor é obtida pela soma de níveis de três fontes distintas, a saber: Vermelho, Verde e Azul, como apresentando na Figura 11.

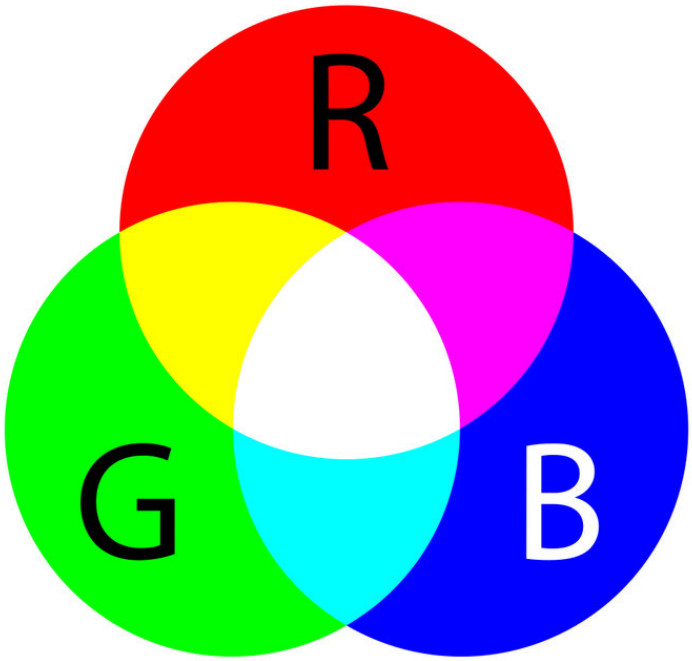


Figura 11 – Espaço de cores RGB
Fonte: Hisour (2022).

2.2.2 Filtros

Um filtro é uma operação matemática. Essa operação é chamada de convolução. Matematicamente a convolução é dada pela Equação 11:

$$g(x,y) = \omega * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) f(x+dx, y+dy), \tag{11}$$

onde g(x,y) será a imagem filtrada, f(x,y) é a imagem original e ω é o kernel da convolução. Essa matriz(kernel) é o que define o resultado final. Por exemplo, se a imagem ficará borrada, realçada, etc. Em geral, a e b tem o mesmo valor e são ímpares. Tal equação, apesar de inicialmente complexa, indica que um pixel é a soma ponderada de seus vizinhos, onde os pesos vem do kernel.

2.2.3 Binarização

Essa transformação é uma operação realizada pixel a pixel, ou seja, o valor final de um pixel depende apenas do valor inicial do mesmo pixel.

A binarização consiste em transformar uma imagem em escala de cinza em uma imagem preta e branca. Alguns algoritmos exigem uma imagem preta e branca ou são mais simples em determinado cenário. A Equação 12 que expressa esse processo é:

$$f(x,y) = \begin{cases} 255 & \text{se } f(x,y) > \alpha, \\ 0 & \text{caso contrário} \end{cases} \tag{12}$$

onde α é um limiar escolhido, arbitrariamente ou automaticamente em alguns algoritmos. Uma pequena variação da binarização é utilizada quando existem variações de iluminação na imagem, por exemplo um flash da câmera. É a chamada binarização local ou adaptativa (GONZALEZ; WOODS, 2006), que olha a região ao redor do pixel para determinar o limiar, ao invés de um limiar global.

2.2.4 Detector de bordas de Canny

Uma tarefa muito importante neste trabalho é encontrar os guias de onda, o método utilizado para isso foi a detecção de bordas, usando o detector de borda de Canny (GONZALEZ; WOODS, 2006). O operador de Canny é um processo de várias etapas *multi-stage* com os seguintes passos:

1. Aplicação de filtro Gaussiano para redução de ruído

2. Cálculo dos Gradientes 2D
3. Remoção de não máximos
4. Limitação por histerese

2.2.4.1 Redução de ruído

A detecção de bordas é um procedimento suscetível a ruídos, por isso a imagem é convoluída com uma Gaussiana, efetivamente borrando levemente a imagem. Esse filtro é preferido ao filtro da média, que também reduz ruídos, pois ele preserva melhor as bordas das imagens. A função Gaussiana 1D com média zero e desvio padrão σ está apresentada na Equação 13.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \tag{13}$$

Na Figura 12 é apresentado o resultado da aplicação do filtro Gaussiano em uma imagem, que consiste em aplicar a convolução na horizontal e, em seguida, na vertical, nesse caso utilizando um kernel de tamanho 5x5, o que implica em $\sigma = 1$.

Figura 12 – Exemplo de convolução Gaussiana
Fonte: OpenCV (2022b).

2.2.4.2 Gradiente

Os gradientes são calculados utilizando-se o kernel de Sobel, que calcula diferenças finitas. Para gradientes horizontais o kernel é $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$, para os verticais $\begin{bmatrix} . -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & 2 & 1 \end{bmatrix}$
Com isso tem-se as magnitudes e os ângulos do gradiente de cada pixel, dados pelas Equa-

ção 14 e Equação 15, respectivamente.

$$M = \sqrt{G_x^2 + G_y^2}, \tag{14}$$

$$A = \arctan(\frac{G_y}{G_x}). \tag{15}$$

2.2.4.3 Filtragem de não máximos

Esse passo é uma filtragem de pixels desnecessários nas bordas, pois o algoritmo deve ter bordas finas como resultado. A imagem inteira é percorrida com os seguintes procedimentos:

1. Dado um pixel, tomar a direção do gradiente;
2. Pegar os vizinhos na direção do gradiente;
3. Se o pixel for um máximo local entre seus vizinhos, mantenha seu valor;
4. Caso contrário, zere o pixel.

Ao final desse procedimento obtém-se uma imagem que contém as bordas afinadas.

2.2.4.4 Histerese

No procedimento dois limites são escolhidos, um limite inferior e um superior. Os pixels são filtrados baseados no valor da intensidade do gradiente.

- Pixels com valor superior ao limite superior são considerados bordas certas;
- Pixels com valor abaixo do limite inferior são descartadaos como ruído;
- Pixels com valor entre os limites, mas conectados a uma borda certa, são mantidos;
- Pixels com valor entre os limites, mas sem conexão são descartados.

2.2.5 Morfologia

Operações morfológicas são operações não lineares que afetam as características da imagem (GONZALEZ; WOODS, 2006), por exemplo, erodindo ou dilatando. Tais operações são feitas sobre imagens binarizadas. Elas também contam com um kernel, no entanto a procedimento para “aplicar” o kernel é diferente. Primeiramente, esse kernel é composto apenas dos valores 0 e 1. A aplicação é semelhante aos operadores lógicos “OU” e “E”.

A operação de erosão, por exemplo, pode ser vista como um “E” lógico entre a imagem e o kernel. Se o pixel tiver um valor em ambos ele se mantém com o valor, caso contrário o

valor do pixel é 0. A operação de dilatação é um “OU” lógico, se o pixel estiver no kernel ou na imagem original ele é definido como 1 (ou 255 dependendo da escala).

Na Figura 13 pode-se ver o exemplo da dilatação seguida de uma erosão, operação chamada de fechamento, preenchendo um buraco interno na imagem, potencialmente um ruído, mostrando uma das utilidades dessas operações. Nessa imagem o kernel utilizado foi

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figura 13 – Dilatação seguida de erosão
Fonte: Academy (2022).

2.2.6 Transformada de Hough

A transformada de Hough é uma técnica para extração de características para encontrar objetos dentro de uma forma geométrica, originalmente linhas, através de um procedimento de votação (GONZALEZ; WOODS, 2006). Neste trabalho a variante da transformada que encontra linhas é utilizada e por isso receberá detalhamento a seguir.

Uma reta pode ser descrita de várias formas, mas para a transformada de Hough a Equação 16 é escolhida por ter parâmetros limitados, o que é conveniente computacionalmente, pois é fácil de implementar.

$$x \cos\theta + y \sin\theta = r. \tag{16}$$

Na equação acima *r* é o comprimento da normal a partir da origem até a reta e θ é a orientação de *r* em relação ao eixo x, conforme ilustrado na Figura 14.

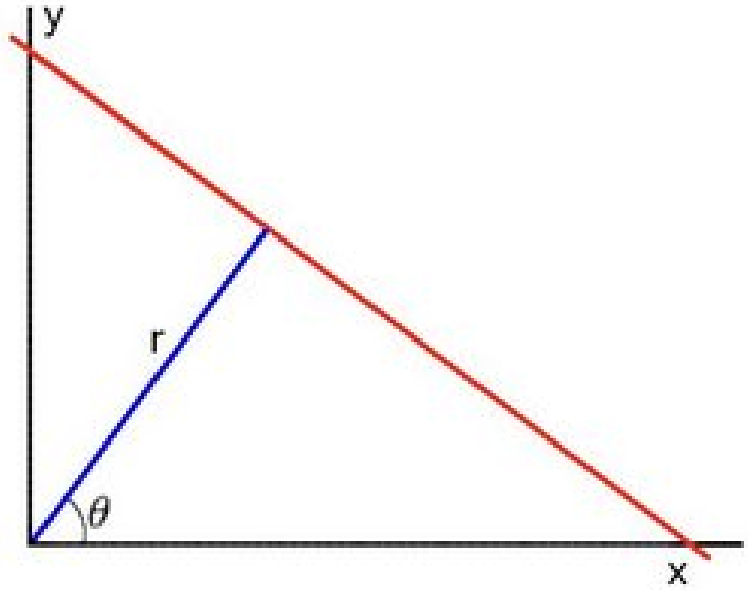


Figura 14 – Sistema polar de coordenadas.
Fonte: OpenCV (2022a).

Cada ponto específico da imagem (x_i,y_i) irá gerar uma curva senoidal no plano polar. A intersecção dessas senoides será o ponto com os parâmetros *r*, θ da reta que passa pelos pontos da imagem original. Vale ressaltar que várias intersecções podem ocorrer e geralmente um limiar(*t*) é definido. Assim, uma linha é detectada tomando os pontos com, no mínimo, *t* intersecções entre as curvas. Por exemplo, na Figura 15, 3 pontos distintos geraram 3 curvas, que se interceptam no ponto (0.925, 9.6), o que significa que a reta com $r = 9.6$ e $\theta = 0.925$ passa por esses 3 pontos.

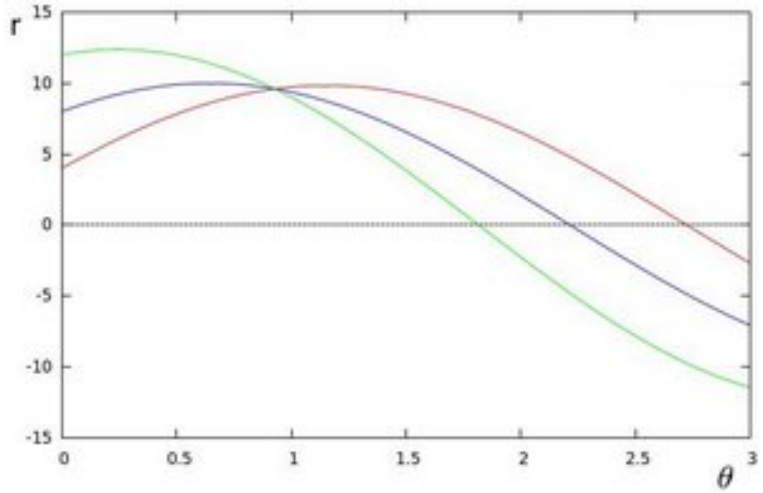


Figura 15 – Curvas geradas por 3 pontos distintos.
Fonte: OpenCV (2022a).

2.3 Redes Neurais Convolucionais

Na área de reconhecimento de imagens, atualmente os melhores resultados costumam ser obtidos por Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks* (CNN), que podem ser vistos como uma evolução das Redes Neurais do tipo Perceptron [1].

A imagem de entrada passa por esses algoritmos de Aprendizado Profundo, do inglês *Deep Learning* (DL) através de várias camadas, conforme ilustrado na Figura 16.

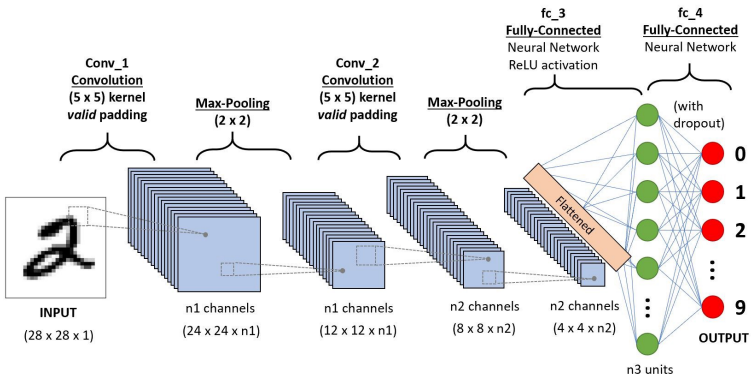


Figura 16 – Rede neural convolucional para classificar digitos
Fonte: Saha (2022).

[1] https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron

2.3.1 Camada de convolução

Essa camada consiste em realizar a convolução entre a imagem e o kernel, assim como foi explicado na subseção 2.2.2. Essa é uma das diferenças mais marcantes entre CNN e as redes neurais do tipo Multi-perceptron, pois nas CNN o pixel está conectado apenas com vizinhos, enquanto que em outras redes neurais todos os pixels estão totalmente conectados.

Esse passo extrai características de alto nível da imagem como bordas, por exemplo. Ele também pode reduzir a dimensionalidade da imagem, o que acelera o processamento.

Caso a imagem de entrada seja aumentada antes da convolução, sua dimensão é mantida após essa etapa. Esse procedimento é chamado de *Same Padding*. Por outro lado, se a imagem de entrada for mantida, a saída terá sua dimensão reduzida. Nesse caso o procedimento é chamado *Valid Padding*. A Figura 17 apresenta esses dois procedimentos.

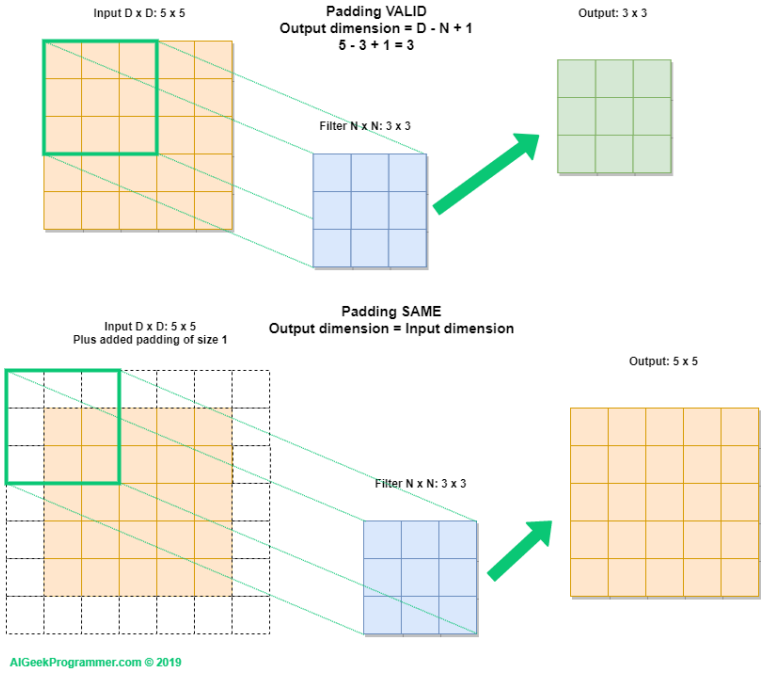


Figura 17 – Técnicas de preenchimento
Fonte: Programmer (2022).

2.3.2 Treinamento

As CNN, assim como qualquer outra rede neural artificial, precisam ser treinadas previamente. O objetivo principal é fazer com que a rede aprenda os pesos para o *kernel*.

Esse procedimento, apesar de essencial para a performance prática da rede, é simples. Manualmente várias imagens são anotadas, identificando-se os objetos de interesse.

A forma de anotar é específica de cada rede, mas em geral, para cada imagem I tem-se N linhas do tipo $(classe, regiao)$, onde N é o número de objetos de interesse na imagem I, $classe$ é a classe do objeto (gato, cão, carro etc) e $regiao$ define o retângulo que delimita

o objeto (*Bounding Box*). Esse último dado é necessário apenas quando a rede irá detectar o objeto, não apenas classificá-lo.

2.3.3 Camada de Pooling

A camada de *Pooling* tem como objetivo principal reduzir a dimensionalidade da imagem para aliviar o processamento, mas também extrai características dominantes da imagem, que se mantém mesmo com variações de translação e rotação (GOODFELLOW; BENGIO; COURVILLE, 2016).

Existem vários tipos de *pooling*, entre eles o *max pooling* e o *average pooling*, que são muito comuns.

O primeiro simplesmente toma o maior valor na região do *kernel*, enquanto o segundo, como o nome implica, faz a média de todos os pontos. Ambos estão ilustrados na Figura 18.

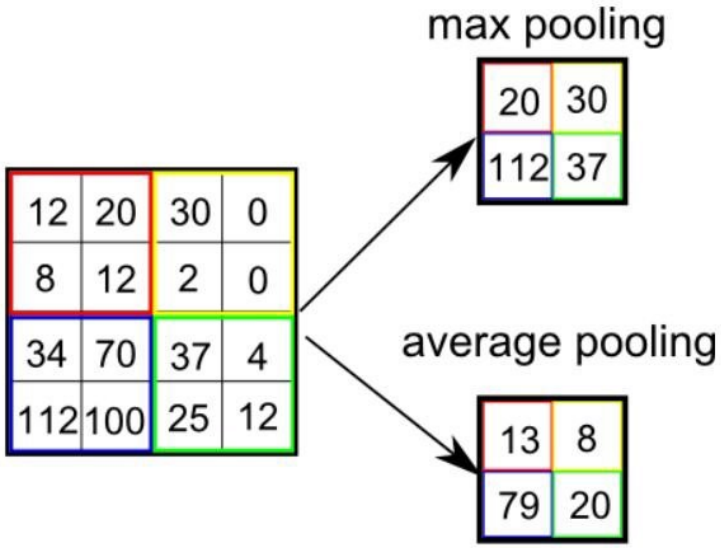


Figura 18 – Técnicas de agrupamentos
Fonte: Saha (2022).

2.3.4 Função de ativação

Após uma camada convolucional, geralmente uma função de ativação é utilizada para introduzir não-linearidade no modelo, permitindo modelos mais complexos e úteis.

As funções e seus gráficos estão apresentados na Figura 19, a mais comum e recomendada é a Unidade linear retificada, do inglês *Rectfied Linear Unit* (ReLU).

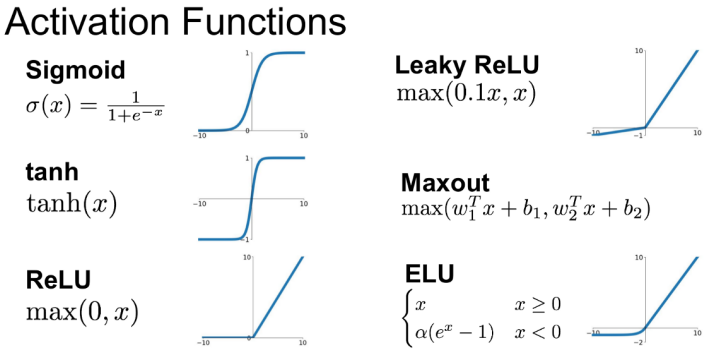


Figura 19 – Diversas funções de ativação
Fonte: Jadon (2018).

2.4 Algoritmos de otimização

A segunda etapa do alinhamento entre os guias de onda consiste em utilizar algoritmos de otimização. Esses algoritmos têm como objetivo maximizar (ou minimizar) uma função com N variáveis. Para esse trabalho a posição de cada motor é uma variável e a função a ser otimizada é a perda de potência ótica conforme a Equação 17

$$F(X,Y,Z,RX,RY,RZ) = Perda \tag{17}$$

Em geral os algoritmos buscam minimizar a função, então para isso basta multiplicar o resultado por -1, pois a perda é negativa e queremos a menor perda (em valor absoluto).

Alguns dos principais algoritmos utilizados na área são

- Busca espiral (Algoritmo heurístico)
- Busca Simplex (Pesquisa operacional)
- Hill climbing (IA clássica)
- Pattern Search (Metaheurística de trajetória)
- Algoritmos genéticos (Metaheurística evolucionária)

Cada um com vantagens e desvatagens (ZHENG; DUAN, 2012). Os algoritmos utilizados serão brevemente descritos a seguir.

2.4.1 Busca espiral

Esse algoritmo é o mais simples dos utilizados, ele é utilizado como a primeira etapa no alinhamento fino. O procedimento consiste em fazer uma espiral buscando o melhor ponto. Neste trabalho seria o ponto com a menor perda.

Primeiramente é necessário escolher o plano onde a espiral será realizada, por exemplo o plano da face das peças, formado pelos eixos X e Y na Figura 7. Feito isto, basta seguir os passos descritos no Algoritmo 1. Um ponto relevante no algoritmo é que a espiral é aplicada em um eixo de cada vez, ou seja, é na verdade empregada uma espiral quadrada.

Algoritmo 1 – Algoritmo de busca espiral

inserir $DIAMETRO\_FINAL$
inserir $TAMANHO\_PASSO$
1: $dir \leftarrow 1$
2: $diametro \leftarrow 1$
3: **enquanto** $diametro < DIAMETRO\_FINAL$ **faça**
4: $passo \leftarrow TAMANHO\_PASSO * dir$
5: **para** d in diametro **faça**
6: Move motor X $passo$
7: Guarda perda medida
8: **finaliza para**
9: **para** d in diametro **faça**
10: Move motor Y $passo$
11: Guarda perda medida
12: **finaliza para**
13: $diametro \leftarrow diametro + 1$
14: $dir \leftarrow dir * -1$
15: **finaliza enquanto**
16: Vai para ponto com melhor perda

Fonte: Autoria própria (2022).

2.4.2 Busca Simplex

Esse algoritmo, originalmente proposto por (NELDER; MEAD, 1965), consiste em utilizar um *simplex*, uma forma que tem $n + 1$ vértices em n dimensões, para encontrar um politopo. Usando os vértices do *simplex* atual o algoritmo decide qual o próximo melhor *simplex*. Os passo para o caso de n = 2 estão descritos no Algoritmo 2. A convergência pode ser calculada de várias formas, nesse trabalho o algoritmo simplesmente para depois de N execuções, onde N é configurado no código, mas em geral 50 execuções são permitidas.

2.4.3 Algoritmos genéticos

Essa família de algoritmos se baseia na biologia evolutiva, onde os melhores indivíduos sobrevivem. De maneira geral eles seguem o fluxograma da Figura 20

Os principais componentes desses algoritmos são:

- População inicial de indivíduos (conjunto de posições inicias dos motores)

Algoritmo 2 – Algoritmo de busca simplex

inserir u, v, w
inserir fn
$convergiu \leftarrow false$
enquanto $convergiu \neq true$ **faça**
    Calcula fn nos 3 pontos
    Ordena os pontos {fn(u) < fn(v) < fn(w)}
    $r \leftarrow$ Reflexao w através do centróide de u e v
    se fn(u) < fn(r) < fn(v) **então**
        $w \leftarrow r$
    finaliza se
    se fn(r) < fn(u) **então**
        $e \leftarrow$ Extende r (adiciona $\overrightarrow{wr}$)
        se fn(e) < fn(u) **então**
            $w \leftarrow e$
        senão,
            $w \leftarrow r$
        finaliza se
    senão,
        Contrai $\overrightarrow{wr}$ em dois pontos
        $c_i \leftarrow \frac{1}{4}\overrightarrow{wr}$
        $c_o \leftarrow \frac{3}{4}\overrightarrow{wr}$
        se fn(c_i) < v **então**
            $w \leftarrow c_i$
        senão, se fn(c_o) < v **então**
            $w \leftarrow c_o$
        senão,
            Encolhe o simplex na direção de u
            $w \leftarrow \frac{1}{2}\overrightarrow{wu}$
            $r \leftarrow \frac{1}{2}\overrightarrow{ru}$
        finaliza se
    finaliza se
    Calcula convergência
finaliza enquanto

Fonte: Autoria própria (2022).

- Avaliação dos indivíduos (medição das perdas de potência para cada conjunto de posições)
- Seleção dos melhores indivíduos baseado em probabilidades (quanto menor a perda mais provável aquele conjunto de posições ser escolhido)
- Combinação dos pais em um ou mais filhos (combina posições diferentes para gerar um novo conjunto de posições)
- Mutação para introduzir variação na população (muda alguma posição do conjunto baseado em uma probabilidade baixa)

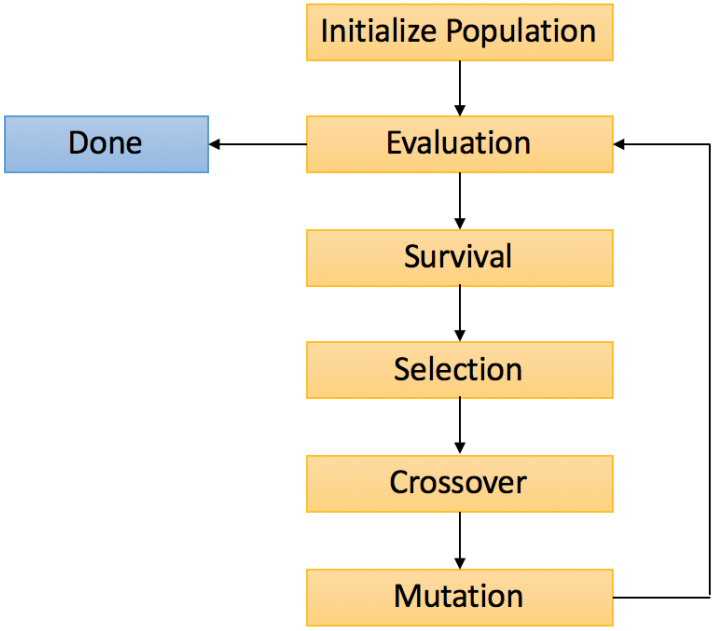


Figura 20 – Fluxograma algoritmos genéticos
Fonte: Blank e Deb (2020).

3 MATERIAIS E MÉTODOS

A seguir são descritos os materiais, tecnologias, métodos e ferramentas utilizados para atingir os objetivos propostos na seção 1.1.

3.1 Materiais e Tecnologias

3.1.1 Python

Para escrever o algoritmo foi utilizada a linguagem de programação Python. Essa linguagem possui uma sintaxe simples o que atrai iniciantes e desenvolvedores, permtindo-se focar na solução do problema ao invés da linguagem.

A linguagem Python vem se tornando cada vez mais popular, em particular, na área de processamento de imagens e Aprendizado de Máquina, do inglês *Machine Learning* (ML). Isso se deve principalmente por suas bibliotecas intuitivas, como OpenCV e Pytorch.

3.1.2 OpenCV

OpenCV[1] é uma biblioteca *open source* de visão computacional e ML que conta com vários algoritmos com códigos otimizados com foco em aplicações de visão em tempo real.

Ela possui interfaces com diversas linguagens, entre elas Python, mas por ser escrita nativamente em C++ apresenta uma performance superior a um código nativo em Python. Combinando, então, alta performance, sintaxe simplificada do Python e uma grande comunidade, essa biblioteca é a escolha padrão para aplicações de visão computacional em Python.

3.1.3 Git e Github

Git é o sistema de versionamento de código mais utilizado atualmente. Com ele é possível gerenciar diferentes versões do mesmo código de maneira simples, permitindo desenvolver um trecho de código de teste de maneira separada e, ao final do desenvolvimento, combinar as códigos ou descartar as mudanças.

Github é a plataforma de hospedagem do código, seu propósito é permitir facilmente a colaboração, além de servir também como uma espécie de backup do código (apesar de não ser o objetivo principal da plataforma).

[1] https://opencv.org/

3.1.4 Yolo

YOLO é um dos algoritmos de detecção e reconhecimento de padrões mais famosos do mundo por ser rápido e preciso. O algoritmo divide a imagem em um *grid* e cada célula é resposável por detectar objetos dentro de si.

O nome vem do fato do algoritmo ser baseado em uma única CNN que olha a imagem apenas uma vez, Você só olha uma vez, do inglês *You Only Look Once* (YOLO). O YOLOV5 (JOCHER, 2020) foi desenvolvido usando a biblioteca PyTorch, o que levou a melhorias de performance em relação as versões anteriores. Ele é composto de 3 partes principais, a saber:

- Backbone - Responsável por extrair as características
- Neck - Responsável por agregar as características
- Head - Responsável por gerar os retângulos delimitadores e definir a classe e confiança

Um diagrama da arquitetura do YOLOV5 pode ser visto na Figura 21.

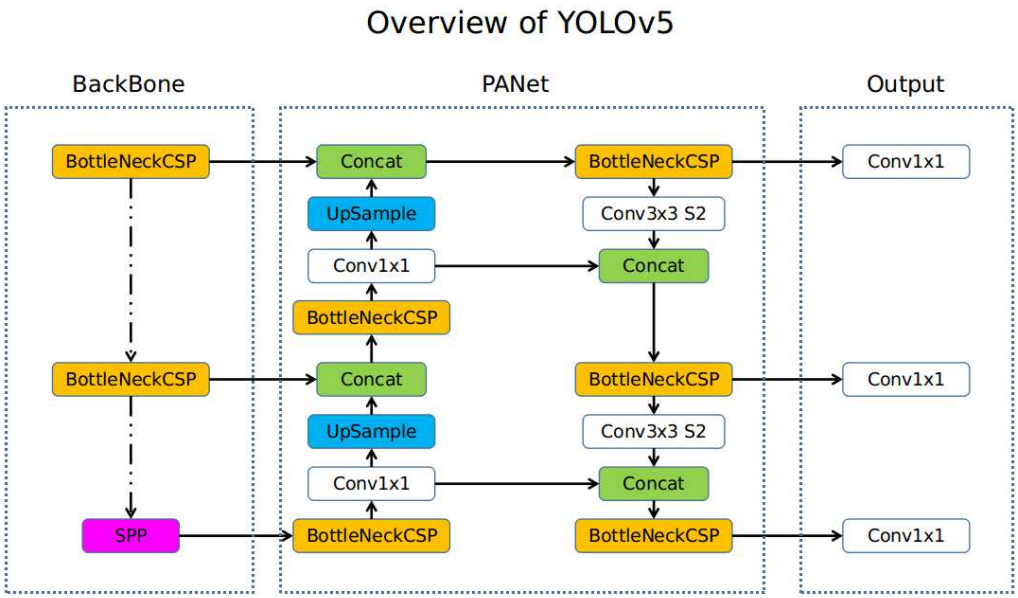


Figura 21 – Arquitetura do YOLOv5
Fonte: seekFire (2020).

3.1.5 Roboflow

Roboflow é uma plataforma completa para desenvolver modelos de visão computacional. Nela é possível fazer o *upload* do *dataset*, anotar as imagens no próprio navegador, aplicar técnicas de *data augmentation*, que na versão gratuita chegam a triplicar o tamanho do *dataset*,

dividir o *dataset* automaticamente e até mesmo treinar o modelo. Além de prover uma biblioteca Python que facilita obter o *dataset* diretamente da plataforma.

3.1.6 Google Colab

O Google Colab, ou *Colaboratory*, permite a execução de código Python diretamente no navegador, sem precisar de qualquer configuração. Além disso os notebooks, como são chamados esses ambientes, tem acesso a Unidade de processamento gráfico, do inglês *Graphics Processing Unit* (GPU) de maneira gratuita permitindo um treinamento muito mais rápido e sem necessidade de configuração alguma para poder utilizá-la, sendo esse o principal motivo para escolha do Colab nesse trabalho.

3.1.7 Whimsical

Whimsical é uma ferramenta web que possibilita desenvolver, de maneira colaborativa, documentos e diagramas de maneira fácil, sendo possível desenvolver fluxogramas, escrever textos ou criar diagramas. Alguns exemplos são Figura 22 e Figura 29

3.1.8 Plataforma de Alinhamento

Essa seção descreve brevemente os componentes da plataforma física que é controlada pelo algoritmo, além de simulada.

A Figura 22 mostra um diagrama simplificado do esquema que representa a plataforma com seus principais componentes.

3.1.8.1 Bases motorizadas

As bases motorizadas são compostas de 6 motores da Thorlabs[2] cada, 3 para translação e 3 para rotação, na entrada e saída da plataforma, respectivamente. Esses estágios de translação e rotação, por sua vez, estão conectados a um controlador Proporcional Integral Derivativo (PID), esse controlador já faz parte do hardware obtido. A Tabela 1 lista os principais componentes das bases. A Figura 23 mostra uma foto das bases motorizadas montadas na plataforma de alinhamento.

[2] https://www.thorlabs.com/

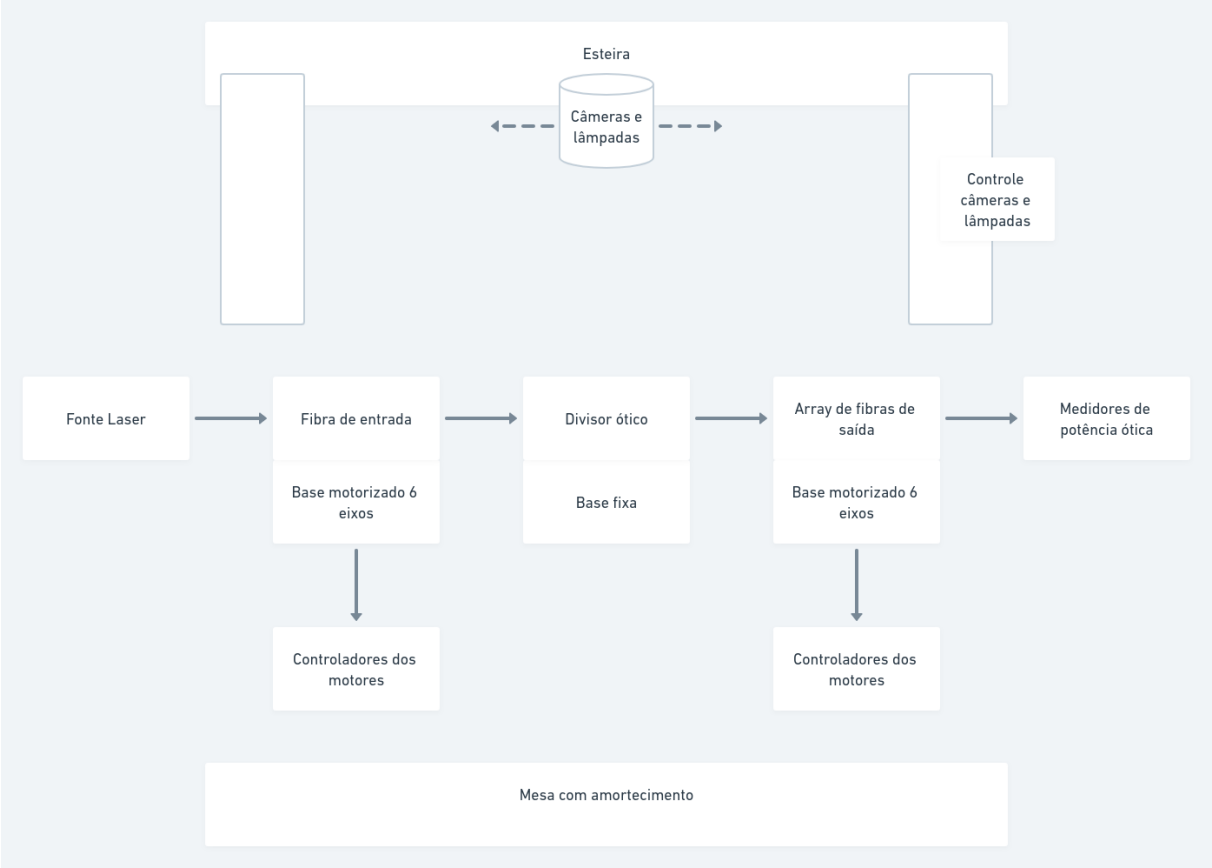


Figura 22 – Diagrama plataforma
Fonte: Autoria própria (2022).

Tabela 1 – Componentes das bases motorizadas

Item	Quantidade	Descrição
Z806	4	Servomotor
ZST213B	6	Motor de passos
Z812	2	Servomotor
TDC0001	4	Controlador servomotor
TST1010	6	Controlador motor de passos
TCH002	2	Controlador servomotor
KCH601	2	Hub controlador USB
CR1/M-Z7E	2	Estágio de rotação
MT3/M	2	Estágio de translação
PT1B/M	3	Estágio de translação

Fonte: Autoria própria (2022).

3.1.8.2 Peças

As peças posicionadas nas bases motorizadas não são os guias de ondas em si, os guias estão localizados dentro das peças. Na Figura 24, Figura 25 e Figura 26 a posição dos guias de onda estão indicados com setas verdes.

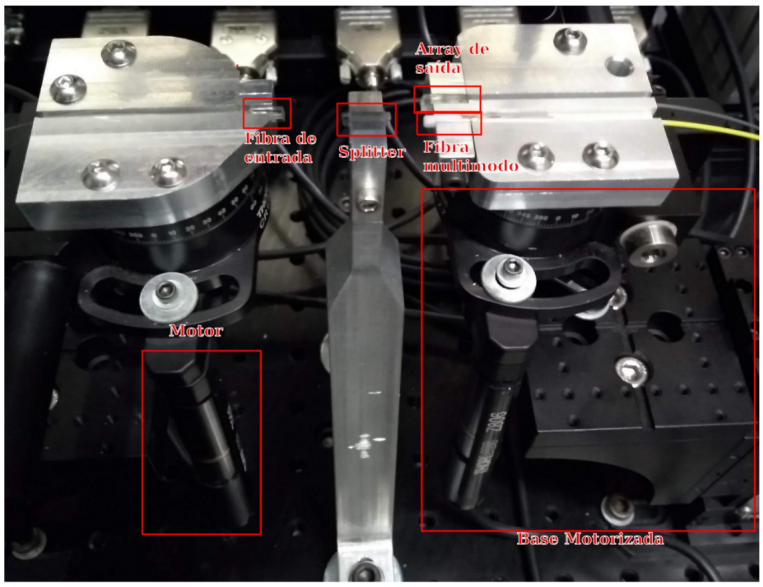


Figura 23 – Base motorizada
Fonte: Nicolas Abril.

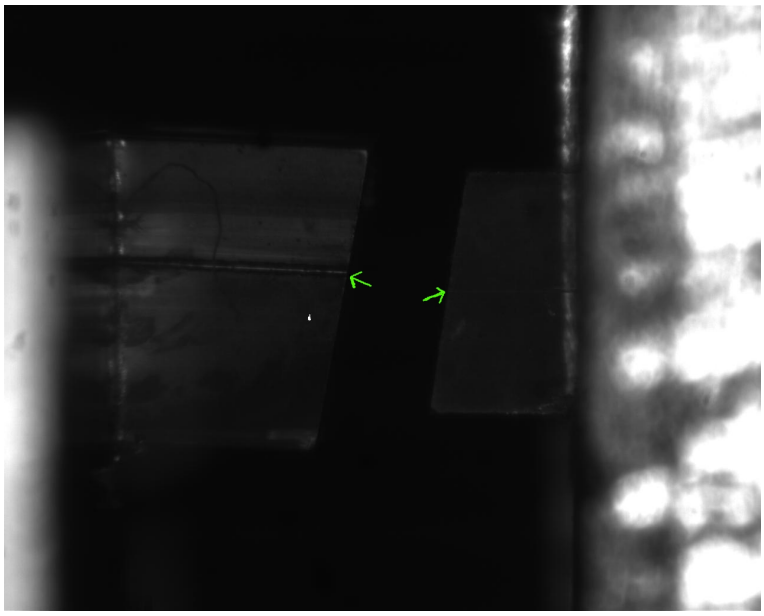


Figura 24 – Destaque guias de onda horizontal
Fonte: Autoria própria (2022).

Vale ressaltar aqui que as peças possuem faces inclinadas, isso pode ser visto nas imagens horizontais, por exemplo Figura 24, o ângulo que pode ser visto em ambas as peças não é um desalinhamento (apesar de potencialmente existir um) elas são produzidas desse modo, essa inclinação ajuda na hora de posicionar as peças nas bases motorizadas, caso uma delas esteja virada ao contrário as peças não poderiam ser conectadas perfeitamente.

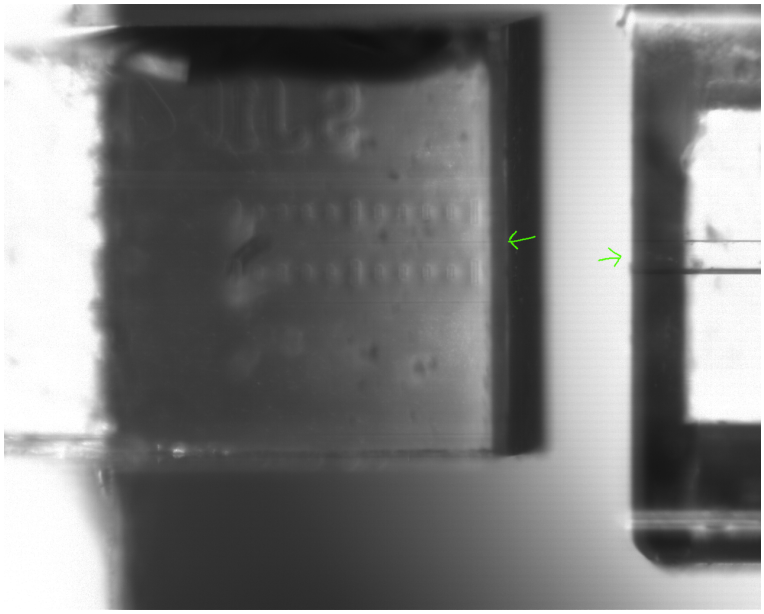


Figura 25 – Destaque guias de onda vertical entrada
Fonte: Autoria própria (2022).

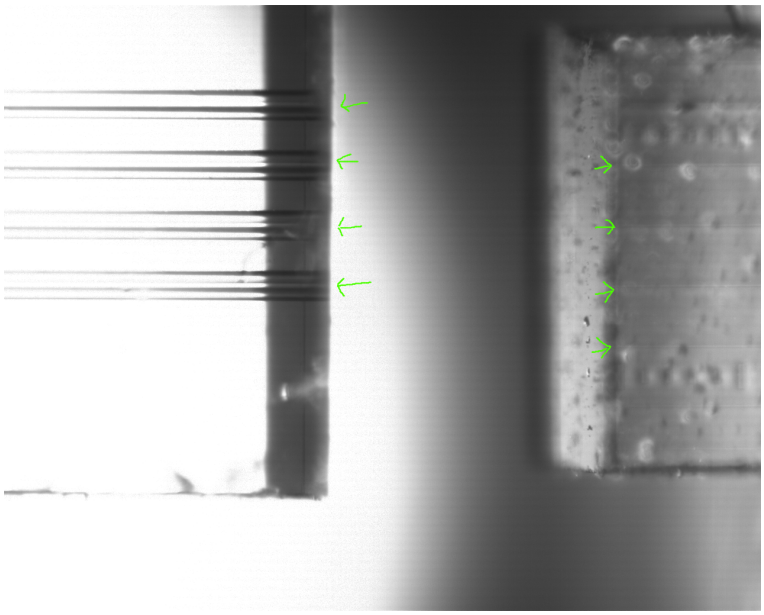


Figura 26 – Destaque guias de onda vertical saída
Fonte: Autoria própria (2022).

3.1.8.3 Câmeras

A plataforma possui 2 câmeras **DCC1545M** da Thorlabs. Uma tem uma visão superior das peças, chamada de câmera vertical, e a outra tem uma visão lateral das peças, chamada de câmera horizontal.

Essas câmeras estão presas a uma esteira que por sua vez está conectada em um motor de passos, permitindo mover em um eixo horizontal o conjunto de câmeras pela plataforma. Isto torna possível capturar imagens da entrada do sistema e da saída, mas apenas uma de cada

vez. Assim, não é possível ter todo o sistema capturado em uma única imagem. A Figura 27 mostra uma foto do sistema de câmeras montado na plataforma.

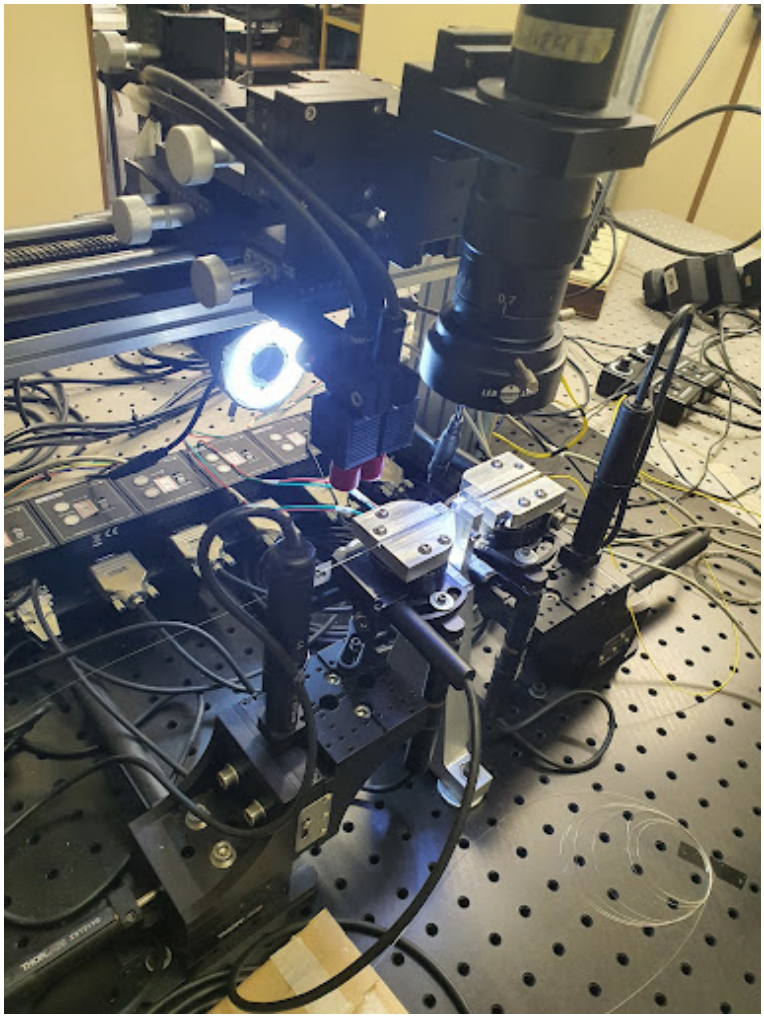


Figura 27 – Visão das câmeras vertical (topo da imagem) e horizontal (na esquerda da imagem)
Fonte: Autoria própria (2022).

3.1.8.4 Fonte de luz e medidores de potência

Na entrada da plataforma é utilizada uma fonte de luz com comprimento de onda $\lambda = 1550nm$ e potência nominal de 4.2 mW. A potência, no entando, varia levemente no decorrer do tempo na ordem de $50\mu W$, o que deve ser considerado ao desenvolver o algoritmo. Na saída da plataforma a potência ótica é medida por dois medidores **PM 100USB** da Thorlabs. Um medidor é acoplado na fibra mais à esquerda e outro na fibra mais à direita, podendo realizar a leitura das respectivas potências óticas e confirmar se o algoritmo está alinhando todas as fibras existentes e não apenas uma saída. A Figura 28 mostra uma imagem do desalinhamento entre o guia planar e o array de fibras para um divisor 1x8. Contudo, neste trabalho é utilizado um divisor 1x4.

Figura 28 – Desalinhamento em RX
Fonte: Abril (2020).

3.2 Métodos

A metodologia consistiu em realizar, primeiramente, uma revisão bibliográfica dos assuntos relacionados ao escopo do trabalho. Assim, foram estudados fundamentos de ótica, processamento de imagens e redes neurais. Em seguida, foi desenvolvido e implementado o alinhamento grosseiro de forma iterativa, ou seja, desenvolvendo, testando e corrigindo em ciclos. Em etapa subsequente foi realizada a adaptação do código já existente para o alinhamento fino e, por fim, a escrita do documento. Essa sequência está ilustrada na Figura 29.

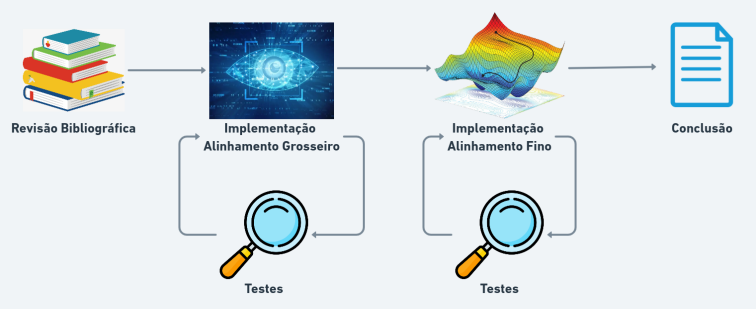


Figura 29 – Metodologia utilizada
Fonte: Autoria própria (2022).

3.2.1 Revisão bibliográfica

Nessa etapa do projeto foram revisados os conceitos necessários para implementar o código, sendo que alguns foram estudados posteriormente, pois não estavam nos planos originais de implementação. Este foi o caso da Redes Neurais Convolucionais (seção 2.3).

3.2.2 Implementação do alinhamento grosseiro

Essa etapa foi a mais complexa e demorada. Nela foi utilizado um processo iterativo, desenvolvendo-se pequenas soluções e testando frequentemente. O objetivo era diminuir o espaço de busca para a próxima etapa de alinhamento (subseção 3.2.5). Essencialmente consiste em aproximar as peças o suficiente para que o próximo algoritmo, que trata do alinhamento fino, consiga executar o processo em tempo hábil.

3.2.2.1 Iteração 1

Inicialmente foi proposto utilizar unicamente algoritmos de visão computacional clássica (seção 2.2). Assim, foi utilizado o detector de Canny, seguido da transformada de Hough de forma que as linhas de contorno das peças pudessem ser encontradas.

Esse processo pode ser observado na Figura 30. Em seguida a transformada de Hough é aplicada, como pode ser visto na Figura 31. Usando alguns dos parâmetros disponíveis para a transformada no OpenCV é possível filtrar as linhas baseado na ângulação. O resultado é mostrado na Figura 32.

Os resultados mostraram que esse procedimento não era robusto o suficiente para as imagens da câmera vertical, não sendo possível obter as linhas nas imagens verticais de maneira consistente. Por outro lado, os resultados foram satisfatórios no processamento das imagens obtidas com a câmera horizontal.

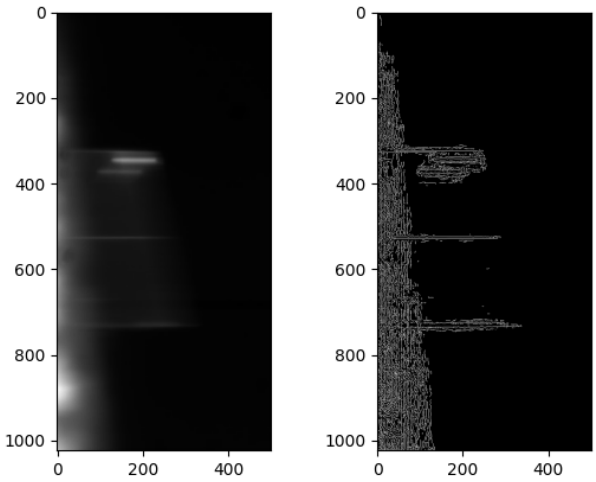


Figura 30 – Resultado após aplicar detector de Canny na imagem horizontal
Fonte: Autoria própria (2022).

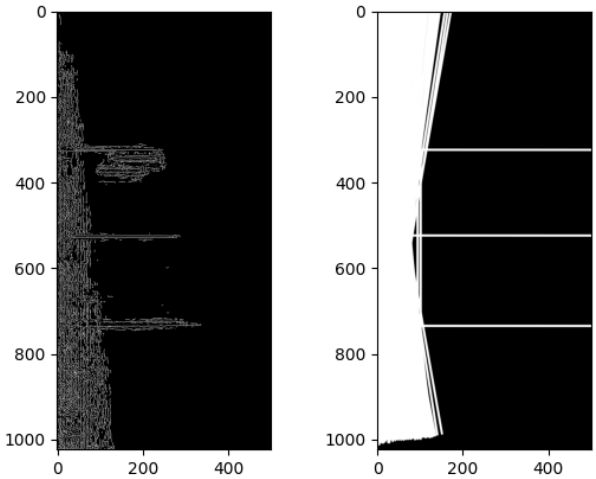


Figura 31 – Resultado após aplicar a transformada de Hough
Fonte: Autoria própria (2022).

3.2.2.2 Iteração 2

Decidiu-se, então, adotar uma abordagem de ML para identificar as peças. Para isso foi utilizada a versão 5 do algoritmo YOLO.

Para essa solução foi necessário criar um *dataset* com imagens das peças e usá-lo para treinar a rede. No decorrer do projeto várias fotos foram tiradas das peças, cerca de 2000 imagens, no entanto a maioria é muito semelhante ou mesmo inútil para o treinamento.

Por exemplo, algumas imagens foram obtidas durante um teste para encontrar a luminosidade ideal. Isso foi feito passando por todas as combinações possíveis, sendo que em muitas delas não é possível ver as peças por problemas de luminosidade, como ilustrado na Figura 33.

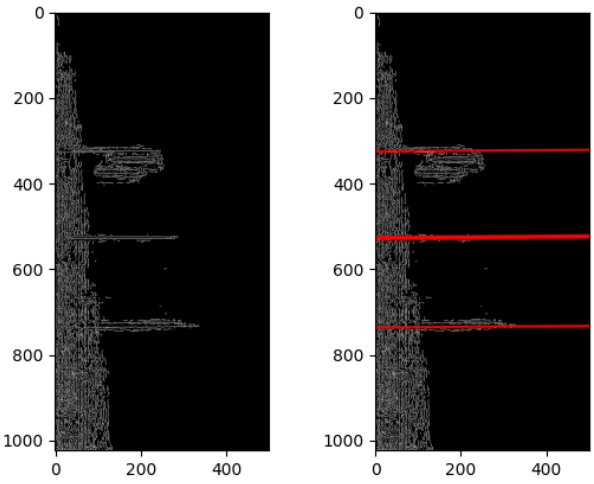


Figura 32 – Resultado após filtrar a saída da transformada
Fonte: Autoria própria (2022).

Ou, em outra situação, as imagens são muito parecidas, como mostrado na Figura 34, e não acrescentam valor significativo ao treinamento, podendo até mesmo piorar o desempenho para imagens, cujas peças não são vistas.

Filtrando-se apenas imagens adequadas para o treino foram selecionadas um total de 135 imagens, o que, para os padrões de redes atuais, é muito pouco. Para melhorar o conjunto de imagens para o treinamento, técnicas de *data augmentation* foram aplicadas. Isto foi feito automaticamente na plataforma Roboflow. Exemplos da técnica de *data augmentation* são:

- Adição de ruído;
- Adição de *blur*;
- Alteração de brilho e exposição.

Após a aplicação das técnicas descritas foram criados dois *datasets*. O primeiro com 285 imagens para treino, 15 imagens para validação e 17 para teste. O segundo conjunto foi criado aplicando-se a técnica de *data augmentation* no primeiro, resultando em 759 imagens para treino, 32 imagens para validação e 32 imagens para teste.

As imagens foram classificadas em 2 câmeras x 2 peças x 2 lados da plataforma = 8 classes. A Figura 35 apresenta alguns exemplos de imagens do *dataset* (o ruído visível foi adicionado pela etapa de *data augmentation*). A Figura 36 mostra uma imagem anotada, cuja informação sobre a classe está omitida na mesma.

(a) Muito escura

(b) Muito clara

Figura 33 – Duas imagens com problemas de luminosidade
Fonte: Autoria própria (2022).

A rede foi treinada com diferentes parâmetros no Google Colab, que conta com uma GPU Nvidia Tesla T4. A Tabela 2 apresenta os parâmetros testados e o resultado. A YOLO contém muitos parâmetros, apenas os que foram alterados estão apresentados, nos restantes foi mantido o valor padrão.

Vale ressaltar que as 2 últimas execuções foram interrompidas prematuramente, de acordo com o parâmetro *Early stopping*, que para o treinamento quando 100 épocas passam e não existe evolução.

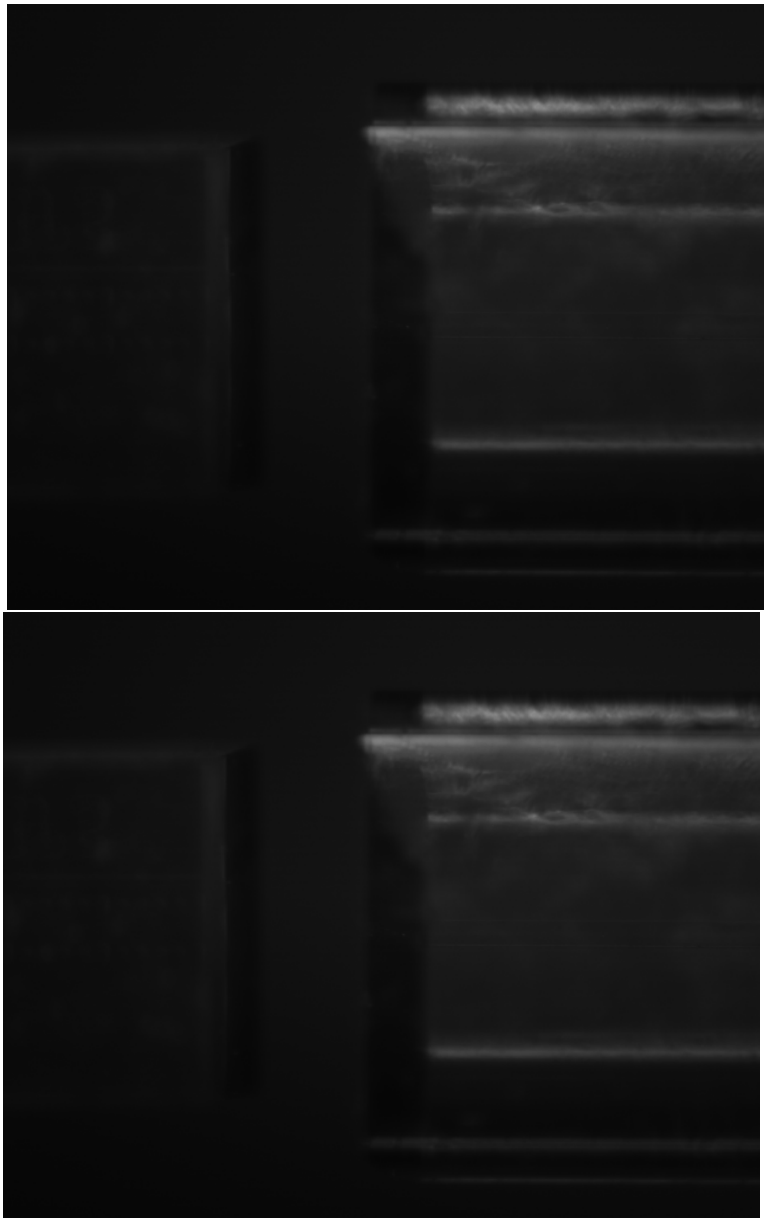


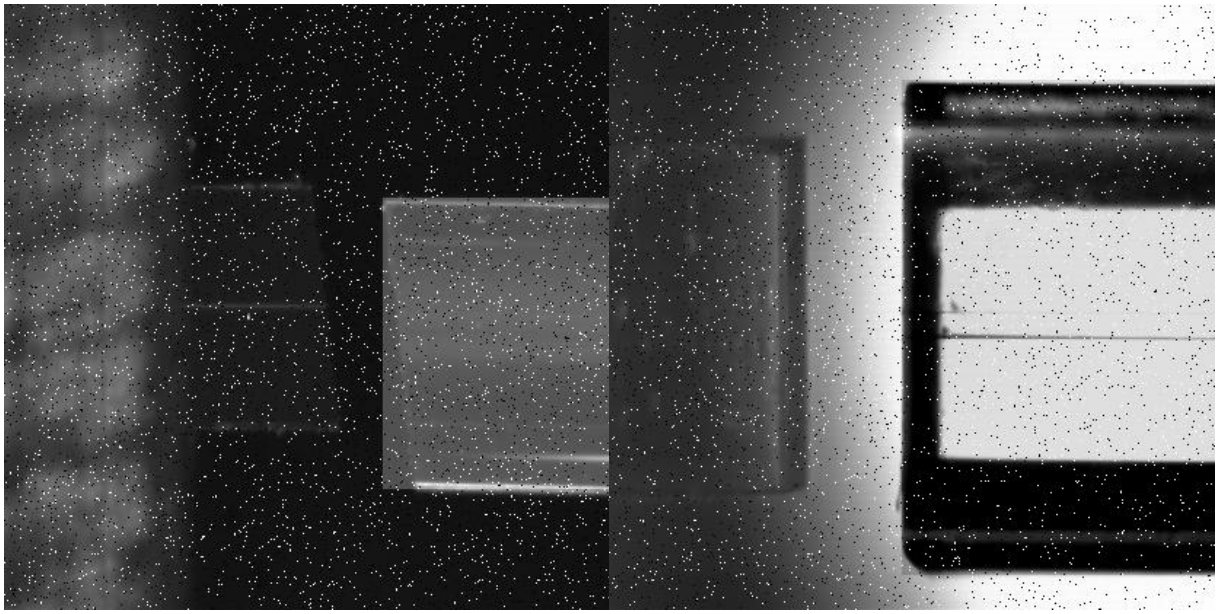
Figura 34 – Duas imagens muito parecidas
Fonte: Autoria própria (2022).

Sobre a última execução, nela as 10 primeiras camadas (que formam o *backbone* da rede YOLO) foram travadas/congeladas. Assim, apenas as camadas finais foram treinadas. Esse procedimento é chamado *transfer learning*[3] e é útil quando o tamanho do *dataset* é pequeno

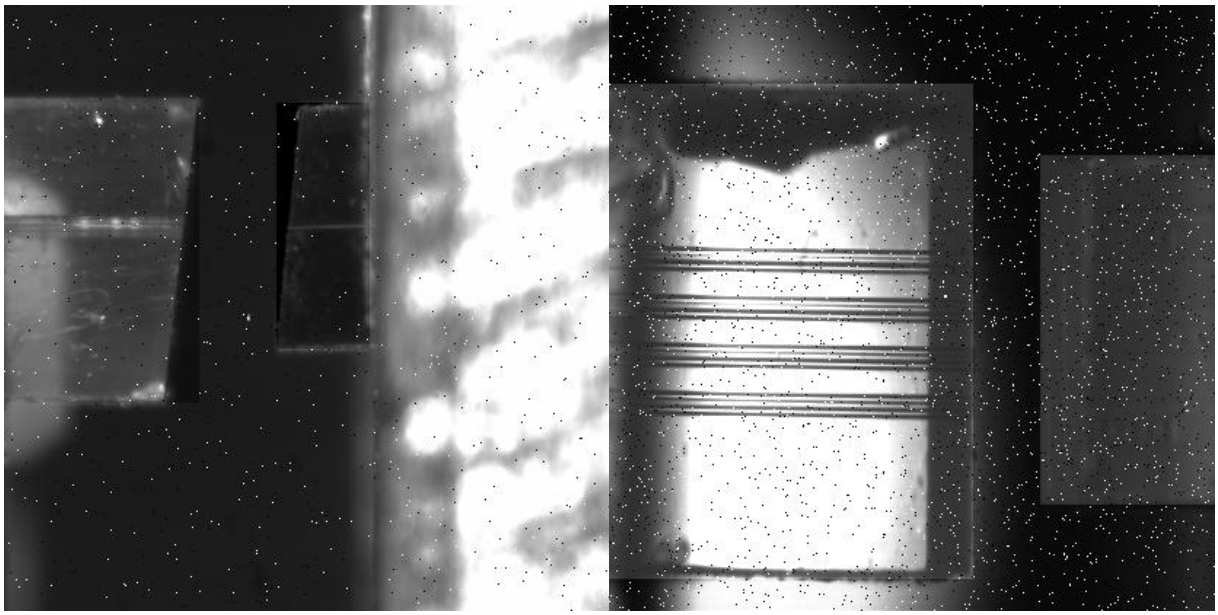
O primeiro *dataset*, que tem menos imagens, é identificado como **1** na tabela, o segundo *dataset* é identificado como **2**. O segundo *dataset* teve resultado melhor, como era esperado, por isso apenas 1 teste foi realizado com o *dataset* **1**.

A execução 3, onde o *batch size* foi alterado, não apresentou mudança de perfomance, o que é condizente com a documentação da YOLO[4]. No entanto, o tempo de treinamento foi

[3] https://github.com/ultralytics/yolov5/issues/1314
[4] https://github.com/ultralytics/yolov5/discussions/2452



(a) Horizontal Entrada **(b) Vertical Entrada**



(c) Horizontal Saída **(d) Vertical Saída**

Figura 35 – Exemplos de imagens do conjunto de dados

consideravelmente maior, 29 minutos contra 20 minutos utilizando *batch size* = 16. Por isso o valor utilizado posteriormente foi 16.

Na execução 4 o resultado caiu significativamente, muito possivelmente devido ao fenômeno de *overfitting*. A rede aprende a identificar muito bem somente as imagens de treinamento, mas não consegue detectar corretamente imagens novas.

A partir da execução 6 os pesos bases utilizados foram trocados para utilizar os pesos da rede **yolov5x**, ao invés da **yolov5s**, que todas as execuções anteriores utilizaram. O resultado

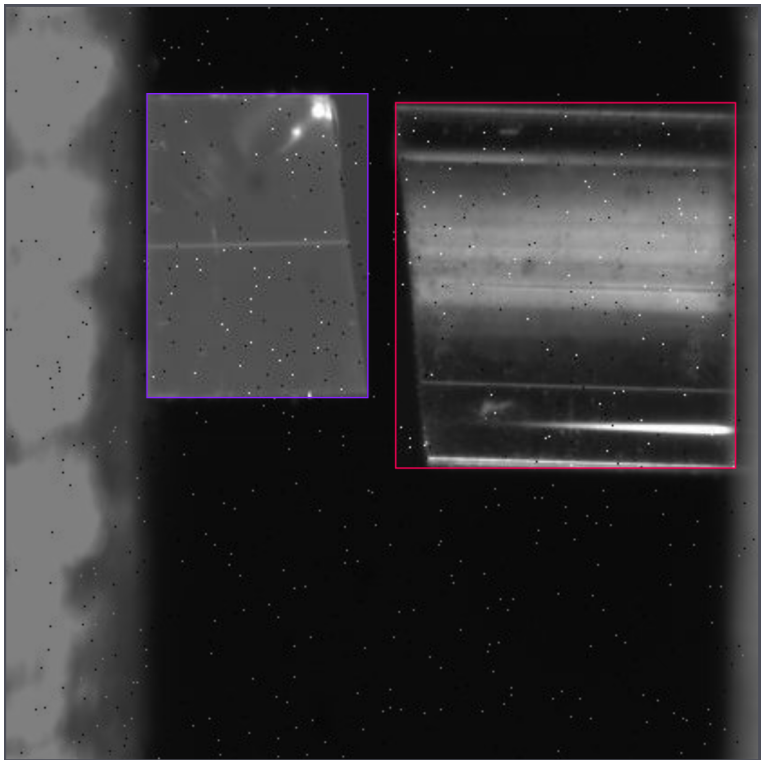


Figura 36 – Imagem Horizontal Entrada anotada
Fonte: Autoria própria (2022).

Tabela 2 – Diversos parâmetros utilizados para treinamento

Dataset	Épocas	Batch size	Layers congelados	Percentual de detecção	Gráfico
1	300	16	0	56.25%	fancy-frog-6
2	300	16	0	62.5%	unique-brook-8
2	300	32	0	62.5%	fallen-blaze-9
2	1000	16	0	59.5%	worldly-microwave-10
2	1000	16	10	78.2%	feasible-haze-11
2	1000	16	10(yolov5x)	87.5%	polished-brook-13
2	1000	16	10(yolov5x)	78.2%	laced-smoke-16

Fonte: Autoria própria (2022).

foi melhor, apesar de demorar significativamente mais tempo para treinar e inferir. No entanto, isso não é um problema para este trabalho.

Na última execução o dataset foi dividido novamente para aumentar o conjunto de validação. A divisão passou a ser 663 imagens para treino, 64 imagens para validação e 32 imagens para teste. O percentual de detecção foi obtido rodando o modelo no conjunto de teste, a detecção é considerada como bem sucedida se ambas as peças forem encontradas na imagem.

A Figura 37 apresenta a evolução da *mAP*[5] durante o treinamento. É possível observar que todas finalizaram acima de 0.9, logo nenhuma conclusão pode ser tirada dessa imagem.

[5] https://blog.paperspace.com/mean-average-precision/

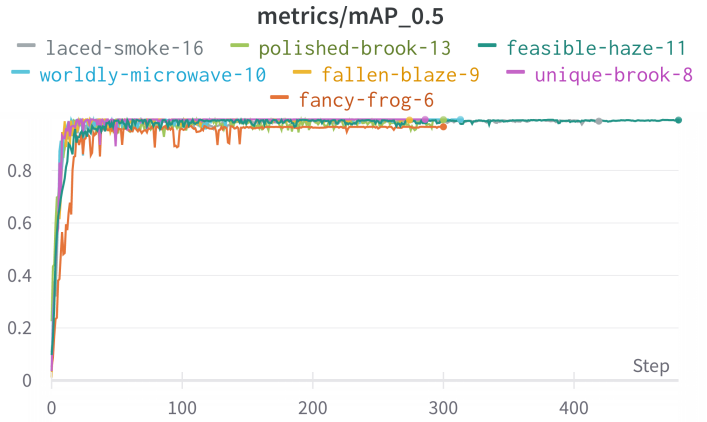


Figura 37 – Evolução da mAP das execuções
Fonte: Autoria própria (2022).

No entanto a Figura 38 apresenta a evolução das métricas de *obj_loss* (confiança da presença de um objeto na imagem) e a métrica *cls_loss* (confiança na classe inferida) apenas das 3 últimas execuções, para melhor visualização.

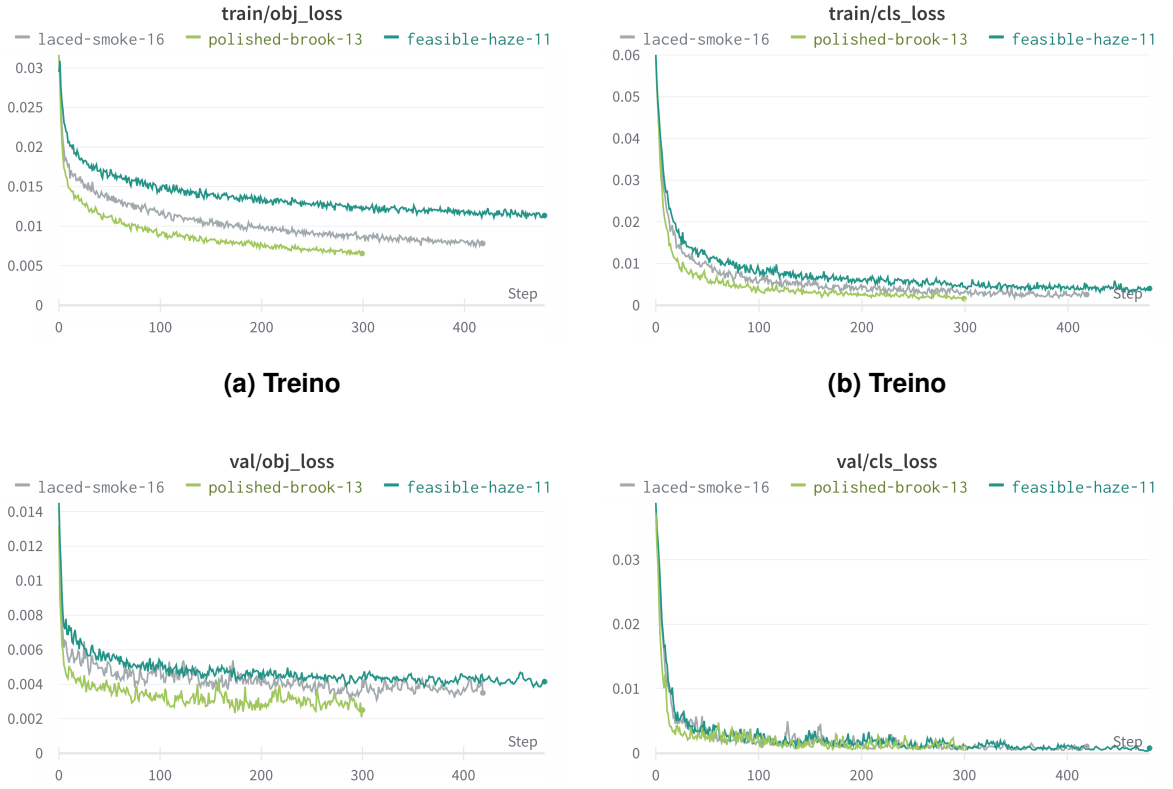
Pelos gráficos é possível verificar que a melhor execução em relação à detecção do objeto foi a penúltima (*polished-brook-13*) (ver Tabela 2). Em relação à classificação, todas performaram de maneira semelhante. Com esses resultados o modelo da penúltima execução foi escolhido para ser utilizado no algoritmo. No entanto, caso ele falhe, o modelo da última execução é testado também.

3.2.2.3 Iteração 3

Apesar da solução anterior conseguir identificar bem as peças, a saída da rede não conta com nenhuma informação de rotação, ou seja, a rede não consegue identificar o ângulo de uma peça inclinada, como está ilustrado na Figura 39. A peça está inclinada, apontando para baixo, no entanto o retângulo delimitador encontrado pela rede não tem essa informação. Isso é esperado já que a rede não foi treinada com informação de ângulo. Tal situação pode ser vista também na Figura 36.

Para resolver o problema tentou-se utilizar uma rede que tivesse as informações de ângulo, uma versão adaptada da YOLO chamada YOLO-OBB[6]. No entanto os resultados foram muito ruins e a ideia foi descartada. Exemplificando, um dos melhores resultados de detecção pode ser visto na Figura 40.

[6] https://github.com/hukaixuan19970627/yolov5_obb



(a) Treino **(b) Treino**

(c) Validação **(d) Validação**

Figura 38 – Evolução da loss das últimas execuções

3.2.3 Iteração 4

Por fim, a solução encontrada foi combinar as abordagens. Utilizar a YOLO para encontrar as peças e recortá-las da imagem e em seguida utilizar um procedimento semelhante ao descrito na subseção 3.2.2.1, mas buscando a reta que mais se adequa à face da peça. Vale ressaltar que esse procedimento extra é necessário apenas para obter o ângulo entre as peças, já que os desalinhamento translacionais (X, Y e Z) são obtidos diretamente da *bounding box* do resultado da inferência da YOLO. A implementação está detalhada na subseção 4.2.2

3.2.4 Testes

Inicialmente foi planejado desenvolver uma versão simulada da plataforma e realizar os testes simulados, com o objetivo de reduzir problemas de hardware e o tempo necessário para corrigí-los. No entanto, uma simulação minimamente fiel provou-se complicada. Assim, optou-se por realizar os testes diretamente na plataforma disponível no laboratório da Universidade Tecnológica Federal do Paraná (UTFPR), sendo a mesma estabilizada contra vibrações por meio de um compressor que sustenta a mesa sobre a qual a plataforma se apoia.

Os passos seguindos durantes os testes estão descritos abaixo:

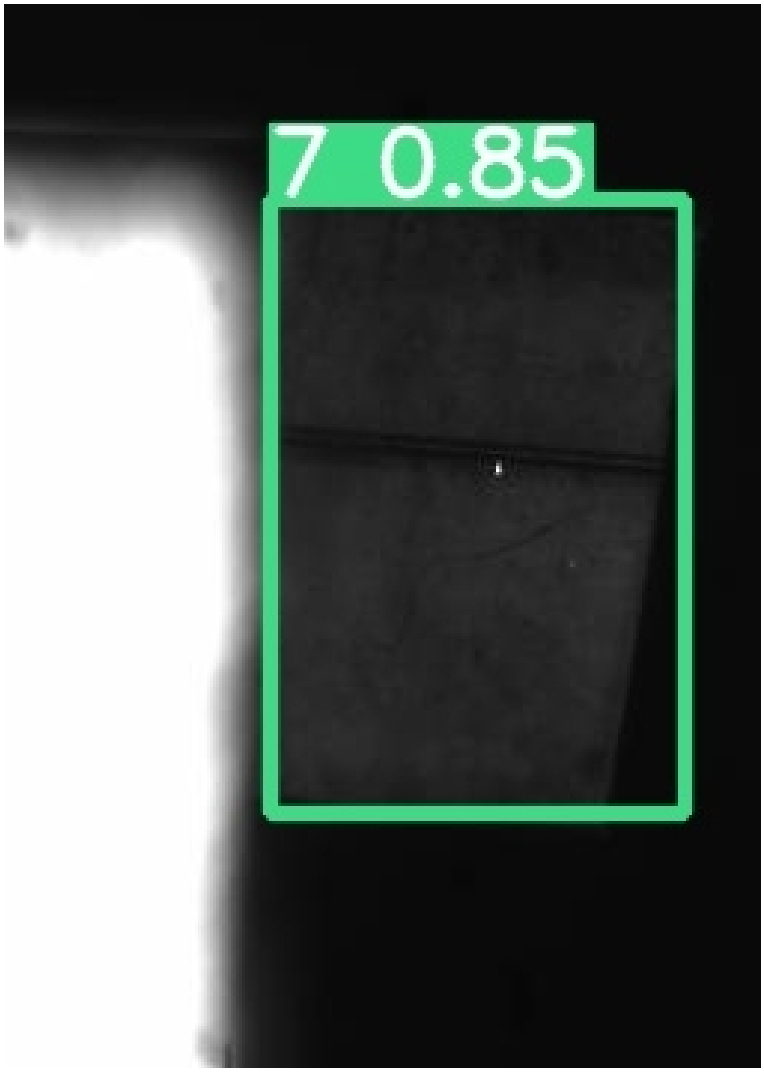


Figura 39 – Ilustração problema de peças inclinadas
Fonte: Autoria própria (2022).

1. Definir uma distância de segurança entre as peças;
2. Executar o algoritmo com interação com motores desativada;
3. Diminuir a distância de segurança das distâncias encontradas;
4. Anotar as distâncias finais;
5. Usar o software proprietário da Thorlabs para mover os motores de acordo com as distâncias;
6. Se as peças estiverem visualmente longe, diminuir distância de segurança e reiniciar o processo;
7. Se peças estiverem suficientemente perto, ativa a interação com os motores e tenta novamente;
8. Ajusta o algoritmo de acordo com resultados.

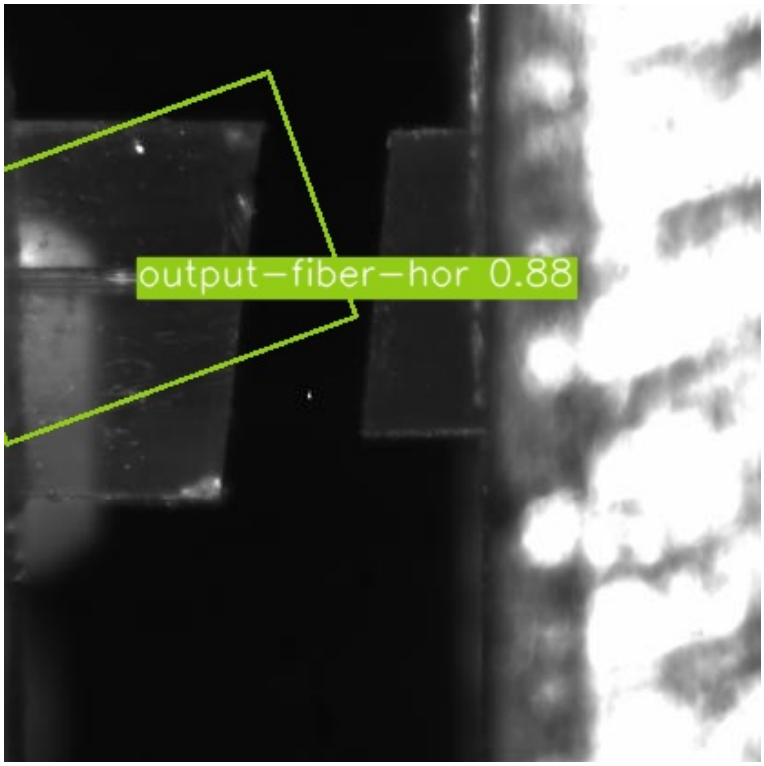


Figura 40 – Detecção falha da YOLO-OBB
Fonte: Autoria própria (2022).

3.2.5 Implementação do alinhamento fino

Os algoritmos de alinhamento fino foram adaptados de (ABRIL, 2020) e não foram o foco do trabalho, sendo descritos adiante.

Após realizada a etapa do alinhamento grosseiro com o algoritmo desenvolvido neste trabalho, inicia-se o processo de alinhamento fino, o qual é constituído de 3 partes:

1. Balanceamento dos canais de saída (alinhamento no eixo RX) com emprego dos medidores de potência ótica;
2. Busca em espiral que melhora levemente o resultado do alinhamento grosseiro;
3. Busca usando algoritmo de otimização (Simplex) de forma a otimizar os valores lidos nos medidores de potência.

3.2.6 Desafios

Após executar o alinhamento grosseiro apenas um dos canais de saída do divisor ótico apresentava uma perda satisfatória. O motivo suspeitado inicialmente era o desalinhamento, ilustrado em Figura 28.

No entanto, ao tentar realizar um balanceamento dos canais o resultado piorava consideravalmente. Além de não se conseguir balancear adequadamente a potência entre os dois

guias monitorados, tinha-se a impressão que um dos canais do *splitter* ou do *array* de saída (ou ambos) estava com problema. Vários alinhamentos manuais foram tentados, buscando-se excluir erros do software, no entanto o problema persistia.

Buscando confirmar se o problema era no software ou hardware, foi realizada inicialmente uma inspeção visual das peças, utilizando as câmeras da plataforma e posicionando o *splitter* cuidadosamente com uma pinça, procurando algum defeito visível, além de buscar confirmar se a razão de divisão entre os canais era realmente 1x4.

A Figura 41 mostra a entrada do *splitter*. Na visão superior é possível ver o canal central onde a luz é acoplada, também existem dois canais laterais, mas não foi descoberta a utilidade desses dois supostos canais nas extremidades, pois não parece ser possível acoplar luz neles.

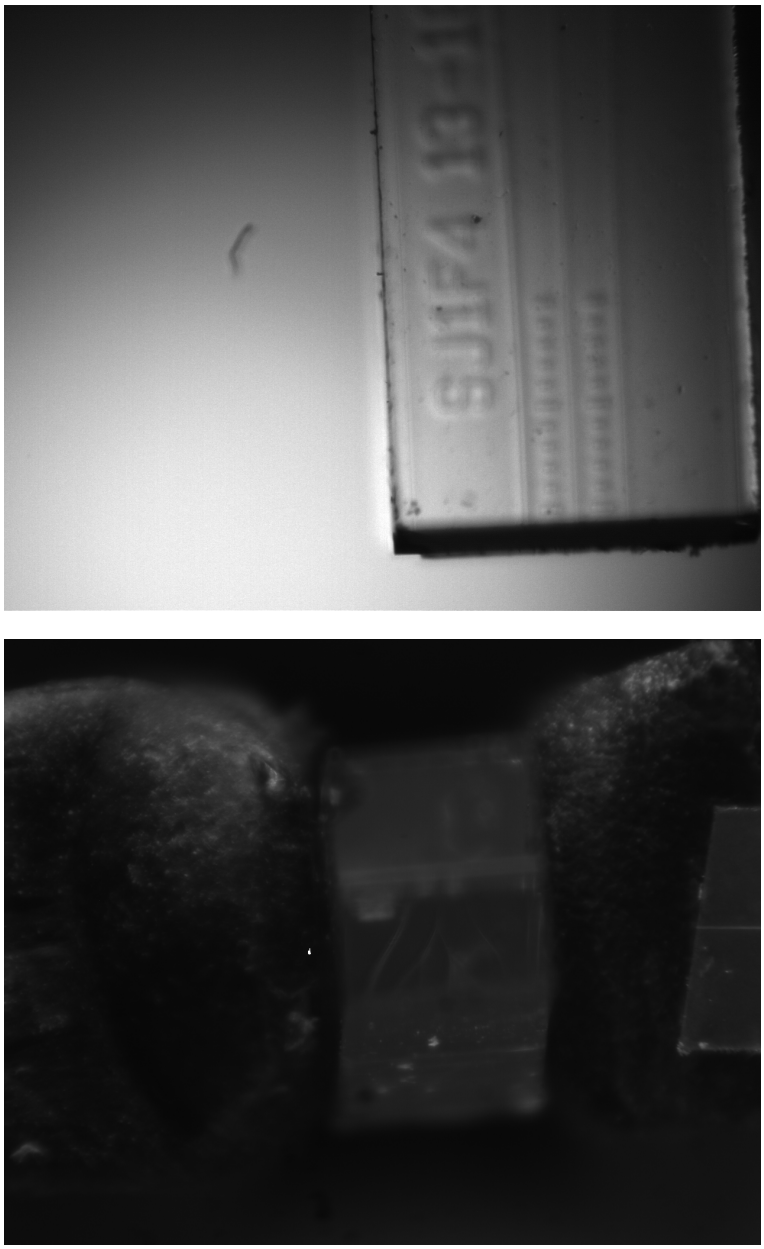


Figura 41 – Vista da entrada do divisor
Fonte: Autoria própria (2022).

A Figura 42 mostra uma foto do meio do *splitter*. É possível ver a divisão do canal de entrada em 2. Cada um desses 2 canais é novamente dividido em 2, sendo possível ver essa segunda separação na parte de baixo da imagem Figura 43.

Figura 42 – Vista superior do meio do divisor
Fonte: Autoria própria (2022).

A Figura 43 mostra a saída do *splitter* em 2 ângulos diferentes na vista superior, além da seção tranversal, onde se pode notar os guias na parte central.

Com isso foi confirmado que o *splitter* realmente tem a razão 1x4 e a localização dos guias de onda encontra-se no centro da peça.

Após esses testes um perfilômetro (Figura 44) da Thorlabs foi utilizado para se verificar a distribuição do sinal na saída do *splitter*. Com esse equipamento foi possível ver os *spots* de saída do splitter. Para isso o perfilômetro foi posicionado, com um alinhamento manual, à saída do divisor, procurando-se otimizar manualmente a potência ótica na saída, buscando-se encontrar os 4 *spots*.

O resultado pode ser visto na Figura 45, onde os 4 *spots* podem ser vistos. Observa-se que apenas os dois canais da esquerda apresentam maior intensidade de sinal. Isto reporta a um possível problema de construção do divisor 1x4, que não está separando corretamente o feixe de luz na entrada. Este resultado corrobora os dados de potência ótica encontrados nos testes de alinhamento realizados com o algoritmo, onde apenas um dos canais (o mais da esquerda) tinha a potência ótica em nível mais elevado. Um outro *splitter* foi testado, mas ele também apresentou problema e nenhum alinhamento suficientemente bom foi obtido.

Aproveitando o uso do perfilômetro, o perfil da fibra de entrada também foi medido e está ilustrado na Figura 46. Pode-se ver que o perfil é simétrico e condiz com o apresetado na Figura 8, onde se assume que o perfil de distribuição do campo elétrico pode ser modelado por uma curva gaussiana.

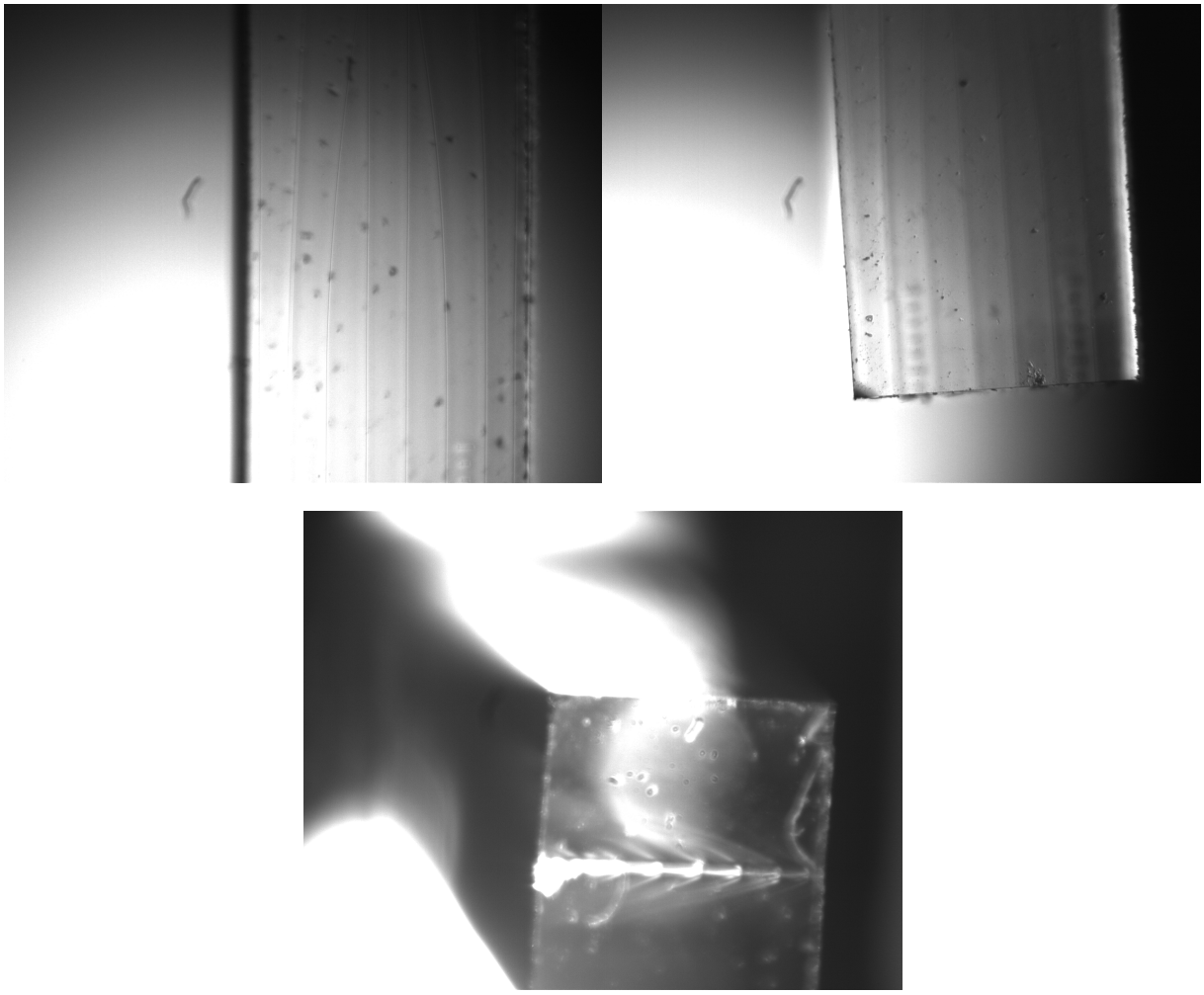


Figura 43 – Vista da saída do divisor

Para dar prosseguimento no trabalho foi substituído o *array* de saída por uma única fibra e *splitter* central foi removido, de modo que problemas no hardware pudessem ser minimizados. O alinhamento passou a ser realizado entre duas fibras diretamente conforme Figura 47. Nesse cenário o melhor alinhamento obtido gerou uma perda de -7dBm, as posições finais podem ser vistas na Figura 48 e Figura 49. Apesar de não parecer perfeitamente alinhado, principalmente na vertical, essas posições tiveram perda menor que um alinhamento visual, possivelmente por alguma variação intrínseca das fibras.

Feitos esses testes voltou-se a utilizar o arranjo original (fibra-divisor-fibra) levando em conta os fatos observados:

- Os canais não acoplam luz de maneira igual, apenas um dos canais parece estar acoplando de maneira aceitável. Assim apenas um canal será alinhado efetivamente, o outro será sempre ignorado.
- A melhor perda possível é de -7dBm.

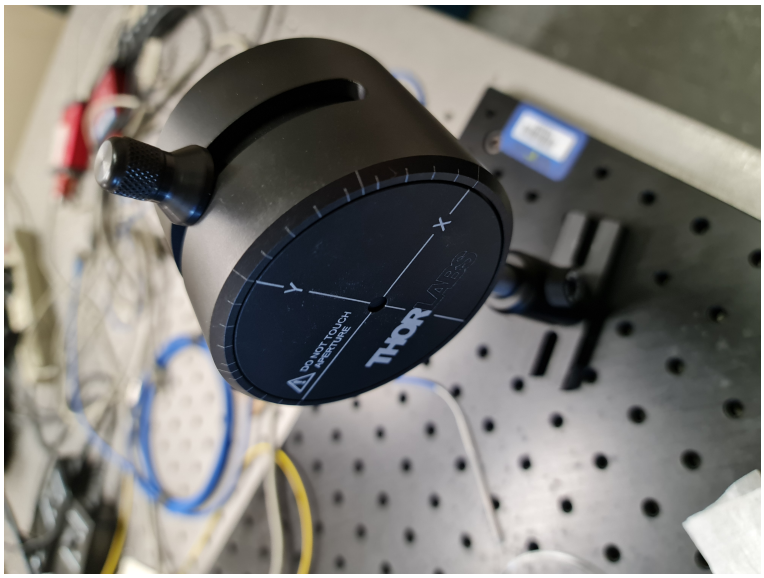


Figura 44 – Perfilômetro
Fonte: Autoria própria (2022).

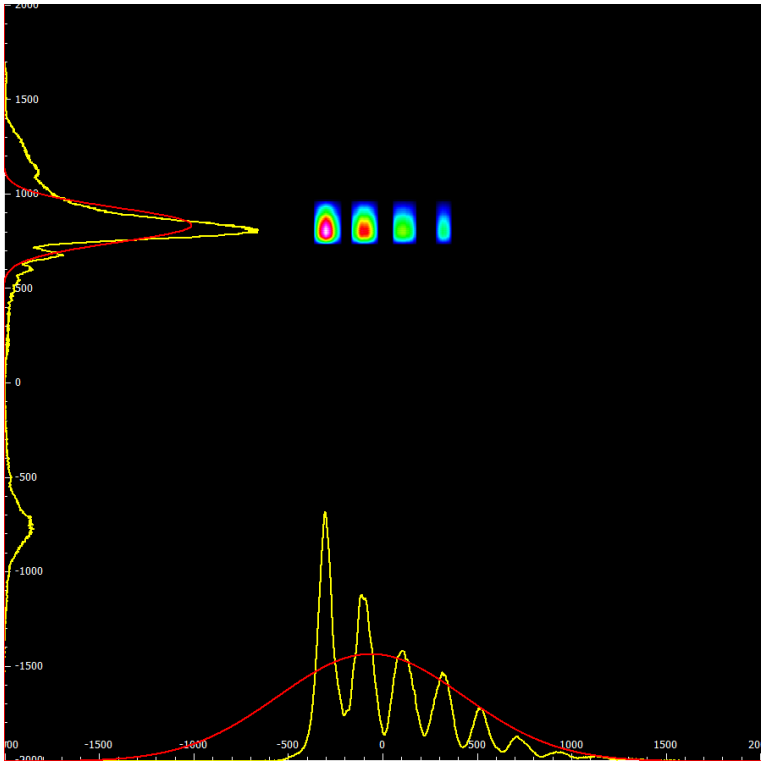


Figura 45 – Perfil do divisor de saída
Fonte: Autoria própria (2022).

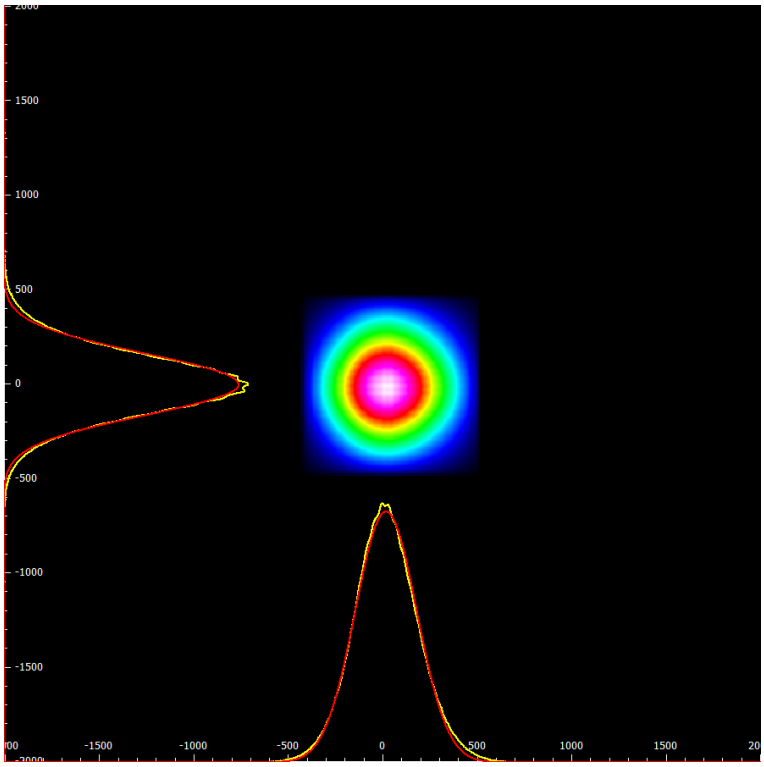


Figura 46 – Perfil da fibra de entrada

Fonte: Autoria própria (2022).

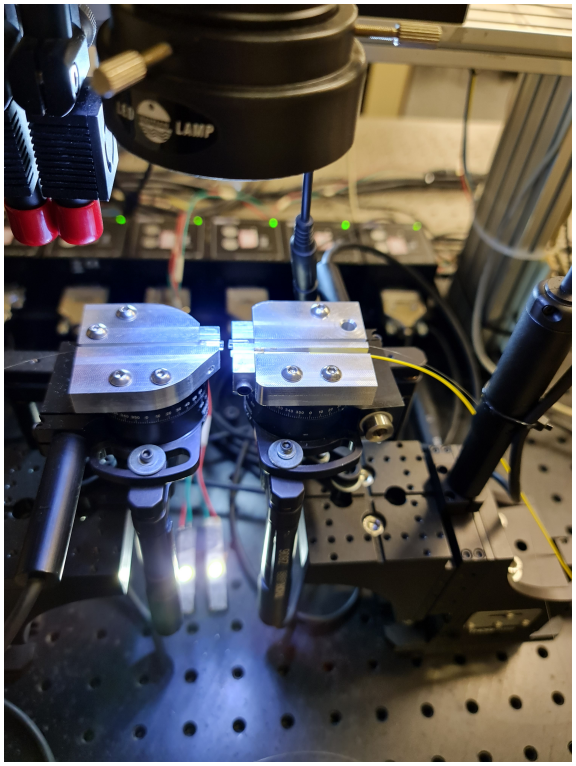


Figura 47 – Novo arranjo para alinhamento fibra-fibra

Fonte: Autoria própria (2022).

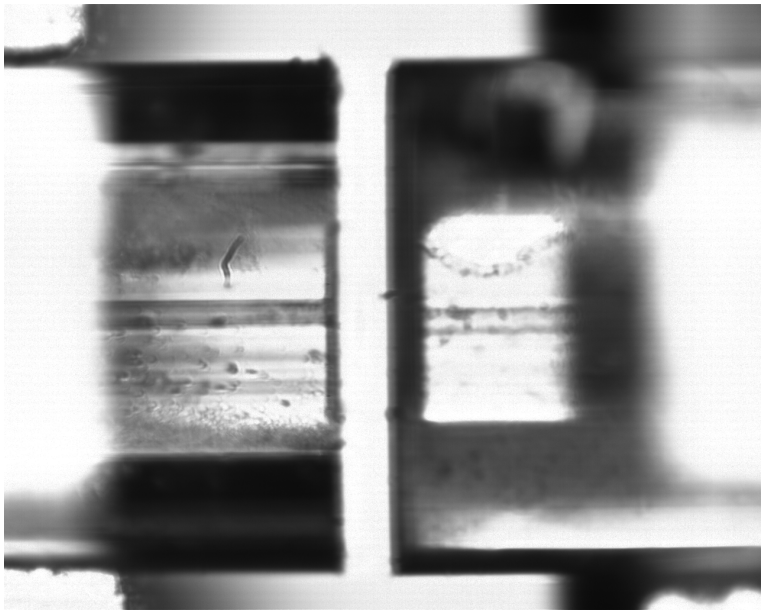


Figura 48 – Melhor alinhamento fibra-fibra vertical
Fonte: Autoria própria (2022).

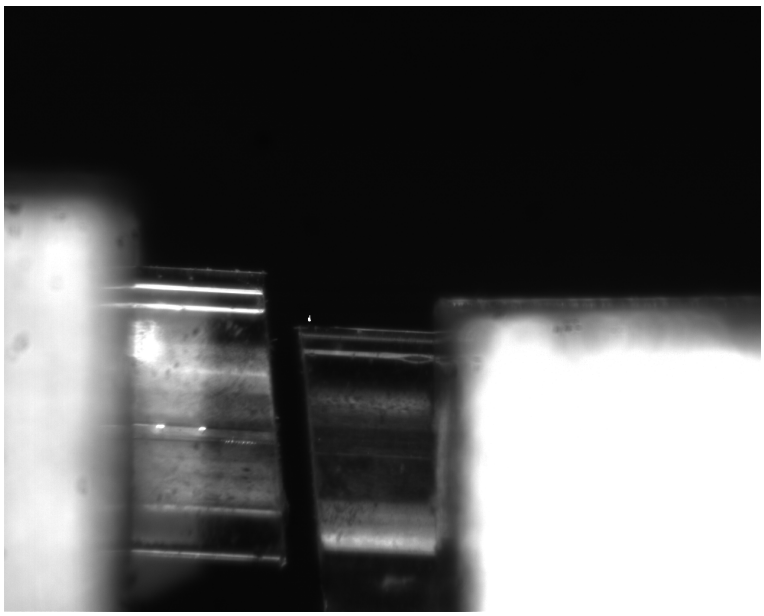


Figura 49 – Melhor alinhamento fibra-fibra horizontal
Fonte: Autoria própria (2022).

4 RESULTADOS

Neste capítulo são apresentados os resultados obtidos aplicando a metodologia apresentada no Capítulo 3. Por resultado entende-se os valores finais de atenuação de potência ótica e outras métricas obtidas durante os diversos testes, assim como a avaliação de desempenho dos algoritmos desenvolvidos.

4.1 Escopo do sistema

Como discutido previamente, o foco deste trabalho é o alinhamento grosseiro, a etapa inicial do algoritmo. O alinhamento fino foi adaptado diretamente do trabalho de Nicolas Abril. Além disso esteve fora do escopo desenvolver qualquer tipo de interface gráfica ou via terminal. O algoritmo é executado diretamente a partir da execução de um script Python via linha de comando (PowerShell[1]), sem interatividade alguma com o usuário.

4.2 Implementação do sistema

Nesta seção são apresentados os algoritmos para o alinhamento grosseiro e fino, assim como as medidas realizadas após cada etapa.

O software é composto de várias partes, algumas específicas para interação com o hardware e outras com a lógica específica para alinhamento. De acordo com o escopo desse projeto, as partes de interação com o hardware são omitidas para que possa ser dado o foco adequado aos algoritmos de alinhamento.

4.2.1 Visão geral

O software implementado segue uma ordem relativamente simples

- Cria as interfaces para os componentes de hardware (motores, medidores e câmeras);
- Move câmeras e motores para uma posição inicial;
- Passa essas interfaces para os **Aligners**;
- Executa o alinhamento com o método *align()* de cada um em uma ordem específica.

[1] https://learn.microsoft.com/pt-br/powershell/scripting/overview?view=powershell-7.3

4.2.2 Alinhamento Grosseiro

Em alto nível o alinhamento grosseiro segue os passos descritos abaixo. O procedimento é análogo para a **SAIDA** do sistema.

1. Move sistema para ENTRADA
2. Tira fotos
3. Obtém distâncias
4. Converte distâncias em passos dos motores
5. Move motores

A parte central do algoritmo é a obtenção das distâncias, cujo o fluxograma está apresentado na Figura 50

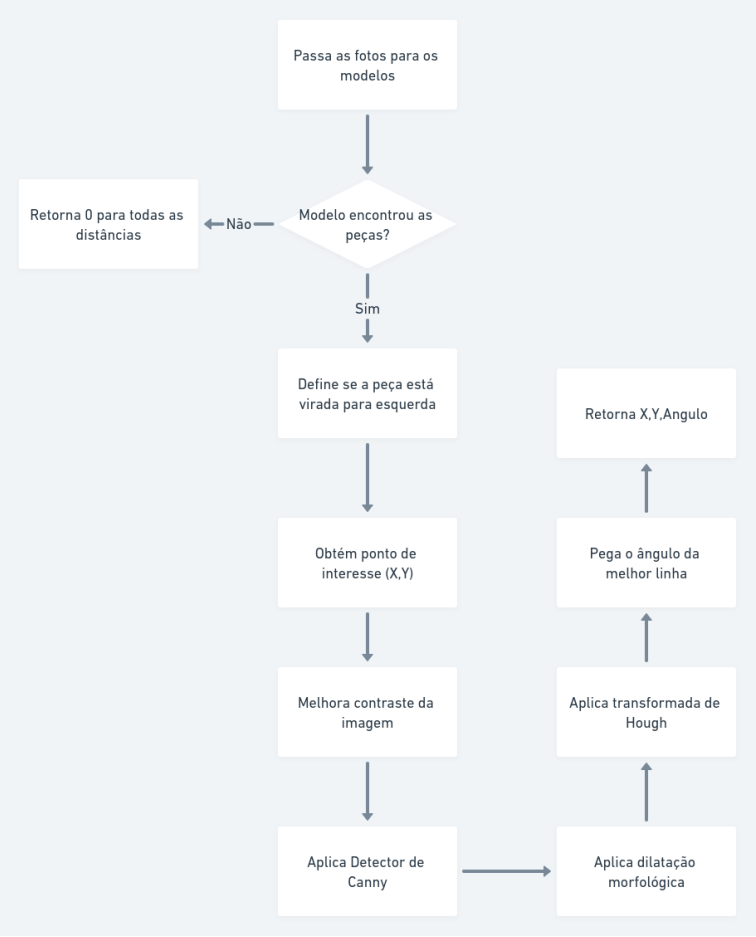


Figura 50 – Fluxograma obtenção das distâncias
Fonte: Autoria própria (2022).

O conjunto de imagens a seguir ilustra os passos descritos na Figura 50.

Dada a entrada da Figura 51 o modelo recorta as peças (Figura 52), o ponto de interesse é encontrado (Figura 53), em seguida o processamento é realizado (Figura 54) dessa imagem final o ângulo é retirado.

No entanto, percebeu-se que os ângulos não estavam precisos o suficiente, em geral pioravam o alinhamento, por isso eles foram descartados, ainda são calculados, mas no momento são ignorados. Assim o alinhamento grosseiro passa a supor que não existe desalinhamento angular entre as peças.

Vale destacar na Figura 53 como os pontos de interesse estão mais próximos que o esperado, isso ocorreu pelo recorte da fibra não ter sido muito preciso, como pode ser visto na esquerda da Figura 52.

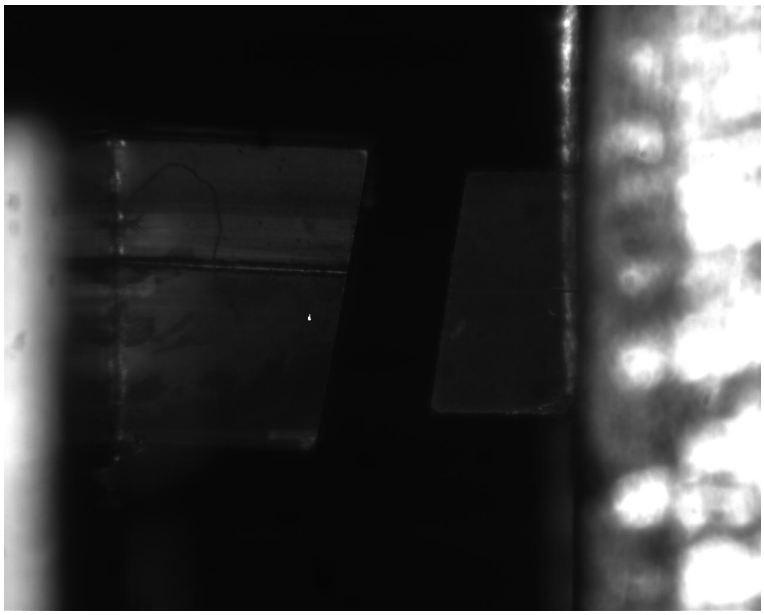


Figura 51 – Entrada do alinhamento grosseiro
Fonte: Autoria própria (2022).

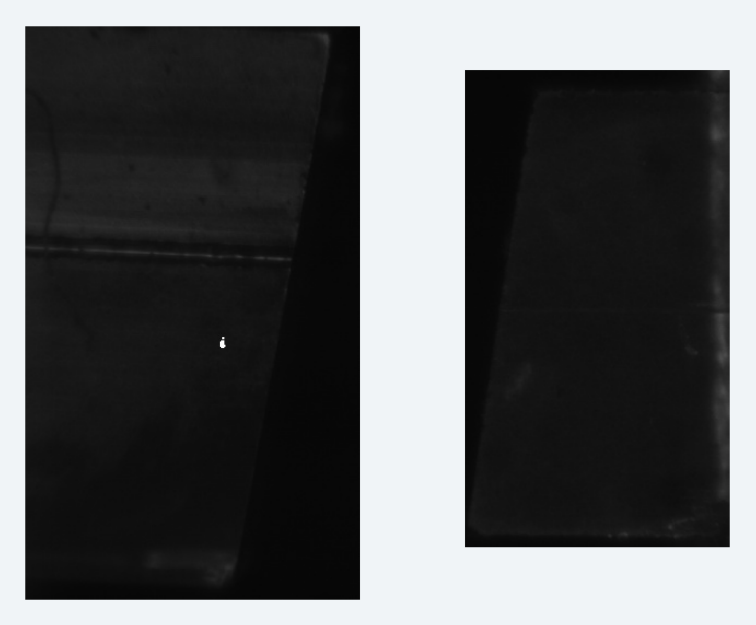


Figura 52 – Recortes do modelo treinado

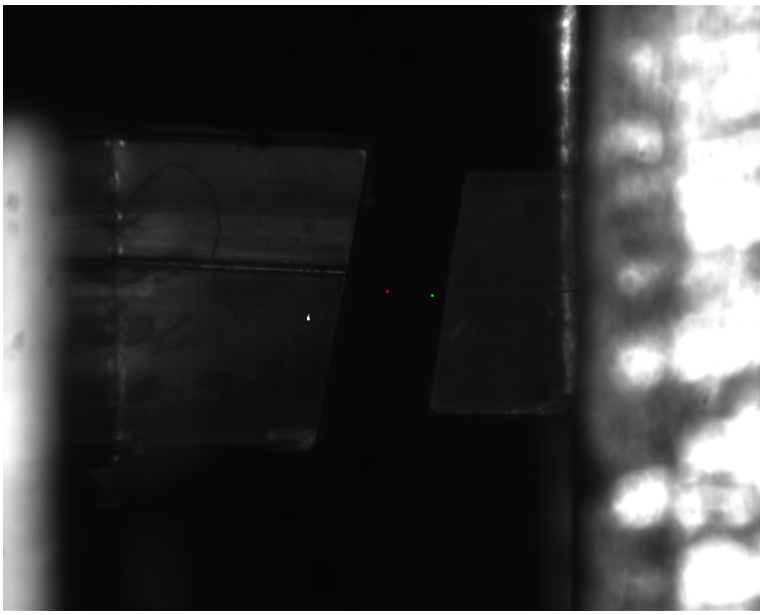


Figura 53 – Ponto de interesse

4.2.3 Alinhamento Fino

Após realizar o alinhamento grosseiro a etapa final do *software* é iniciada.

Essa parte pode ser subdividida nos seguintes passos, cada passo tem o objetivo de reduzir o espaço de busca do próximo. A busca em espiral é mais rápida, por isso é executada antes e utilizada para restringir o espaço da busca Simplex.

1. Busca em espiral em YZ com passos de 0.01 mm
2. Busca em espiral em YZ com passos de 0.001 mm

Figura 54 – Recorte da fibra processada

3. Busca Simplex em todos os eixos.

4.3 Testes

A Tabela 3 apresenta os resultados dos testes de repetibilidade. Esse teste consiste em executar o algoritmo completo 3 vezes partindo, aproximadamente, do mesmo ponto e comparar o resultado das execuções.

No decorrer do alinhamento as perdas de potência ótica foram amostradas. As curvas para os 3 testes estão na Figura 55. É possível observar que todos os testes tem um comporta-

Tabela 3 – Testes de repetibilidade

Perda antes (dBm)	Perda depois (dBm)	Tempo de execução (seg)
-49.5	-30.5	400
-49.9	-30.5	401
-49.9	-29.7	408

Fonte: Autoria própria (2022).

mento semelhante, além de nenhum ter apresentado melhoria significativa após a amostra 150, ou seja o algoritmo está muito próximo do seu limite.

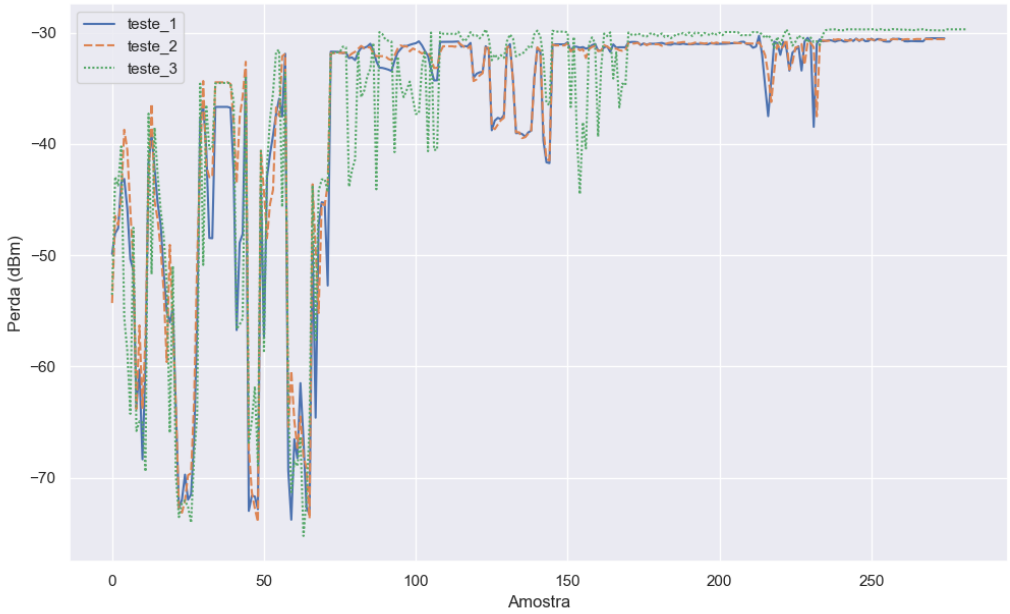


Figura 55 – Evolução da perda de potência ótica durante a execução do algoritmo

5 CONCLUSÃO

Conforme o mundo fica mais conectado a largura de banda e velocidade de transmissão elevada de dados tornam-se mais importante. A fibra ótica provou-se um dos meios de transmissão mais efetivos para atingir esses objetivos e com isso surge a necessidade cada vez maior de dispositivos óticos com baixa perda e fabricação rápida e automatizada. Nesse sentido, *softwares* para controlar plataformas de alinhamento dos componentes desses dispositivos se tornam essenciais e o caminho mais viável para produção em massa, permitindo que as redes óticas tenham maior penetração.

Utilizando a plataforma apresentada nesse trabalho, em conjunto com técnicas de visão computacional, ML foi possível detectar as peças, calcular as distâncias necessárias e controlar os atuadores para realizar um alinhamento grosseiro, de modo que o objetivo proposto foi atingido. Apesar de o resultado final não ser satisfatório, o foco desse projeto concentrou-se no alinhamento grosseiro, que foi realizado com sucesso. O trabalho também explorou diversas estratégias e realizou inúmeros testes, encontrando falhas nas peças e limitações no hardware.

Baseado nos procedimentos aqui descritos e resultados obtidos, verificou-se que ainda não foi possível satisfazer o objetivo de baixa perda almejado com o atual estado da plataforma. Além do hardware, problemas na fabricação dos guias de onda nos substratos planares foram identificados, o que impediu a realização otimizada do alinhamento final. Por fim, certamente existe possibilidade de melhoria no *software*, em particular na parte de alinhameno fino, por intermédio de abordagens completamente diferente daquelas descritas nesse trabalho.

5.1 Trabalhos futuros

Algumas melhorias que podem ser implementadas no futuro são:

- Analisar possibilidade de reprojeto da plataforma;
- Melhorar o modelo treinado, possivelmente reduzindo o número de classes, de maneira que apenas um modelo seja o suficiente para detectar as peças;
- Treinar o modelo para reconhecer diretamente os guias de onda, ao invés das peças inteiras;
- Aprimorar o alinhamento grosseiro pra trabalhar com ângulos (de maneira precisa);
- Analisar outras formas de otimização que não dependam dos medidores de potência;
- Melhorar a precisão do alinhamento fino;
- Desenvolver uma inteface gráfica para o algoritmo.

REFERÊNCIAS

ABRIL, N. Relatório técnico. 2020.

ACADEMY, R. **Mathematical Morphology**. 2022. Disponível em: https://robotacademy.net.au/lesson/mathematical-morphology/. Acesso em: 2022-10-27.

ANATEL. **Anatel - Banda Larga Fixa**. 2021. Disponível em: https://informacoes.anatel.gov.br/paineis/acessos/banda-larga-fixa. Acesso em: 2021-07-18.

Blank, J.; Deb, K. pymoo: Multi-objective optimization in python. **IEEE Access**, v. 8, p. 89497–89509, 2020.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing (3rd Edition)**. USA: Prentice-Hall, Inc., 2006. ISBN 013168728X.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S.l.*]: MIT Press, 2016. http://www.deeplearningbook.org.

HELEN, K. **Reflection, Refraction and Diffraction**. 2022. Disponível em: http://www.thestargarden.co.uk/Copyright.html. Acesso em: 2022-10-26.

HISOUR. **Modelo de cores RGB**. 2022. Disponível em: https://www.hisour.com/pt/rgb-color-model-24867. Acesso em: 2022-10-27.

IRRERA, F. Improved analysis of coupling between 2-d optiocal waveguides and signle mode fibers. **Optical Engineering**, 1988.

JADON, S. Introduction to different activation functions for deep learning. **Medium, Augmenting Humanity**, v. 16, 2018.

JOCHER, G. **YOLOv5**. 2020. Disponível em: https://github.com/ultralytics/yolov5. Acesso em: 2022-11-04.

NELDER, J. A.; MEAD, R. A Simplex Method for Function Minimization. **The Computer Journal**, v. 7, n. 4, p. 308–313, 01 1965. ISSN 0010-4620. Disponível em: https://doi.org/10.1093/comjnl/7.4.308.

OPENCV. **Hough Line Transform**. 2022. Disponível em: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html. Acesso em: 2022-10-25.

OPENCV. **OpenCV: Smoothing Images**. 2022. Disponível em: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html. Acesso em: 2022-12-21.

OPTIK, S. **Fiber Acceptance Cone**. 2022. Disponível em: http://www.sinaranoptik.com/fiber-optics-blog/fiber-acceptance-cone. Acesso em: 2022-10-26.

PROGRAMMER, A. G. **Convolutional neural network 2**. 2022. Disponível em: https://aigeekprogrammer.com/convolutional-neural-network-image-recognition-part-2/. Acesso em: 2022-10-29.

RYU, J. H. *et al.* Optical interconnection for a polymeric plc device using simple positional alignment. **Opt. Express**, Optica Publishing Group, v. 19, n. 9, p. 8571–8579, Apr 2011. Disponível em: https://opg.optica.org/oe/abstract.cfm?URI=oe-19-9-8571.

SAHA, S. **A Comprehensive Guide to Convolutional Neural Networks**. 2022. Disponível em: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Acesso em: 2022-10-29.

SALEH, B. E. A.; TEICH, M. C. **Fundamentals of Photonics**. [*S.l.*]: John Wiley Sons, Ltd, 1991. ISBN 9780471213741.

SEEKFIRE. **Overview of model structure about YOLOv5**. 2020. Disponível em: https://github.com/ultralytics/yolov5/issues/280. Acesso em: 2022-11-04.

WORLD, T. . M. **Advantages disadvantages of Step Index, Graded Index Fiber**. 2021. Disponível em: https://www.test-and-measurement-world.com/Terminology/Advantages-and-Disadvantages-of-Step-Index-and-Graded-Index-Fiber.html. Acesso em: 2022-10-26.

ZHENG, Y.; DUAN, J. an. Coupling analysis between planar optical waveguide and fiber array. **Journal of Central South University**, v. 40, n. 3, p. 681–686, 2009.

ZHENG, Y.; DUAN, J. an. Alignment algorithms for planar optical waveguides. **Optical Engineering**, SPIE, v. 51, n. 10, p. 103401, 2012. Disponível em: https://doi.org/10.1117/1.OE.51.10.103401.

ZHENG, Y. *et al.* Automatic planar optical waveguide devices packaging system based on polynomial fitting algorithm. **Advances in Mechanical Engineering**, v. 5, p. 398092, 2013. Disponível em: https://doi.org/10.1155/2013/398092.