ESTUDO DA APLICAÇÃO DE ALGORITMO DE VISÃO YOLO NA INSPEÇÃO DE COMPONENTES ELETRÔNICOS EM PLACAS DE CIRCUITO IMPRESSO

Autor: Carlos Eduardo de Souza Brümmer

RESUMO

O século XXI consolidou a eletrônica como base tecnológica essencial para setores estratégicos da economia, como IoT e computação em nuvem. Dentro do segmento eletrônico, as placas de Circuito Impresso (PCBs) desempenham um papel central ao interconectar componentes e viabilizar dispositivos cada vez mais complexos. Contudo, fabricar placas eletrônicas requer técnicas modernas não apenas na montagem, mas também na inspeção, para revisão de possíveis problemas causados ao longo do processo de montagem. A inspeção tradicional por métodos manuais ou por sistemas de inspeção óptica automatizada (AOI) apresenta limitações significativas, ainda que a última já se utilize de visão computacional.O YOLO (You Only Look Once) surge como alternativa completa na detecção de componentes, permitindo qualificação de defeitos com alta precisão. Este processa imagens integralmente em uma única passagem, identificando automaticamente os possíveis problemas de qualidade, como falta, posicionamento e orientação de componentes. Essas técnicas de automação tornam-se viáveis tanto em indústrias manufatureiras de larga escala, quanto em operações de menor porte, promovendo padronização e precisão na qualidade do processo produtivo.

Palavras-Chave: PCB; Inspeção; AOI; YOLO.

STUDY OF THE APPLICATION OF THE YOLO VISION ALGORITHM IN THE INSPECTION OF ELECTRONIC COMPONENTS ON PRINTED CIRCUIT BOARDS

ABSTRACT

The 21st century has consolidated electronics as an essential technological foundation for strategic sectors of the economy, such as IoT and cloud computing. Within the electronics field, Printed Circuit Boards (PCBs) play a central role by interconnecting components and enabling increasingly complex devices. However, the manufacturing of electronic boards requires modern techniques not only for assembly but also for inspection, aiming to identify potential issues that may arise during the production process. Traditional inspection methods, whether manual or based on Automated Optical Inspection (AOI) systems, present significant limitations, even though the latter already incorporates computer vision. YOLO (You Only Look Once) emerges as a complete alternative for component detection, enabling the identification of defects with high accuracy. The method processes images entirely in a single pass, automatically identifying quality issues such as missing components, misplacement, and incorrect orientation. These automation techniques are viable for both large-scale manufacturing industries and smaller

operations, promoting standardization and greater precision in the production process.

Keywords: PCB; Inspection; AOI; YOLO.

1 INTRODUÇÃO

Os diversos dispositivos eletrônicos cotidianos, antes de chegarem ao público, passam por etapas críticas na indústria, como projeto, construção e validação. Sua fabricação enfrenta desafios crescentes, como a miniaturização extrema, a alta demanda e o rigoroso controle de qualidade, exigido pela quantidade elevada de componentes cada vez menores, porém, mais densamente agrupados (LIDAK, 2005).

O processo produtivo inicia-se com o projeto e desenvolvimento da placa eletrônica, que utiliza softwares robustos e precisos, como KiCad ou Altium Designer. Na etapa de fabricação das PCBs, empregam-se diversas técnicas avançadas de gravação do cobre, como por exemplo fotolitografia e gravação ácida. Em seguida, na montagem dos componentes, integram-se técnicas modernas e tradicionais: a SMT (Surface Mount Technology), para soldagem direta na superfície da PCB, e a THT (Through-Hole Technology), na qual os componentes são inseridos em furos e soldados no lado oposto da placa. Ao final, cabe o controle de qualidade da produção, validando os processos anteriores (LIDAK, 2005).

Na indústria manufatureira, o controle de qualidade tradicional traz deficiências evidentes. Na inspeção manual, a baixa celeridade do processo, alto custo, baixa escalabilidade junto a imprecisão de olhos humanos, fazem com que o método esteja ultrapassado. Mais recente, métodos de automação em visão computacional, baseada em comparação de imagens (Automatic optical inspection) já otimizaram o sistema, trazendo métodos computacionais que retém as diferenças de uma placa correta versus placa de teste, via comparação de imagens (Harshitkumar, 2024). Porém, para esse método de inspeção óptica, assim como outras variantes dos métodos tradicionais de detecção de defeitos, trazem a necessidade de serem extremamente calibrados, de forma que variações mínimas na posição da placa, ângulo de captação da imagem, assim como alteração na cor, forma ou posição do componente já desestabilizam o sistema, necessitando recalibragem. A figura 1 ilustra essas diferenças entres os métodos tradicionais, e o uso de inteligências artificiais nos sistemas de controle de qualidade.

Parameter	**Traditional QC**	**AI-Driven QC**
Defect Detection Speed	Moderate	High
Precision	Medium	Very High
Cost	High (manual labor, advanced equipment)	Initial investment, low maintenance
Human Involvement	High	Minimal
Scalability	Limited	Easily scalable

Figura 1: Comparação de métodos de controle de qualidade tradicionais e baseados em IA.Fonte: Harshitkumar Ghelani. (2024)

2 TAXONOMIA DOS DEFEITOS EM PCB

Na fabricação de placas eletrônicas, a corrosão, a temperatura, problemas de solda, entre outros, são frequentemente destacados como vilões comuns. No entanto, para o controle de qualidade e a engenharia de produção, uma categoria de defeito é notoriamente persistente e crítica: os erros de montagem relacionados aos componentes. Estas são falhas humanas ou de automação que ocorrem durante a fase de colocação dos componentes na placa, e seus impactos variam de simples mal funcionamento até a perda total do dispositivo (Jian et al., 2024).

Dentre esses erros, quatro se destacam pela sua frequência e pelo seu poder de causar falhas catastróficas: Adição de componente, falta de componente, inversão de componente e o erro no posicionamento do componente (Jian et al., 2024). A figura 2 desenvolve uma breve taxonomia dos defeitos em PCB, focando no componente.

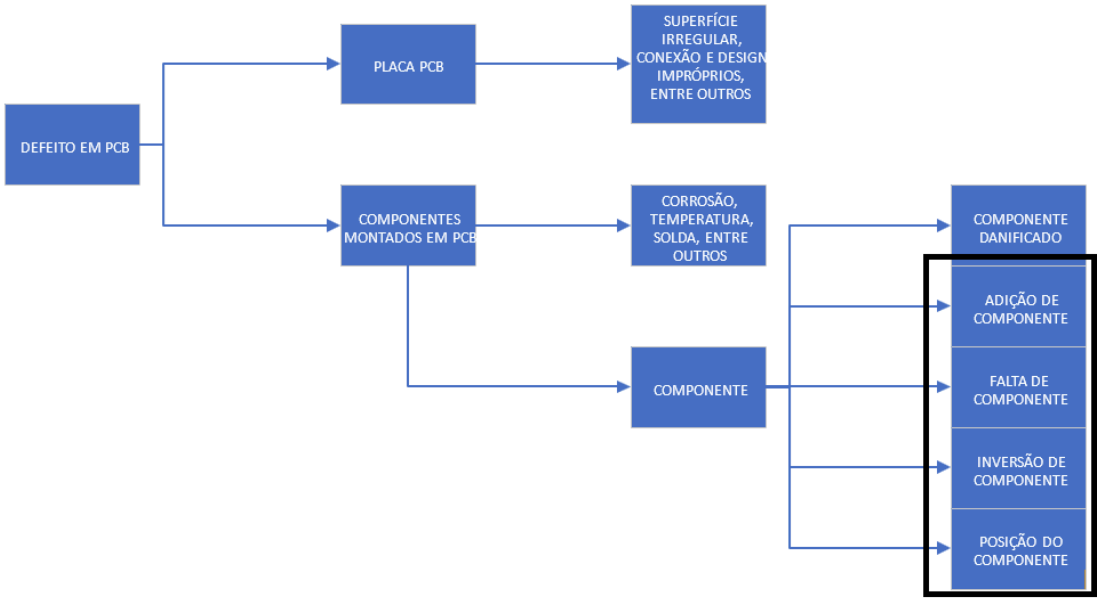

Figura 2: Taxonomia dos Possíveis defeitos em PCBs. Fonte: Adaptado de (Jian et al., 2024)

A adição de um componente onde não é projetado para estar é um erro que frequentemente causa curtos-circuitos imediatos. Um componente extra pode criar conexões elétricas não intencionais entre trilhas, alimentação e terra, resultando em superaquecimento, queima de outros componentes e, potencialmente, na inutilização da placa. A depuração desse problema é particularmente complexa, pois a placa contém um elemento que sequer deveria existir no seu projeto, desviando a análise do caminho lógico esperado.

De modo oposto, porém igualmente grave, é a falta de um componente. A ausência de um resistor, capacitor ou circuito integrado crucial interrompe o fluxo de

sinal ou energia, paralisando uma função essencial do circuito. O sintoma é a não funcionalidade de um módulo inteiro, e a identificação do componente faltante pode ser um processo minucioso de verificação ponto a ponto contra o diagrama esquemático, consumindo tempo de diagnóstico.

Talvez o pior, dentre esses erros, seja a inversão de componente. Este defeito específico afeta componentes polarizados, como transistores, diodos e circuitos integrados. Ao ser soldado com a polaridade invertida, um circuito integrado pode queimar. Um diodo colocado ao contrário simplesmente não conduzirá a corrente no sentido necessário, bloqueando o funcionamento do circuito. O perigo maior é que a placa pode energizar e parecer funcionar por instantes antes de falhar de forma abrupta e, muitas vezes, destrutiva.

De forma geral, enquanto fatores ambientais e de soldagem são contornados com processos e materiais adequados, os erros de montagem, adição, falta, inversão e posicionamento, exigem um rigoroso sistema de inspeção.

3 ARQUITETURA DE INSPEÇÃO EM PLACAS ELETRÔNICAS

Contudo, os métodos de aprendizado de máquina, quando integrados à visão computacional e à inteligência artificial, superam essas limitações das AOIs ao oferecer uma perspectiva mais flexível e assertiva, que não depende de parâmetros predefinidos e excessivamente regulados. Atualmente, técnicas modernas de detecção e classificação de objetos, como o YOLO (You Only Look Once), combinadas com pré-processamento de imagem (J. Chen et al., 2022), aumento de dados (YANG et al., 2022), e métodos de iluminação eficazes (*Zhou et al., 2023*), elevam significativamente o desempenho do sistema de inspeção, garantindo maior precisão e eficiência na modelagem. A figura 3 demonstra o ciclo de inspeção de componentes usualmente utilizados.

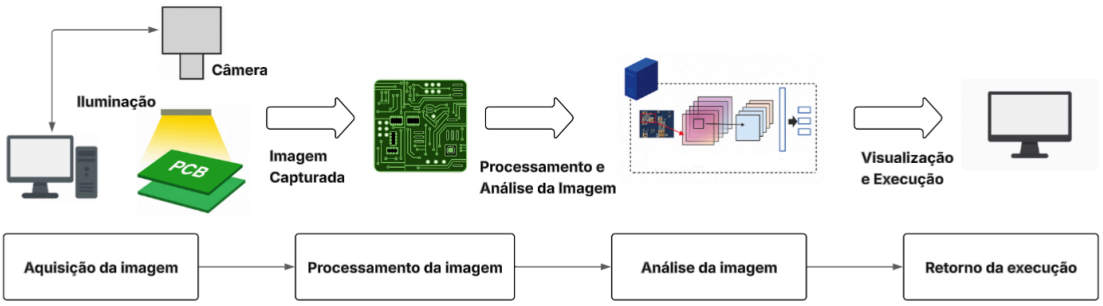

Figura 3: Processo de Inspeção de componentes. Fonte: Adaptado de Zhou et al., 2023

O aprendizado de máquina (machine learning) é um método matemático que aprende de forma autônoma a partir de dados, com intervenção humana apenas na extração de características, realizando reconhecimento de padrões, classificação e predição. Dessa forma, sua aplicação deve-se à sua eficácia no processamento e na aprendizagem de grandes volumes de informação. No contexto específico de inspeção de placas de circuito impresso, por exemplo, as técnicas de aprendizado

de máquina são utilizadas para classificar defeitos com base em características extraídas por métodos convencionais de processamento de imagem.

Por sua vez, o Aprendizado profundo (deep Learning), subárea do aprendizado de máquina, diferencia-se em suas abordagens, uma vez que, diferente dos métodos tradicionais de aprendizado de máquina, a abordagem baseada em deep learning não depende de métodos convencionais de extração de características. Em vez disso, oferece uma solução de aprendizado do início ao fim do processo, na qual a extração, identificação e classificação de características defeituosas em imagens de PCB são realizadas automaticamente pelas redes, que reconhecem defeitos com base em características aprendidas, exigindo menos conhecimento prévio ou especializado (Y. Zhou et al.,2023).

O deep learning envolve duas etapas principais: treinamento e teste. Durante a etapa de treinamento, uma quantidade substancial de dados passa por um processo de rotulagem, que define características identificáveis. Por meio da comparação dessas características, o modelo aprende a identificar padrões, tornando-se capaz de realizar previsões e decisões precisas ao encontrar dados similares no futuro. Na etapa de teste, o modelo aplica o conhecimento adquirido para analisar novos dados e fazer previsões (Patel e Thakkar, 2020).

Para o contexto da detecção e classificação de objetos, O YOLO traz uma abordagem de aprendizado de máquina que adota um problema de regressão para separar espacialmente as caixas delimitadoras (bounding boxes) de um determinado objeto e as probabilidades de classe associadas, realizando assim a classificação de objetos. Trata-se de uma rede convolucional única que prevê simultaneamente múltiplas caixas delimitadoras para múltiplos objetos e, em seguida, gera uma probabilidade de classe para cada objeto. (Redmon, J. et al., 2016).

Na sequência do processo do YOLO, a imagem de entrada é dividida em uma grade, SxS, onde cada célula da grade é responsável por prever se há um objeto em sua região. Em seguida, o modelo gera caixas delimitadoras (bounding boxes) associadas a um valor de confiança que indica a probabilidade de um objeto estar presente e a precisão da previsão. Paralelamente, é produzido um mapa de probabilidades de classe, que aponta a categoria mais provável para cada região da grade. Combinando as informações de localização, confiança e classe, e aplicando técnicas para remover sobreposições redundantes, se obtém os objetos identificados com suas caixas e rótulos definidos. Sendo assim, a figura 4 indica a estrutura base de detecção de um estágio, caso do YOLO.

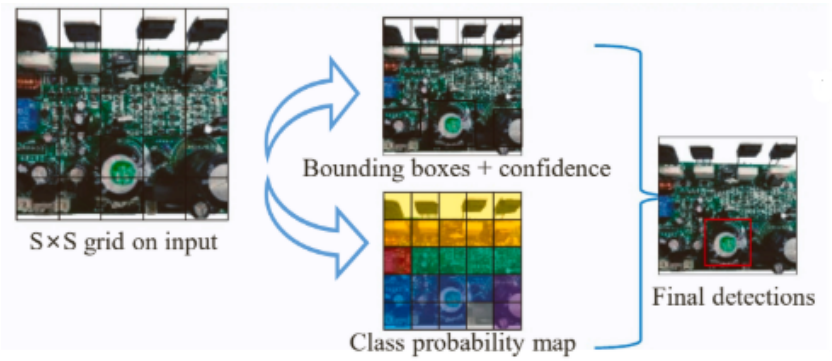

Figura 4: Detecção de objeto de um estágio. Fonte: Adaptado de Zhou et al., 2023

4 DESIGN DO SISTEMA

Desde a captação das imagens, pré-processamento e então modelagem, diversos critérios são adotados, seja de iluminação, posição de captação da imagem, formato de arquivo de foto para tratamento, técnicas de aumento de dados e também de modelagem da Deep Learning. De forma geral, cada um dos aspectos comentados anteriormente, tem grande influência nos resultados, por isso, capturar não só a melhor técnica, mas desenvolver um padrão para cada processo é essencial.

Para desenvolver o projeto, foi utilizado um protótipo de placa eletrônica base. A partir dele, criaram-se modelos capazes de identificar cada componente individualmente, prevendo não apenas sua presença ou ausência, mas também se estão com a polaridade correta. Para o bom funcionamento do modelo, foi essencial escolher componentes que se diferenciam entre si, seja no formato ou na coloração. Isso porque o modelo identifica as peças justamente por suas características visuais, como forma e cor, e não por meio de nomenclaturas ou outros tipos de diferenciação. Após essa etapa, ainda existe um processo importante, a validação da posição dos componentes. Com os dados de posicionamento captados pelo modelo, utiliza-se cálculos euclidianos para comparar as coordenadas identificadas com aquelas consideradas corretas.

4.1 PROTÓTIPO DE PLACA PARA AQUISIÇÃO DOS DADOS

O protótipo utilizado foi uma placa de circuito impresso (PCI) com componentes soldados, contendo 2 terminais borne, 1 capacitor cerâmico, 1 capacitor de poliéster, 1 relé, 1 potenciômetro, 8 resistores (de 4 modelos diferentes), 2 transistores BJT, 1 diodo e 1 circuito integrado. Todos os componentes se diferenciam entre si pelo formato, pela cor ou por ambos os aspectos.

Para o projeto, a imagem completa da placa apresentava pouca nitidez. Por isso, optou-se por utilizar uma imagem segmentada em duas, com a linha cortada ao lado do circuito integrado, como na figura 5, o que permitiu capturar fotos mais próximas da lente e com melhor definição.

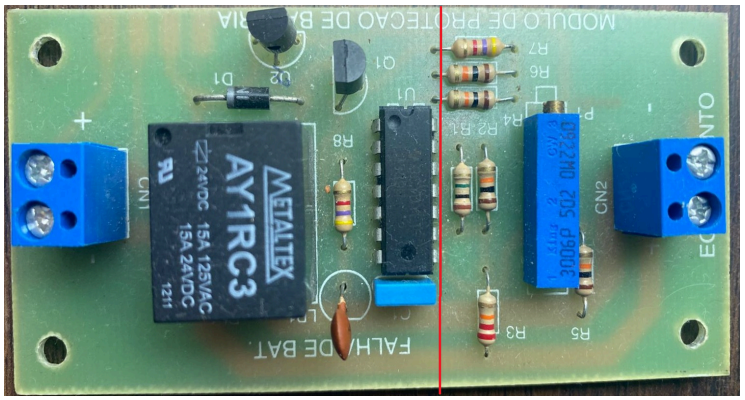

Figura 5: Placa eletrônica utilizada como protótipo. Fonte: Autor

4.2 ESQUEMAS DE ILUMINAÇÃO

A escolha do esquema de iluminação é crucial para a detecção eficiente de defeitos em placas de circuito impresso (PCBs), uma vez que diferentes técnicas destacam características específicas dos componentes e anomalias.

Para gerar um volume significativo de imagens sob diversas condições de iluminação, utilizou-se uma luminária simples e monofocal com luz contínua. Foram disponibilizados 3 formatos de ângulos de iluminação sobre a placa: 0 graus, 45 graus à esquerda da placa, e 45 graus à direita da placa, obtendo luz total aos componentes, e luz parcial (nos casos de 45 graus), tendo assim sombra sobre alguns componentes. A iluminação seguiu a técnica de campo claro, com posicionamento angular que permitiu a captura direta da maior parte da luz refletida pela câmera. Adicionalmente, empregou-se a técnica de campo escuro, na qual ângulos rasos de incidência luminosa produzem sombras pronunciadas, simulando cenários com componentes irregulares e sobreposição de sombras. A figura 6 explica a estrutura base de iluminação, em conjunto com a câmera, para capturar as imagens.

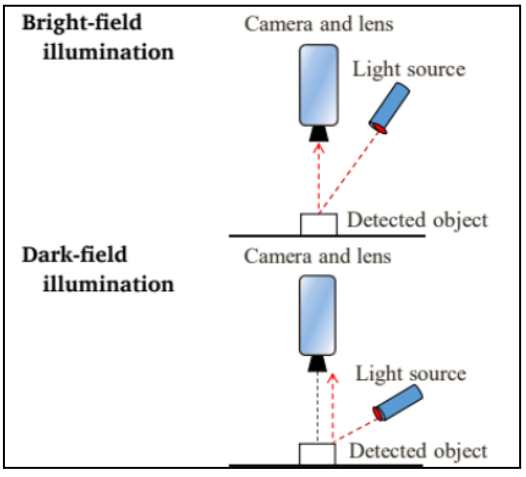

Figura 6: Esquema de iluminação de campo aberto e de campo fechado. Fonte: Adaptado de Zhou et al., 2023

4.3 CAPTURA DE IMAGENS

Para o desenvolvimento do modelo, a captura de imagens requer uma câmera com lente, onde a qualidade da lente dirá muito sobre a qualidade das fotos, a necessidade de tratamento, e a complexidade das características das imagens ao que se querem localizar e classificar. Portanto, uma boa resolução aliada a uma boa iluminação traz resultados eficazes, na medida do possível, ainda mais quando se trata de algo pequeno e com múltiplos formatos, como o caso da placa eletrônica.

Para tal, está sendo utilizada uma câmera de um Iphone 11, de 12 megapixels, com uma resolução de 4000x3000 pixels, aliada a um suporte de madeira, que garanta estabilidade e precisão nas imagens, a uma distância focal no eixo Z de 10cm, com zoom de 2x na lente. Essa distância focal da imagem no ângulo Z, que deve ser o mesmo independente da imagem, pois afeta o cálculo do distanciamento dos componentes.

A figura 7 mostra o resultado do protótipo de apoio para a câmera e para a placa, a câmera sobre o andar, com a lente sobre o buraco, virada para baixo, enquanto a placa fica logo abaixo.

Figura 7: Protótipo de ancoragem da filmadora (Iphone 11) sobre a placa. Fonte: Autor

A quantidade e qualidade das imagens a serem utilizadas nos treinamentos e validação do projeto estão entrelaçada com o desempenho da aprendizagem do modelo, fazendo com que uma amostragem maior, ainda que com custo maior de hardware para processamento, tenha resultados mais precisos. Embora quantidade e qualidade estejam diretamente relacionadas, a distribuição equilibrada das imagens entre as classes, ou seja, entre todos os tipos de componentes a serem classificados, é crucial para garantir que o modelo não apresente viés preditivo. Por exemplo, se houver 100 imagens de capacitores e apenas 10 de resistores, o modelo tenderá a ter um desempenho significativamente melhor na detecção de capacitores, enquanto os resultados para resistores serão extremamente inferiores. Essa disparidade pode comprometer a eficácia geral do sistema de inspeção, destacando a importância de um conjunto de dados balanceado para todas as classes envolvidas.

4.4 ROTULAÇÃO DA IMAGEM COM SOFTWARE LABELIMG

A anotação de imagens por classe é etapa fundamental para modelos de aprendizado de máquina, particularmente em tarefas de localização de objetos como componentes em placas de circuito impresso. O processo envolve rotular (labeling) individualmente cada componente, estabelecendo uma correlação precisa entre o objeto e suas coordenadas espaciais na imagem.

Uma das ferramentas disponíveis gratuitamente é o LabelImg, que permite criar anotações diretamente nas imagens. Para modelos YOLO, o software gera arquivos .txt contendo as coordenadas normalizadas dos objetos e suas respectivas classes. Essas coordenadas definem retângulos delimitadores (bounding boxes) que envolvem precisamente cada objeto de interesse, servindo como referência tanto para a localização e classificação dos componentes pelo modelo, quanto para a posterior validação dos resultados obtidos em relação ao esperado. A figura 8 sintetiza o software de desenvolvimento das rotulagens das imagens, demonstrando o passo a passo para criar e salvá-las.

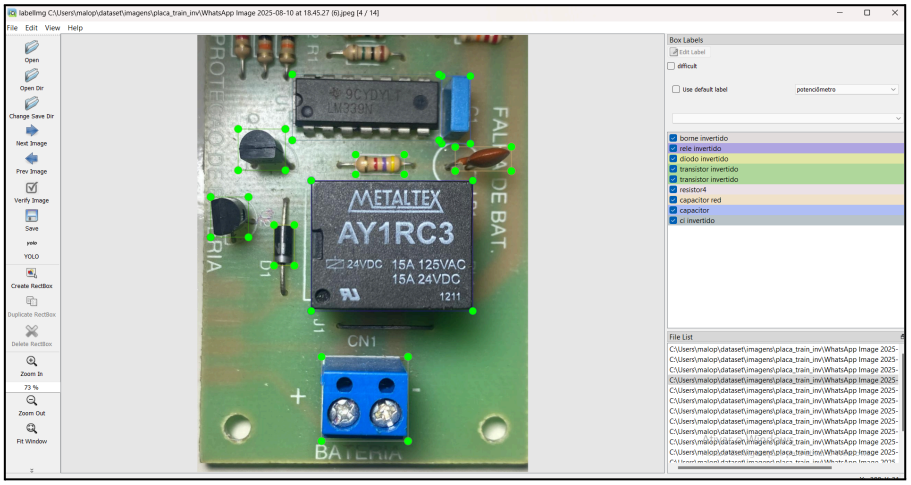

Figura 8: Desenvolvimento de labels .txt no LabelImg. Fonte: Autor

4.5 AUMENTO DE DADOS COM BIBLIOTECA IMGAUG + YOLO

A técnica de *data augmentation* expande artificialmente o conjunto de dados de treinamento através da geração de variantes realistas das imagens originais, atuando como um método eficaz de regularização que reduz o sobreajuste (overfitting). O princípio fundamental dessa abordagem reside na criação de exemplos sintéticos que preservam as características essenciais dos dados originais, de modo que, idealmente, um observador humano não consiga distinguir entre uma imagem original e uma aumentada (YANG et al., 2022).

Para estruturar essa técnica, utilizou-se a biblioteca imgaug. A sequência de aumento é configurada com uma série de transformações, incluindo ajustes de contraste linear e gama, variação de temperatura de cor, aplicação de desfoque gaussiano e adição de ruído gaussiano. Essas operações visam simular condições diferenciadas de captura de imagem.

No código, cada imagem da pasta de origem é lida e, caso possua um canal alpha, este é removido para garantir a compatibilidade com o formato RGB esperado pela maioria dos modelos de rede neural. A imagem então passa pela sequência de transformações definida, criando uma nova versão artificialmente expandida. Para verificação visual imediata, cada imagem aumentada é exibida em tela com a biblioteca matplotlib, e então, salva adicionalmente no *dataset* para modelagem.

A Divisão de Imagens, fora proposta em dois *dataset*, Train, relacionado ao treinamento, com 84 imagens (843 componentes), enquanto a Val, relacionada a validação dentro da modelagem, ficou com 36 imagens (374 componentes). No código do modelo, existe ainda um aumento de dados, de forma a auxiliar no aumento de dados, incluindo variação de cores (hsv), inversão vertical/horizontal (flip) e combinação de imagens (mosaic e mixup).

4.6 AMBIENTE DE DESENVOLVIMENTO

Todo o ambiente de programação Python foi desenvolvido no Google Colaboratory™, uma plataforma de computação em nuvem, baseada em Jupyter Notebooks. Sua vantagem em relação à computação local, reside no acesso gratuito, a recursos computacionais de alto desempenho, como GPUs (Unidades de Processamento Gráfico) e TPUs (Unidades de Processamento Tensor). Esses recursos aceleram tarefas exigentes, como treinamentos e inferências de modelos de aprendizado de máquina, como no caso da detecção de componentes eletrônicos com o YOLOv10.

Com a utilização dos servidores Google, em vez de infraestrutura local, há uma simplificação no setup do ambiente de programação, uma vez que as bibliotecas de ciências de dados e de Inteligência artificial, já estão pré-instaladas, agilizando todo o processo de estruturação dos códigos.

5 TREINAMENTO DO MODELO

Para o desenvolvimento do modelo, a partir da captura e do pós-tratamento das imagens, é implementado o código do YOLOv10n, a versão nano da arquitetura YOLOv10. Essa variação do YOLO foi escolhida por seu bom desempenho, por ser consolidada e por ser menos exigente computacionalmente, fatores importantes para viabilizar o treinamento mesmo em ambientes com recursos limitados, como este caso.

O primeiro passo da codificação consiste no pré-processamento por meio de um arquivo YAML, que especifica os caminhos para os *datasets* de treinamento e validação, os nomes e a quantidade de classes de componentes eletrônicos, além da estrutura de diretórios onde estão armazenadas as imagens e suas respectivas anotações. A figura 9 expõe o código de pré-processamento utilizado no modelo.

```yaml
path: /content/drive/MyDrive/TCC_Carlos/Yolov10_final/dataset
train: images/train
val: images/val
test: images/test

names:
  0: capacitor red
  1: ci
  2: ci invertido
  3: capacitor
  4: diodo
  5: diodo invertido
  6: placa
  7: rele
  8: rele invertido
  9: transistor
  10: transistor invertido
  11: potenciometro
  12: potenciometro invertido
  13: borne
  14: borne invertido
  15: resistor1
  16: resistor2
  17: resistor3
  18: resistor4
```

Figura 9: Código de pré-processamento, YAML. Fonte: Autor

Em seguida, são definidos os parâmetros de treinamento. Foram 1000 épocas de iterações setadas, tamanho de lote de 16 imagens por atualização, redimensionamento das imagens de 640x640 pixels, uso do otimizador AdamW com decaimento de peso de 0,05, taxa de aprendizado inicial de 1e-3 que se reduz até 1e-4 ao final do treino, e uma paciência de 100 épocas para interrupção antecipada, caso não haja melhoria significativa.

Também foram aplicadas técnicas de aumento de dados diretamente no código de treinamento, complementando o trabalho anterior de expansão do conjunto de imagens. Essas técnicas incluem modificações no espaço de cores HSV, com ajuste de matiz em 0,015, saturação em 0,7 e valor em 0,4, flip vertical e horizontal, cada um com probabilidade de 0,8, mosaic, que combina quatro imagens com probabilidade de 1,0 e mixup, que realiza mistura entre imagens com probabilidade de 0,5.

O próximo passo é a implementação propriamente dita, utilizando um modelo pré-treinado do repositório do *GitHub* de Jameslahm (yolov10n), que já contém pesos aprendidos a partir do *dataset* COCO. Essa abordagem de transfer learning (transferência de aprendizado) é importante para adaptar o conhecimento geral de detecção de objetos à tarefa específica de reconhecimento de componentes eletrônicos.

O processo de treinamento se inicia com um período de *Warmup* de três épocas, que estabiliza o aprendizado nas fases iniciais. A função de perda utilizada combina três componentes: perda de localização (box) com peso 7,5, perda de classificação (cls) com peso 0,9 e perda de distribuição focal (dfl) com peso 1,5. Essa combinação busca um equilíbrio entre a precisão na localização dos componentes e a acurácia em sua classificação.

Nas configurações avançadas, o *mixed precision* (amp) é ativado para acelerar o treinamento sem perda significativa de precisão, são utilizados oito processos paralelos (workers) para o carregamento eficiente dos dados, e a opção

single_cls é definida como False, indicando que se trata de um problema multiclasse, onde o modelo deve aprender a distinguir entre diferentes tipos de componentes eletrônicos presentes nas placas PCB.

A Figura 10 mostra os resultados da detecção de rótulos no conjunto de treino, enquanto a Figura 11 exibe as predições dos componentes no conjunto de validação. Ambos são demonstrados de forma parcial, já que existem diversas imagens que compõem o conjunto.

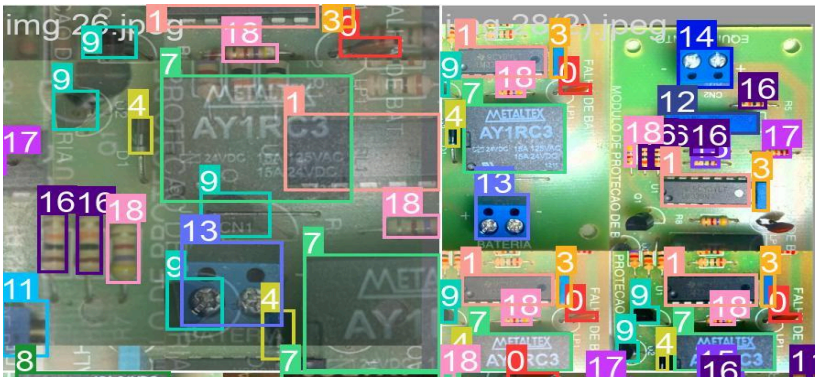

Figura 10: Resultados ampliados de detecção do treino (Train Batch). Fonte: Autor

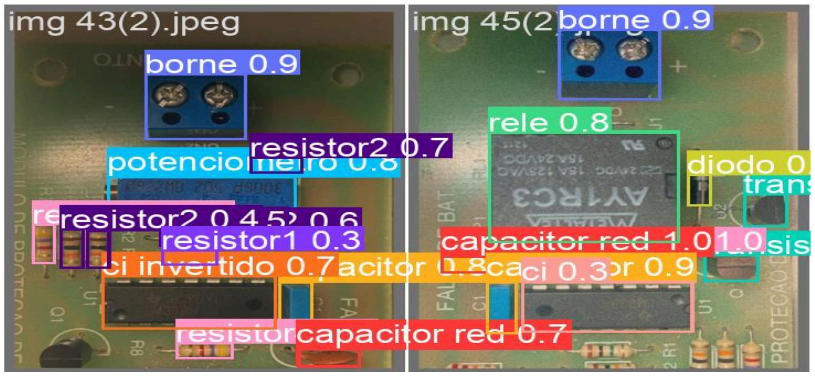

Figura 11: Resultados ampliados de predição (Validation Batch Prediction). Fonte: Autor

6 MÉTRICAS DE VALIDAÇÃO E DESEMPENHO

Em geral, o modelo mostra um bom progresso de aprendizado ao longo das épocas de treinamento, com tendências geralmente positivas em todas as métricas principais. Observa-se que as curvas começam com valores baixos nas primeiras épocas e evoluem consistentemente, estabilizando-se ao final do ciclo de treinamento (744 épocas).

Analisando as métricas específicas, como Precision (Precisão), ela atinge valores máximos de 0,897, indicando que quando o modelo detecta um componente, ele está correto em aproximadamente 89,7% dos casos. Esta é uma performance boa para aplicação industrial, pois significa que haverá poucos falsos positivos, um bom aspecto para evitar rejeição desnecessária de placas boas. Outra

métrica é a Recall (Revocação), a qual mostra valores máximos de 0,959 (95,9%), trabalhando com valores próximos a Precision. Isto sugere que o modelo é capaz de encontrar a grande maioria dos componentes presentes nas placas, minimizando o risco de componentes faltantes passarem despercebidos pelo sistema de inspeção. As figuras 12 e 13 demonstram a curva de Precision (Precisão) e Recall (Revocação) versus Confidence (Confiança) do modelo utilizado no projeto.

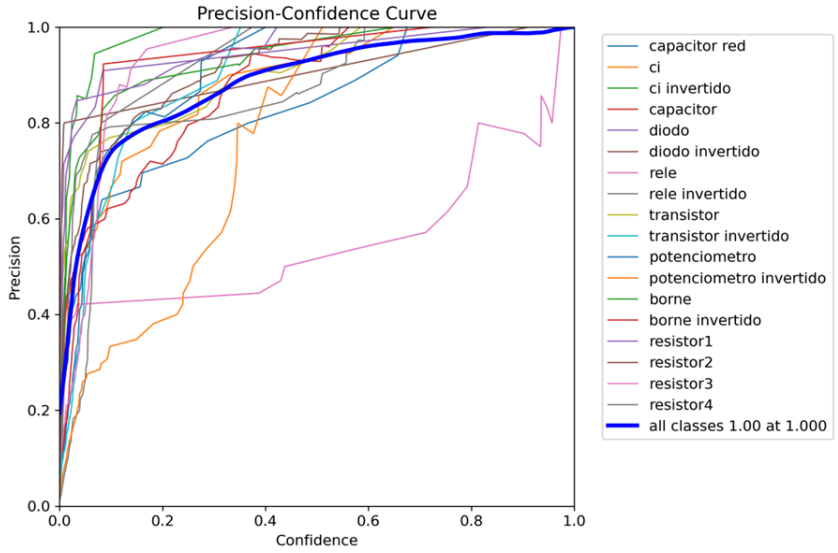

Figura 12: Curva Precision-Confidence, resultado da modelagem. Fonte: Autor

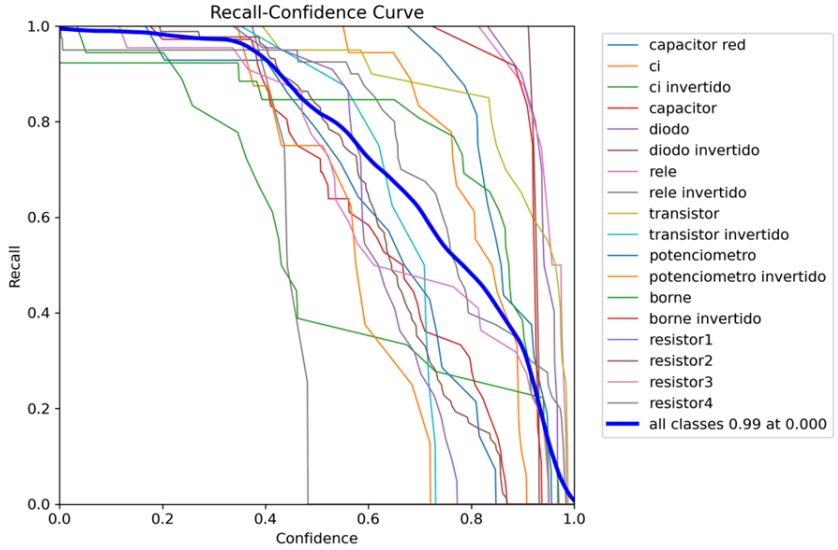

Figura 13: Curva Recall-Confidence, resultado da modelagem. Fonte: Autor

Avaliando a Capacidade de Detecção, a métrica mAP50 (mean Average Precision com IoU=0,5) apresenta a precisão média quando se considera uma sobreposição moderada entre as detecções e as anotações reais(bounding box), trabalhando com uma evolução de ~0,4 até a média geral ~0,978. Já o desempenho da mAP50-95, que avalia o modelo sob critérios progressivamente mais rigorosos de precisão na localização, encontrou uma subida de ~0,5 até a média geral ~0,875,

ainda que haja variação entre componentes, a exemplo de 0,754 para o “Resistor 4” e 0,995 para o “Relé”. O fato destas métricas atingirem valores relativamente altos demonstra que o modelo não apenas detecta os componentes, mas também os localiza com boa precisão espacial, um requisito importante para inspeção dos componentes, visto que haverá um pós processamento utilizando das coordenadas detectadas para validação. A figura 14 demonstra os principais resultados do modelo utilizado no projeto.

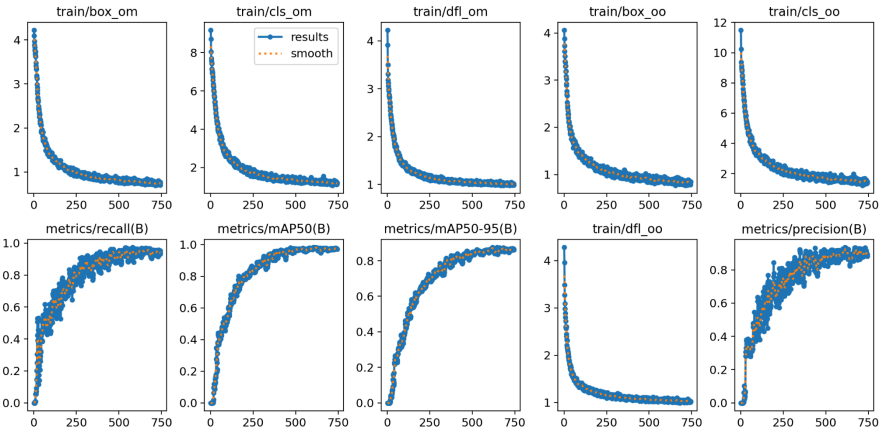

Figura 14: Principais métricas, resultado da modelagem. Fonte: Autor

O F1-Score é calculado a partir de uma média harmônica entre Precision e Recall, utilizando a fórmula F1-Score = 2 × (Precision × Recall) / (Precision + Recall). No modelo, esse indicador atingiu 92% com um limiar de confiança de 0,36, o que mostra um equilíbrio satisfatório entre precisão e revocação. Ao analisar o desempenho por classe, observam-se diferenças entre os componentes, mas todos ainda próximos de 90%, a maioria acima disso.

O limiar de 0,36 representa o ponto de operação ideal para o modelo, pois é onde ele consegue, ao mesmo tempo, detectar a maior quantidade possível de componentes presentes e reduzir ao máximo as detecções falsas. Se operar abaixo desse valor, o sistema geraria um excesso de falsos positivos. Já acima dele, deixaria de identificar componentes que de fato estão presentes. A figura 15 demonstra as curvas F1-score do modelo.

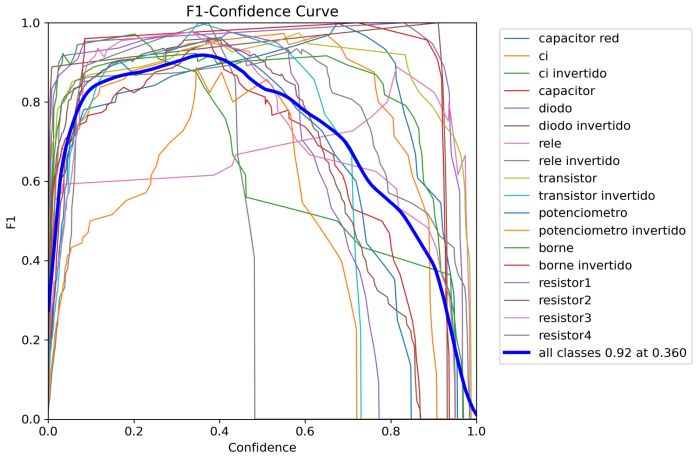

Figura 15: Resultados da F1-score nos componentes. Fonte: Autor

7 TESTES E VALIDAÇÃO DO MODELO

7.1 PREDICT

O processo de inferência do modelo YOLOv10 foi conduzido utilizando a biblioteca Ultralytics. Inicialmente, o modelo pré-treinado personalizado foi carregado a partir dos pesos do arquivo resultante do treinamento dedicado à detecção de componentes eletrônicos. Em seguida, a partir do *dataset* contendo imagens de teste inéditas (*dataset* teste), contendo 20 imagens, que não participaram da etapa de treinamento, fora usado para avaliar a capacidade de generalização do modelo. A predição foi executada com configurações que permitiram salvar tanto as imagens anotadas com as detecções quanto arquivos de texto contendo as coordenadas normalizadas das caixas delimitadoras, os identificadores das classes e os respectivos níveis de confiança. A figura 16 mostra os resultados de predição em duas imagens do *dataset* teste.

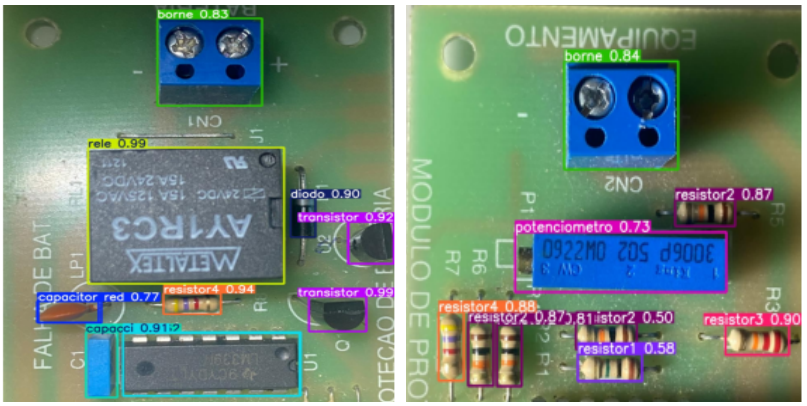

Figura 16: Resultados de predição em duas imagens do dataset teste. Fonte: Autor

Para cada imagem processada, as detecções foram individualmente analisadas, extraindo-se as coordenadas dos cantos da caixa delimitadora no espaço da imagem, a confiança da predição e o label correspondente à classe do componente identificado. Essas informações foram impressas de forma organizada, associando cada detecção à imagem de origem, o que permitiu uma verificação detalhada e imediata do desempenho do modelo. Paralelamente, a geração dos arquivos de saída garantiu que os resultados pudessem ser utilizados posteriormente para a etapa de validação de posicionamento dos componentes através de métodos geométricos. As figuras 17 e 18 demonstram as coordenadas e confiança para cada classe encontrada na figura 16, esquerda e direita, respectivamente.

```
Imagem: /content/drive/MyDrive/TCC_Carlos/Yolov10_final/dataset_teste/Teste 01.jpeg
Classe: rele, Confiança: 0.99, Caixa: (209, 439) - (690, 833)
Classe: transistor, Confiança: 0.99, Caixa: (751, 870) - (892, 971)
Classe: resistor4, Confiança: 0.94, Caixa: (393, 862) - (534, 921)
Classe: transistor, Confiança: 0.92, Caixa: (848, 653) - (959, 776)
Classe: ci, Confiança: 0.91, Caixa: (294, 973) - (731, 1157)
Classe: diodo, Confiança: 0.90, Caixa: (705, 585) - (766, 699)
Classe: borne, Confiança: 0.83, Caixa: (378, 66) - (635, 318)
Classe: capacitor red, Confiança: 0.77, Caixa: (86, 883) - (241, 946)
Classe: capacitor, Confiança: 0.62, Caixa: (204, 974) - (280, 1163)
```

Figura 17: Dados da figura 16, a esquerda, englobando a coordenada e confiança por classe encontrada. Fonte: Autor

```
Imagem: /content/drive/MyDrive/TCC_Carlos/Yolov10_final/dataset_teste/Teste 02.jpeg
Classe: resistor3, Confiança: 0.90, Caixa: (786, 828) - (931, 892)
Classe: resistor4, Confiança: 0.88, Caixa: (125, 801) - (184, 946)
Classe: resistor2, Confiança: 0.87, Caixa: (193, 821) - (252, 957)
Classe: resistor2, Confiança: 0.87, Caixa: (670, 560) - (808, 617)
Classe: borne, Confiança: 0.84, Caixa: (414, 264) - (676, 496)
Classe: resistor2, Confiança: 0.81, Caixa: (259, 826) - (317, 959)
Classe: potenciometro, Confiança: 0.73, Caixa: (302, 629) - (787, 758)
Classe: resistor1, Confiança: 0.58, Caixa: (445, 889) - (594, 948)
Classe: resistor2, Confiança: 0.50, Caixa: (441, 823) - (581, 879)
```

Figura 18: Dados da figura 16, a direita, englobando a coordenada e confiança por classe encontrada. Fonte: Autor

7.2 PADRONIZAÇÃO DAS LOCALIZAÇÕES DOS COMPONENTES

Para ter os valores definitivos de posicionamento dos componentes, o processo de análise de posicionamento de componentes tem início com uma lista pré-definida de detecções, entendendo que está é a posição “perfeita”. Cada elemento contém a classe do componente e as coordenadas de sua caixa delimitadora, dados esses recolhidos da função “Predict”, já relatada anteriormente. Definiu-se então o potenciômetro, ou relé, dependendo da imagem, como ponto de referência para a análise, localizando suas coordenadas na lista de detecções.

Para todos os demais componentes, calculou-se o centro geométrico de suas caixas delimitadoras, determinando assim sua posição absoluta em pixels na imagem. A partir do centro do potenciômetro/relé, computou-se o deslocamento relativo de cada componente, tanto no eixo X quanto no eixo Y, gerando um vetor de distância em pixels a partir da referência. Essas informações foram consolidadas em uma nova estrutura de dados para cada componente, contendo sua classe, centro absoluto, centro relativo ao potenciômetro, coordenadas da caixa e confiança original.

Os resultados foram impressos, exibindo os deslocamentos relativos de cada item em relação ao ponto de referência. Para visualização, plotou-se um gráfico de dispersão onde a posição absoluta de cada componente foi marcada, conectando-se cada ponto ao centro do potenciômetro através de uma linha tracejada. O potenciômetro foi destacado em vermelho com um marcador quadrado, enquanto os demais componentes foram legendados com suas respectivas classes e valores de deslocamento. O gráfico proporciona uma representação visual clara da distribuição espacial dos componentes em relação ao ponto de referência estabelecido.

A figura 19 retrata o modelo base de coordenadas a ser buscada nos testes com o *dataset* Teste.

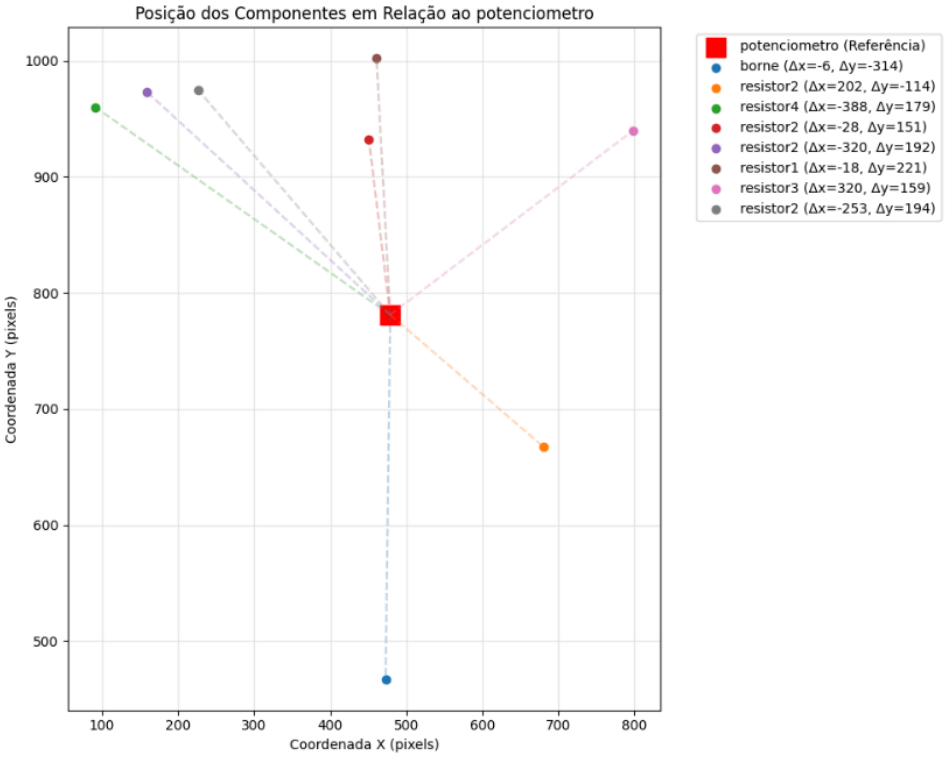

Figura 19: Modelo ideal de coordenadas para os componentes. Fonte: Autor

7.3 VALIDAÇÃO DO POSICIONAMENTO DOS COMPONENTES

Nessa fase, o código realiza uma análise de posicionamento de componentes eletrônicos detectados na placa eletrônica, utilizando um potenciômetro, ou um relé, como ponto de referência central. O processo inicia com uma lista de detecções contendo informações sobre cada componente identificado: classe, nível de confiança da detecção e coordenadas da caixa delimitadora (bounding box) no sistema original de coordenadas da imagem, sendo os dados recolhidos da função predict, agora utilizando o *dataset* teste.

O algoritmo estabelece um novo sistema de coordenadas cartesianas tendo o potenciômetro ou relé como origem (ponto 0,0). Para cada componente detectado, calcula-se o centro de sua caixa delimitadora e, em seguida, converte essas coordenadas absolutas em coordenadas relativas, que representam o deslocamento horizontal e vertical em relação ao Potenciômetro/Relé.

A validação do posicionamento é realizada comparando as posições detectadas com um mapa de posições esperadas pré-definido para cada tipo de componente, desenvolvido anteriormente. Para cada instância detectada, o código calcula a distância euclidiana em pixels até a posição esperada mais próxima. Com base em uma tolerância pré-estabelecida de 20 pixels, cada componente é classificado com status “OK” (dentro da tolerância) ou “FORA” (acima da tolerância), com a distância real explicitada.

Os resultados são consolidados em um relatório detalhado, que apresenta para cada componente sua instância, posição detectada, posição esperada mais próxima, distância calculada, status de posicionamento e o nível de confiança da detecção original. Este relatório permite uma verificação da conformidade do layout da placa.

Finalizando, é gerado uma visualização gráfica, onde o potenciômetro ou relé é plotado no centro do gráfico. Cada componente é representado por um marcador de cor e formato específicos, com anotações que identificam sua classe e instância. O gráfico, com seu sistema de eixos cruzando na origem (potenciômetro/relé), fornece uma representação espacial intuitiva da disposição geral dos componentes na placa em relação ao ponto de referência, servindo como uma ferramenta para inspeção de qualidade e controle de layout da placa. As figuras 20 e 21 trazem os resultados das posições encontradas na figura 16 (direita), garantindo que o resultado foi dentro do esperado (aba de Status dentro da tabela) para todos os componentes, com variação de posição dentro do limite de 20 pixels.

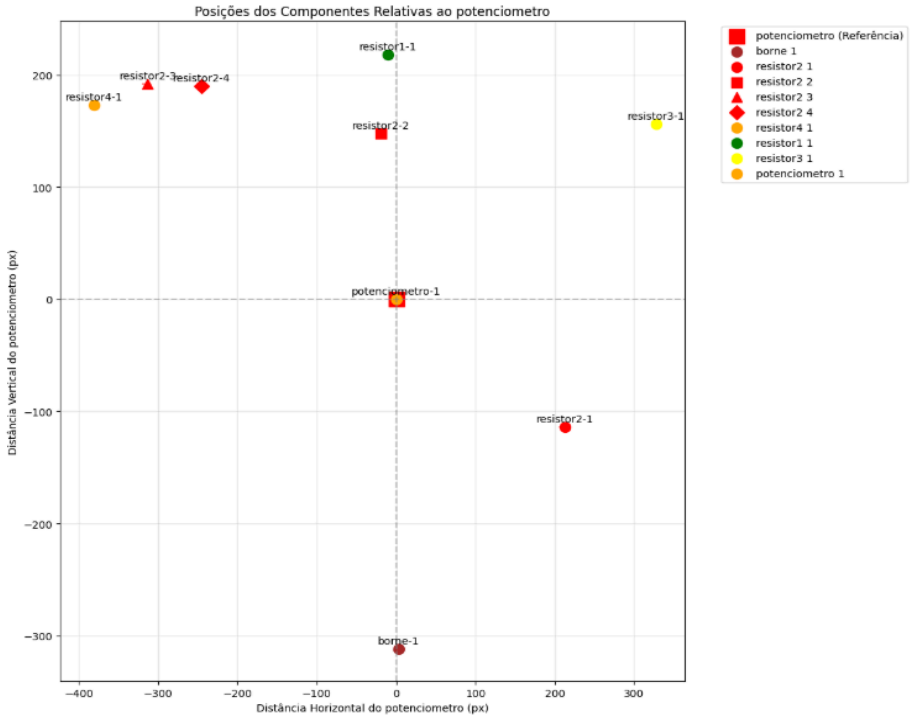

Figura 20: Posições das coordenadas encontradas, utilizando os resultados da figura 16 (direita). Fonte: Autor

REFERÊNCIA: potenciometro em (544, 693) (sistema original)

Componente	Instância	Posição Detectada	Posição Esperada	Distância	Status	Confiança
borne	1	(1, -313)	(-6, -314)	7.1	OK	0.84
resistor2	1	(-322, 196)	(-320, 192)	4.5	OK	0.87
resistor2	2	(195, -105)	(202, -114)	11.4	OK	0.87
resistor2	3	(-256, 199)	(-253, 194)	5.8	OK	0.81
resistor2	4	(-33, 158)	(-28, 151)	8.6	OK	0.50
resistor4	1	(-390, 180)	(-388, 179)	2.2	OK	0.88
resistor1	1	(-25, 225)	(-18, 221)	8.1	OK	0.58
resistor3	1	(314, 167)	(320, 159)	10.0	OK	0.90
potenciometro	1	(0, 0)	(0, 0)	0.0	OK	0.73

Figura 21: Comparação entre as posições das coordenadas encontradas na figura 20 versus a esperada (dados da figura 19). Fonte: Autor

8 CONCLUSÃO E ESTUDOS FUTUROS

Entendendo o gargalo na manufatura causado por falhas na inspeção de placas PCB, o mercado carece de alternativas viáveis, e de certa forma, de baixo custo, para que se construa uma ponte entre velocidade e precisão. A abordagem presente no estudo é um primeiro passo para uma solução complexa, visto que existem diversos defeitos dignos de validação dentro da placa. Para muitos deles, a modelagem YOLO também pode ser aplicada, utilizando partes das técnicas utilizadas até aqui.

O estudo trouxe uma técnica de visão computacional adaptada para uma solução de detecção de componentes, fundindo modelagem YOLO para detecção, com os cálculos de distanciamento euclidiano para análise espacial. Dentro do esperado, as métricas de precisão e revocação se mostraram eficazes, ainda que existam alguns aspectos a serem melhorados, como mostrado nos resultados de mAP, onde houve um distanciamento entre alguns componentes, tendo resultado extremamente positivos para alguns, enquanto outros se mostraram razoáveis, ainda que a média geral seja muito boa.

Alguns aspectos dentro do modelo, trazem atenção quanto ao seu uso. Todos os componentes utilizados, trazem diferenças reais de cores e formatos, o que é imprescindível para validação dos resultados. Uma vez que componentes com o mesmo encapsulamento sendo usados na mesma imagem, ocorrerão problemas na detecção.

Para estudos futuros, o sistema proposto neste artigo representa uma das etapas no desenvolvimento de um sistema robusto de controle de qualidade. Além das soluções apresentadas, existe uma gama de outras a serem incorporadas, como a detecção de componentes com defeitos visíveis. Essa combinação resultaria em uma solução completa focada em componentes. Tais sistemas ainda necessitam ser implementados em controladores como Raspberry Pi e afins, para que possam ser utilizados na prática.

REFERÊNCIAS

Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J., & Shen, F. (2022). Image Data Augmentation for Deep Learning: A Survey. https://arxiv.org/abs/2204.08610v2

Hu, S, Zhang, X, Liao, H, Liang, X, Zheng, M, & Behdad, S. “Deep Learning and Machine Learning Techniques to Classify Electrical and Electronic Equipment.”

Y. Zhou, M. Yuan, Zhang, G. Ding, & Qin, S. (2023). “Review of vision-based detection and its perspectives for printed circuit board.”

J. Chen, E. Bao, & Pan, J. (2022). “Classification and positioning of Circuit Board Based on Improved YOLOv5.”

S. Luo, F. Wan, G. Lei, L. Xu, Z. Ye, W. Liu, W. Zhou & Xu, C. (2024). “EC-YOLO:Improved YOLOv7 Model for PCB Eletronic Component Detection.”

Harshitkumar Ghelani. (2024). “AI-Driven Quality Control in PCB Manufacturing: Enhancing Production efficiency and Precision.”

Patel, P., & Thakkar, A. (2020). The upsurge of deep learning for computer vision applications. International Journal of Electrical and Computer Engineering (IJECE), 10(1), 538–548. https://doi.org/10.11591/ijece.v10i1.pp538-548

Géron, Aurélien. Mãos à obra: Aprendizado de máquina com Scikit-Learn, Keras e TensorFlow. 2. ed. Rio de Janeiro: Alta Books, 2021.

Jian, T. S., Fauadi, M. H. F. M., Yahaya, S. H., Noor, A. Z. M., & Saptari, A. (2024). "A deep learning approach for automated PCB defect detection: A comprehensive review."

Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016

LIDAK, Gerson. Controle de Qualidade e a Redução do Tempo de Set-up em Linhas de Montagens SMT. 2005. 110 f. Dissertação (Mestrado em Engenharia de Produção e Sistemas) - Programa de Pós-Graduação em Engenharia de Produção e Sistemas, Pontifícia Universidade Católica do Paraná, Curitiba, 2005.

CARLOS EDUARDO DE SOUZA BRÜMMER

ESTUDO DA APLICAÇÃO DE ALGORITMO DE VISÃO YOLO NA INSPEÇÃO DE COMPONENTES ELETRÔNICOS EM PLACAS DE CIRCUITO IMPRESSO

Monografia apresentada ao curso de Engenharia Elétrica do Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina, para obtenção do título de Bacharel em Engenharia Elétrica.

Joinville, 12 de dezembro de 2025.

Msc. Stefano Romeu Zeplin
Orientador
Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina

Msc. Joice Luiz Jerônimo
Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina

Dr. Rodrigo Coral
Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina