**UNIVERSIDADE ESTADUAL DE CAMPINAS**

FACULDADE DE TECNOLOGIA

RODRIGO JUN MATSUMURA OTANI

# SEGMENTAÇÃO E ANÁLISE DE ELEMENTOS DE PLACAS DE CIRCUITO IMPRESSO

LIMEIRA, 29 DE NOVEMBRO DE 2019

RODRIGO JUN MATSUMURA OTANI

# SEGMENTAÇÃO E ANÁLISE DE ELEMENTOS DE PLACAS DE CIRCUITO IMPRESSO

Trabalho de Conclusão de Curso submetido à Faculdade de Tecnologia da Universidade Estadual de Campinas, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Telecomunicações.

Orientador: Rangel Arthur

LIMEIRA, 2019

**UNIVERSIDADE ESTADUAL DE CAMPINAS**

FACULDADE DE TECNOLOGIA

RODRIGO JUN MATSUMURA OTANI

# SEGMENTAÇÃO E ANÁLISE DE ELEMENTOS DE PLACAS DE CIRCUITO IMPRESSO

**Banca Examinadora:**

Prof. Dr. Rangel Arthur
FT/UNICAMP

Profª Drª Talía Simões dos Santos Ximenes
FT/UNICAMP

Prof. Dr. Leandro Rochini Ximenes
FT/UNICAMP

LIMEIRA, 29 DE NOVEMBRO DE 2019

## AGRADECIMENTOS

Agradeço aos meus pais, que me ensinaram o respeito e a convivência harmoniosa dentro da sociedade. Sempre disponíveis para suportar e compartilhar os momentos do meu progresso de vida.

À Unicamp, que me proporcionou uma alavanca de desenvolvimento social e técnico. Foi onde consegui vivenciar cenários de desafios inéditos em minha vida e consegui tratá-los como ganhos de experiência para diversas novas adversidades que surgirem adiante.

Ao professor Rangel Arthur, que, desde quando ingressei na Faculdade de Tecnologia da Unicamp, vem me apoiando no aprendizado de novas tecnologias. Iniciando com as disciplinas dos primeiros semestres, em seguida a iniciação científica, a monitoria e, por fim, este trabalho de conclusão de curso, acredito que a relação de parceria vem se fortalecendo.

## RESUMO

Este trabalho visa aplicações voltadas para soluções de um assunto contemporâneo envolvendo tecnologia, a Indústria 4.0. Desenvolveu-se um sistema por processamento de imagens, visão computacional e aprendizado de máquina voltado para classificação e validação de placas de circuito impresso e agregados. Inicialmente, foi feito um estudo de conceitos que envolviam processamento de imagens clássico, processamento de imagens por meio de ferramentas baseadas em morfologia matemática e soluções clássicas de visão computacional. Em seguida, foram abordados conceitos e aplicações envolvendo descritores de características, classificadores e ferramentas baseadas em aprendizagem profunda de máquina. Por fim, foi dado ênfase a aplicações voltadas para identificação e classificação de objetos presentes em placas de circuito impresso. O estudo se direcionou na implementação do sistema utilizando a plataforma Anaconda, a linguagem Python e as bibliotecas OpenCV e TensorFlow.



**Palavras-chave:** placa de circuito impresso, visão computacional, aprendizado profundo, Python, Indústria 4.0

## ABSTRACT

This work aims applications focused on solutions of a contemporary subject involving technology, the Industry 4.0. An image processing, computer vision and machine learning system for the classification and validation of printed circuit boards and aggregates was developed. Initially, a study of concepts involving classical image processing, image processing using tools based on mathematical morphology and classical computer vision solutions was made. Then, concepts and applications involving characteristics descriptors, classifiers and tools based on deep machine learning were discussed. Finally, emphasis was given to applications aimed at identifying and classifying objects in printed circuit boards. The study focused on the implementation of the system using the Anaconda platform, Python language and OpenCV and TensorFlow libraries.



**Keywords:** printed circuit board, computer vision, deep learning, Python, Industry 4.0

## LISTA DE FIGURAS

## LISTA DE ABREVIATURAS E SIGLAS

AI – *Artificial Inteligence*; Inteligência Artificial

AOI – *Automated Optical inspection*; Inspeção Óptica Automatizada

CI – Circuito Integrado

cm – Centímetro

CNN – *Convolutional Neural Network*; Rede Neural Convolucional

CPU – *Central Processing Unit*; Unidade Central de Processamento

GB – Gigabyte

GPU – Graphics Processing Unit; Unidade de Processamento Gráfico

HOG – *Histogram of Oriented Gradient*; Histograma de Gradientes Orientados

M2M – *Machine-to-Machine*; Máquina-Máquina

mAP – *mean Average Precision*; “média Precisão Média”

MB – Megabyte

OCR – *Optical Character Recognition*; Reconhecimento Óptico de Caractere

PCB – *Printed Circuit Board*; Placa de Circuito Impresso

pol – Polegada

RAM – *Random Access Memory*; Memória de Acesso Aleatório

**SUMÁRIO**

# 1 INTRODUÇÃO

Durante cerca de 260 anos a partir da primeira revolução industrial, com os processos de manufatura por meio de máquinas mecânicas e a transição das atividades artesanais, o propósito da indústria sempre visou alcançar uma maior eficiência produtiva. Iniciando-se com a mecanização por meio de motores à base de água ou vapor, passou-se para o aproveitamento da energia elétrica e para o avanço da eletrônica, surgindo, recentemente, o conceito de Indústria 4.0 ou Quarta Revolução Industrial (PERASSO, 2016). Ela propõe uma solução na qual se apropria da evolução da internet das coisas e da computação em nuvem, justamente quando se vivencia a era da informação.

Dentro das segmentações que se vêm desenvolvendo na área da tecnologia da informação, um dos responsáveis mais impactantes dessa nova era, podem-se destacar a interação máquina-máquina, também conhecido como M2M, do inglês *machine-to-machine*, e a inteligência artificial (AI, do inglês *artificial intelligence*).

O primeiro indica a performance de equipamentos eletrônicos conectados em rede, sem assistência humana. Essa relação entre máquinas otimiza o tempo de processos, uma vez que as tarefas de cada equipamento se conectam e tornam o fluxo de trabalho mais abrangente, com a homologação prévia do ser humano. Já o segundo também traz a independência do homem, com a inclusão da capacidade de realizar inferências para otimização do resultado. De acordo com Stuart J. Russell e Peter Norvig (1995, p. 49, tradução nossa), seu comportamento se baseia no estudo de *agentes ideais*, que são “aquilo que percebe e age em um ambiente”, de forma que “sempre executa a ação que é esperada para maximizar a medida de desempenho, dada a sequência de percepção que se observou até agora”. A inteligência artificial pode dar suporte ao funcionamento do M2M, como visto em casos como o automobilismo, que possui dados de telemetria – um caso de comunicação máquina-máquina – de mais de 60 anos, contribuindo para acumular insumos para um aprendizado de máquina robusto e geração de predições mais inteligentes (AWS, 2018, tradução nossa).

Para o sucesso da indústria 4.0, a implementação de inteligência computacional é essencial. Entretanto, continua necessário o uso de materiais até então usados, principalmente a energia elétrica e a eletrônica, bases para todo o ecossistema desta nova geração. Com isso, itens como placas de circuito impresso

(PCB, do inglês *printed circuit board*) são tão importantes quanto softwares de AI, afinal tais placas compõem a maioria dos equipamentos eletrônicos, presentes tanto para uso pessoal e doméstico, como para comercial e industrial.

Desse modo, a difusão da era da informação será mais orgânica com a maior escala de produção de PCBs e, para aumentar a eficiência produtiva, a inclusão de inteligência artificial no processo de fabricação é bem-vinda, criando-se mutualidade.

## 1.1 PROBLEMA E MOTIVAÇÃO

A produção em larga escala de PCBs requer uma alta eficiência e alto controle de qualidade. Com a demanda de equipamentos cada vez menores, com maior capacidade de processamento e menor consumo de energia, comparados com os equipamentos passados ou atualmente usados, surge o desafio de miniaturizar os componentes internos, tais como conectores, elementos acoplados (resistores, capacitores, indutores) e as próprias placas. Assim, é preciso garantir que se evitem falhas tanto no processo de produção quanto no funcionamento final do aparelho.

Para que essa garantia ocorra, é preciso haver consistência, precisão e repetibilidade, as quais são atividades de alta dificuldade para um processo humano. Isso pode ser resolvido por meio de inspeção automatizada por visão computacional (LETA et al., 2005, tradução nossa). Esse sistema de inspeção, também conhecida como AOI (do inglês *automated optical inspection*), “pode adquirir milhões de pontos de dados (pixels) em uma fração de segundo” (TAHA;EMARY;MOUSTAFA, 2014, p. 1095, tradução nossa). Com isso, esses dados servem para obter medidas com alta precisão e velocidade.

Além dos fatores de otimização industriais, deve-se levar em conta o ciclo dos materiais que compõem tais dispositivos. Conforme senso comum e levantado por Pramerdorfer e Kampel (2015), placas de circuito impresso possuem materiais tóxicos e com considerável quantidade de metal, que podem ser reciclados ou descartados de forma correta, contribuindo para a ecologia e a economia. Assim, a implementação de visão computacional para inspeção também pode auxiliar processos de reciclagem, para aumentar a quantidade de coleta de materiais que podem retornar para a indústria, alavancando mais a produtividade e a Indústria 4.0. No Brasil, existe esse conceito, nomeado de “logística reversa” e determinado pela Lei 12.305/2010 (BRASIL, 2010).

## 1.2 OBJETIVOS

### *1.2.1 Objetivo geral*

Identificar a presença de uma placa de circuito impresso (PCB), contabilizando, de forma automática por meio de ferramentas de visão computacional, a presença de componentes agregados.

### *1.2.2 Objetivos Específicos*

Visando atingir o objetivo geral, vale separar alguns tópicos de estudo, planejamento e ação:

- Visão computacional: compreender a evolução do aprendizado de máquina sobre a identificação de objetos por imagens ou vídeos.
- Classificação: atribuir um algoritmo capaz de detectar PCBs e alguns componentes principais.
- Aprendizado de máquina: realizar testes e treinos com diversas amostras de placas para aprimorar o entendimento de máquina, utilizando conceitos de *deep learning* (aprendizado profundo).
- Contabilização: após detecção dos componentes, realizar a contabilização com determinado nível de eficiência.

## 1.3 ORGANIZAÇÃO DO TEXTO

O trabalho a seguir foi organizado com a seguinte estrutura. No Capítulo 2, realiza-se uma pesquisa exploratória sobre morfologia matemática, sua aplicação na evolução histórica da visão computacional, métodos de captação de imagens, conceitos sobre inteligência artificial, realizar o relacionamento entre tais assuntos e, por fim, levantar casos de implementação de inspeção óptica automatizada. No Capítulo 3, é explicado o procedimento adotado para realizar este trabalho. Explica-se desde a forma que é interpretada uma imagem ou quadro de vídeo até alguns testes de algoritmos pré-treinados de *deep learning*. No capítulo, também inclui os

resultados encontrados nas análises. Finalmente, apresenta-se o Capítulo 5, onde traz o conteúdo de conclusão do trabalho.

# 2 LEVANTAMENTO

## 2.1 Contextualização sobre o visão computacional

Com a popularidade de uso de câmeras, ainda mais para armazenar informações semelhantes ao sentido humano da visão, surgem ideias para criação de funções substitutivas ao olho humano. Projetos sobre leitura de texto pelo computador – o OCR (do inglês *optical character recognition*) (ASIF et al., 2014) – e reconhecimento facial (WANG e DENG, 2018) são, possivelmente, os mais explorados nas últimas décadas.

Para a compreensão da morfologia matemática, primordial para a estruturação da visão computacional, realizou-se a leitura de conteúdos de Parker (2011) e Queiroz e Gomes (2006).

## 2.2 Evolução da visão computacional

### *2.2.1 Detecção tradicional*

Sobre a evolução da identificação de objetos por visão computacional, encontrou-se em 2001 o pioneirismo de detecção efetiva de faces, o algoritmo Viola-Jones (VIOLA e JONES, 2001). O sistema conseguia identificar em tempo real algumas demonstrações de face, por meio de determinação de padrões sobrepostos. Alguns anos depois, em 2005, foi apresentada, por Dalal e Triggs (2005), uma nova técnica de detecção denominada de HOG (*histogram of oriented gradient* ou histograma de gradientes orientados), que trouxe uma robustez para a detecção, já que não dependia da visualização da imagem completa, mas sim da procura orientada dos pixels com gradiente de cores. Logo, cada pixel vizinho era checado e criava-se um vetor direcionado para as regiões mais escuras. A partir do HOG, algumas outras pesquisas foram desenvolvidas, com a adaptação do algoritmo, podendo-se citar Wei et al. (2018), que elaborou um algoritmo para detecção de múltiplos veículos, e Pedersoli, Vedaldi e Gonzàlez (2011), que propuseram um processo de identificação de objetos deformados, na imagem, pela aceleração. Até esse momento, vivenciou-se a fase da detecção tradicional de objetos, quando a atualização de ajustes deveria ser feita manualmente.

### *2.2.2 Detecção de dois estágios*

Após sete anos, foi inaugurada a era do aprendizado profundo de máquina, com a apresentação de uma rede neural convolucional (CNN, do inglês *convolutional neural network*) proposta por Krizhevsky, Sutskever e Hinton (2012). O diferencial ocorreu ao utilizar o poder de processamento da unidade gráfica (GPU), capaz de processar diversas camadas convolucionadas. A partir do sucesso obtido, outras pesquisas relacionadas foram levantadas. Para essas soluções, normalmente o comportamento respeitava dois estágios: criação de propostas de regiões e classificação das regiões com possíveis candidatos.

Uma ideia aprimorada da solução de Krizhevsky foi denominada de R-CNN (*region-based* CNN), desenvolvido por Girshick et al (2013), que atribuía diversas áreas propostas (cerca de 2000) e filtrava por pesquisa seletiva. Essa pesquisa consistia em selecionar a área e passar na rede neural já treinada para classificar que tipo de objeto a região deveria ser. A partir da R-CNN, surgiram opções que melhoraram alguma performance, seja de redução de tempo, seja algum aprimoramento de precisão. Podem-se citar o Fast R-CNN (GIRSHICK, 2015), Faster R-CNN (REN et al., 2015) e FPN (*feature pyramid networks*) (LIN et al., 2016).

### *2.2.3 Detecção de um estágio*

Por outro lado, houve pesquisas que direcionaram para caminho divergente à rede convolucional de Krizhevsky. Foram os casos dos algoritmos YOLO (*you only look once*, ou “você apenas observa uma vez” em tradução literal) (REDMON, DIVVALA, *et al.*, 2015) e SSD (*single shot multibox detectors*, ou “detectores de múltiplas caixas em único disparo” em tradução livre) (LIU, ANGUELOV, *et al.*, 2016).

Nesse caso, a observação era feita apenas uma vez, com uma divisão uniforme de áreas na imagem, cada uma responsável por formar caixas de contorno capazes de detectar os objetos da imagem como um todo. A atividade dependendo apenas de uma visão aumentou a velocidade de processamento em relação aos algoritmos voltados para a detecção de dois estágios e trouxe uma autonomia extremamente maior em relação aos algoritmos de detecção tradicional.

### *2.2.4 Inspeção óptica automatizada em PCBs*

Ao conhecer as metodologias aplicadas para a manipulação da visão computacional, uma das aplicabilidades é a inspeção óptica automatizada (AOI). Leta, Feliciano e Martins (2008) realizaram um procedimento de segmentação de itens relevantes em placas de circuito impresso, com destaque à presença de erros presentes e suas implicações no resultado dos testes. Relacionado com o artigo anterior, Taha, Emary e Moustafa (2014) destacaram uma pesquisa da fabricação de PCBs e sua relação com AOI. Retomaram quais os tipos erros (potenciais e fatais) encontrados nas placas, além de explicar os métodos de inspeção e suas categorias.

Alguns demais trabalhos valem destaque, como a verificação de erros em soldas de componentes para verificar a qualidade da junção (BENEDEK, 2011); a preocupação da reciclagem de materiais que constituem as placas como justificativa para apresentar uma solução de segmentação dos componentes para localizá-los (LI, ESDERS e BREIER, 2013); e a contabilização de componentes em certas PCBs encontradas em estado de reciclagem, com a verificação de algum padrão entre dimensões de agregados e a própria placa, para classificar o porte do equipamento (PRAMERDORFER e KAMPEL, 2015)

Com o intuito de obter resultados mais dinâmicos e robustos, Huang e Wei (2018) propuseram o consolidado do que foi estudado até então nesta seção. Foi realizada a etapa de morfologia e foi utilizada a rede neural convolutiva para treinar o algoritmo para testes de precisão. Assim como outros citados anteriormente, o foco foi a identificação de falhas nos trilhos e conectores das placas. Outro projeto relacionado com detecção de erros na placa, utilizando a solução Faster R-CNN, foi mostrada por Ding et al. (2019), utilizando alguns *frameworks* destinados para o aprendizado de máquina, como o TensorFlow. Já o uso do sistema YOLO foi explorado por Li et al. (2019). Por fim, o algoritmo denominado PCB-METAL (*PCB-Micro Electronics Taken Apart Logically*, ou “PCB-Microeletrônicos Separados Logicamente”, em tradução livre) trouxe leitura de diversos componentes e testes com algumas das principais redes neurais utilizadas (MAHALINGAM, GAY e JR., 2019).

É importante ressaltar que estas últimas aplicações dependem diretamente de uma GPU dedicada e compatível com a demanda, o que relembra a mudança do conceito de visão computacional em 2012, época que começou a se popularizar as unidades gráficas para alto processamento.

# 3 DESENVOLVIMENTO

## 3.1 Material

Para executar os testes e treinos, foi criado um ambiente de desenvolvimento que consistiu em uma CPU Intel Core i7-7700HQ, 16 GB de RAM, GPU dedicada NVIDIA GeForce GTX 1060 (4 GB de RAM para a GPU) e sistema operacional 64 bits Windows 10 Pro.

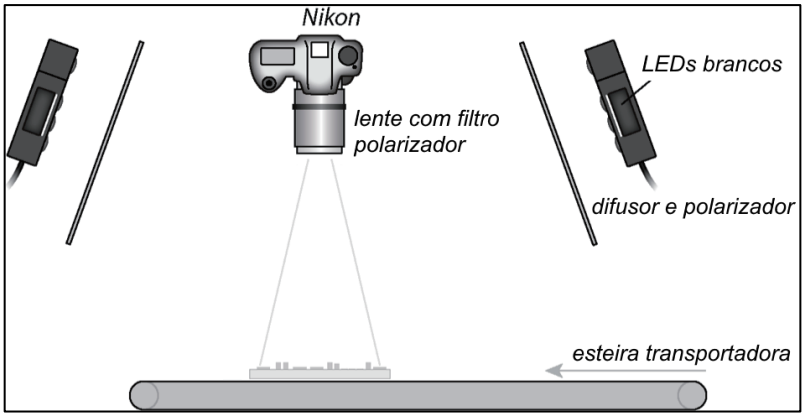
Para o funcionamento adequado da máquina para as atividades propostas, foram instalados os drivers adequados, sendo eles os drivers GeForce na versão 441.20, CUDA Toolkit 10.0, aliado com a biblioteca cuDNN 7.6.5.32. Tal configuração foi necessária para possibilitar o aprendizado de máquina com eficiência. Mesmo com soluções atuais que otimizem o processamento, ainda se torna necessário um hardware[1] capaz de executar sem falhas.

Além disso, alguns softwares para a manipulação foram devidamente instalados. São eles o Python 2.7/3.7 – usando a distribuição dedicada para ciência de dados Anaconda 3.7 – e o Visual Studio Code. Dentro do Python, além dos pacotes já inclusos na distribuição Anaconda, foram instalados alguns outros pacotes, com destaque ao OpenCV 3.4, visto que é o pacote essencial para o funcionamento da visão computacional, e ao TensorFlow, uma biblioteca de código aberto voltada exatamente para o aprendizado de máquina.

Para o banco da imagens de capturas de PCBs, foi utilizado o *dataset* (conjunto de dados) de Pramerdorfer e Kampel (2015), com cerca de 360 imagens capturadas com uma montagem de configuração constituída de uma câmera DSLR (do inglês *digital single-lens reflex*) Nikon D4 com lente 60mm f/2.8, montada a 107 cm a uma base preta com iluminação definida semelhante à ilustração da Figura 1. Essas figuras estão disponíveis para reuso em pesquisas acadêmicas.

[1] Especificações disponíveis em https://developer.nvidia.com/cuda-gpus e https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html#system-requirements

Figura 1 - Captura da imagem de PCBs




Fonte: Adaptado de PRAMERDORFER;KAMPEL, 2015

## 3.2 Metodologia

Para o desenvolvimento da metodologia, segmentou-se em algumas etapas para o entendimento e a aplicação:

- Morfologia matemática: entender o conceito matemático que estrutura a visão computacional.
- Tratamento computacional: realizar a interface inicial sobre a teoria da morfologia com algumas sintaxes de máquina.
- Características básicas de um PCB: verificar a composição de uma placa de circuito impressa básica, destacando a sua fabricação e os seus componentes. A partir disso, mapear a identificação deles para armazenar em memória que serve como base de interpretação dos algoritmos.
- Implementação de AOI: a partir da pesquisa exploratória, decidir qual solução de inspeção óptica automatizada é a mais adequada para o trabalho.
- Captação de imagens: embora citado em Material a escolha da captação das imagens, justifica-se a escolha do pacote e explica-se algumas variações possíveis de captação.
- Identificação de componentes: conseguir segmentar internamente alguns componentes e classificá-los. No trabalho, limitou-se ao estudo apenas de circuitos integrados.
- Testes e treinos: entender como é calculada a precisão e gravar os tipos de componentes, instruir a máquina para a leitura e gerar os resultados, usando a métrica mAP.

- Implementação de segmentação de objetos em imagens: utilizar as ferramentas da AOI, conciliado com o aprendizado sobre morfologia matemática e obter análises da eficiência da solução.

### *3.2.1 Morfologia matemática*

A morfologia, por definição, é a forma e a estrutura de um objeto. Assim, a morfologia matemática tem como o objetivo de reduzir, em um objeto matricial (no caso, uma imagem), os ruídos granulosos e de contornos e ampliar as conexões das regiões de contorno, definindo-se, dessa forma, a detecção eficiente de borda (PARKER, 2011). Os conceitos e aplicações das operações básicas morfológicas são a dilatação e a erosão binárias, além uma operação adicional de preenchimento das lacunas ruidosas dentro dos *blobs* (agrupamentos padrão, “bolha”).

A dilatação binária apresenta uma expansão do objeto original $A$ com um elemento estrutural $B$, resultando em $c$, como visto pela Equação 1.

$$A \oplus B = \{c | c = a + b, a \in A, b \in B\} \quad \text{Eq. 1}$$

A erosão consiste na supressão dos elementos que não compõem um padrão nem de $A$, nem de $B$ da Equação 2, sendo $c$, desta vez, o conjunto de *pixels* que compõem $B$, mas que são pretos em $A$.

$$A \ominus B = \{c | (B)_c \subseteq A\} \quad \text{Eq. 2}$$

É necessário dimensionar adequadamente o fator de dilatação e erosão, uma vez que valores exagerados para os parâmetros do projeto podem ocasionar sobreposições ou desconexões dos objetos, respectivamente.

### *3.2.2 Tratamento computacional*

A escolha da combinação Python, com o OpenCV, dentro do ambiente Anaconda, permite uma abrangência adequada para exploração dos recursos de, respectivamente, uma linguagem dinâmica e com vasta documentação; uma

biblioteca robusta e aberta sobre processamento de imagens, ideal para aprendizado e aplicabilidade; além de uma plataforma popular para ciência de dados, onde se incluem ferramentas de fácil integração ao aprendizado de máquina e aos estudos estatísticos.

Dentro do sistema Python + OpenCV, um dos procedimentos mais tradicionais para identificar o objeto por meio da morfologia matemática, é realizar os seguintes passos, com as sintaxes genéricas correspondentes:

a. Captura da imagem

```
img = img.copy()
```

b. Conversão para escala de cinza

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

c. Desfoque gaussiano

```
img = cv2.GaussianBlur(img, (x, y), 0)
```

d. Limiarização

```
imgthresh = cv2.threshold(img, x1, y1, cv2.THRESH_BINARY)[i]
```

e. Obtenção de elemento estrutural para dilatação e/ou erosão

```
element = cv2.getStructuringElement(cv2.MORPH_RECT, (a, b))
    for j in xrange(i):
        imgthresh = cv2.dilate(imgthresh, element)
        imgthresh = cv2.erode(imgthresh, element)
```

f. Definição de contorno

```
contour = cv2.findContours(imgthresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[i]
```

g. Definição de contorno convexo

```
convex_hulls = []
    for i in xrange(len(contour)):
        hull = cv2.convexHull(contour[i])
         convex_hulls.append(hull)
```

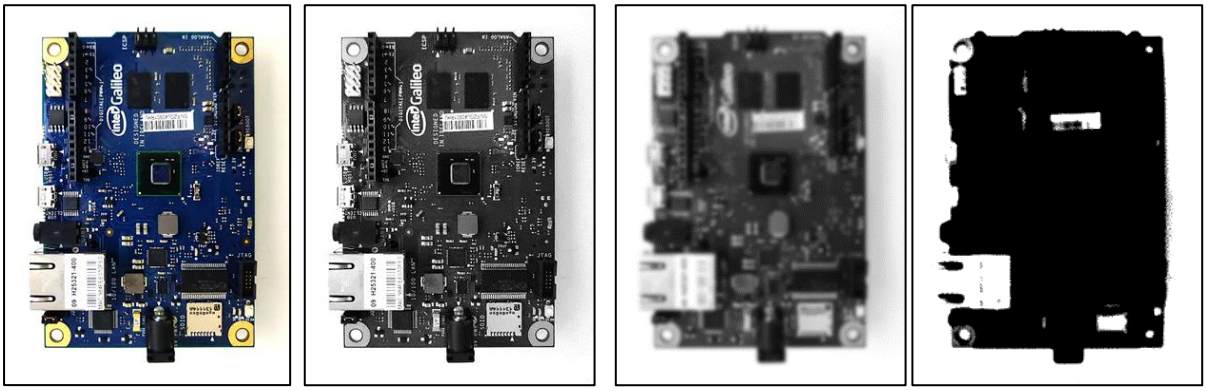
Para o devido funcionamento das linhas acima, inicialmente, é necessário importar a biblioteca do OpenCV por meio do comando:

```
import cv2
```

Vale uma breve explicação de cada etapa. (a.) simplesmente é copiada a imagem de um diretório para dentro do código, para não afetar o arquivo original. (b.) converte-se, a partir da expressão do OpenCV, a imagem original para escala de cinza, para reduzir a complexidade de se manipular matricialmente pixels com uma paleta grande de cores. (c.) realiza-se o desfoque, para atenuar alguns ruídos que pudessem interferir nas bordas das áreas de interesse. (d.) torna-se a imagem em uma matriz binária, com o efeito de limiarização, de forma que as próximas operações possam ser feitas sem maiores complexidades. (e.) a imagem binária sofre um processo de dilatação em seguida de erosão para agregar conjuntos de pixels que tendem a ser relacionados a um mesmo objeto e em seguida remover os ruídos novamente. (f.) define-se o contorno do objeto, com o algoritmo de encontrar pixels brancos (lembrando que a imagem se tornou binária) próximos e uni-los por uma delimitação que seja considerada um objeto completo. (g.) por fim, refina-se o contorno feito anteriormente, eliminando quaisquer segmentos que possuam ângulos agudos, a fim de evitar acoplamentos ou desacoplamento de objetos próximos.

A Figura 2 e a Figura 3 exemplificam as etapas de transformação da imagem captada, explicadas anteriormente, e o resultado final de uma extração de contorno apenas da placa, respectivamente.

Figura 2 - Fases de transformação da imagem capturada de uma PCB



(a) Captura original (b) Escala de cinza (c) Desfoque gaussiano (d) Limiarização

Fonte: Registradas pelo autor

Figura 3 - Captura segmentada por morfologia

Fonte: Registrada pelo autor

### *3.2.3 Características básicas de uma PCB*

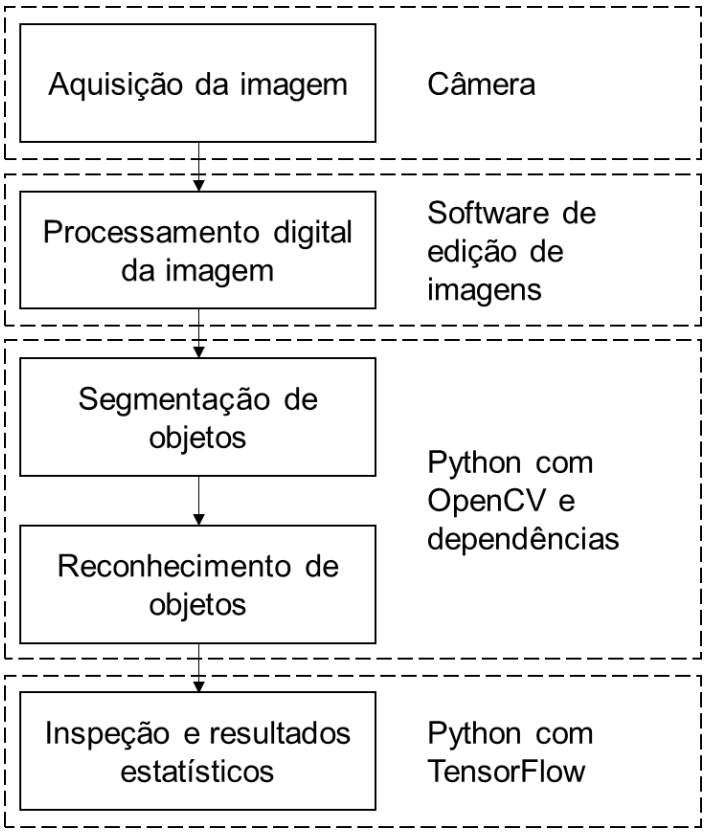
Há dois tipos de processos de montagem de uma PCB: o *Through-Hole Mounting* (THM) e o *Surface Mount Technology* (SMT). Segundo Hoof (2019), em THM, os componentes são fixados através de furos presentes nas placas, soldados até o contato de cobre existente do outro lado da placa; são vistos em resistores e capacitores. Já, na SMT, não há necessidades dos furos, visto que a junção é feita por áreas de contato (*pads*) na mesma superfície; são comuns para circuitos integrados e amplificadores operacionais.

Com essa diferenciação, esclarece-se a classificação dos componentes. Para este trabalho, o foco foi em identificar componentes SMT, com a justificativa explicada em 3.2.6.Identificação de componentes

### *3.2.4 Implementação de AOI*

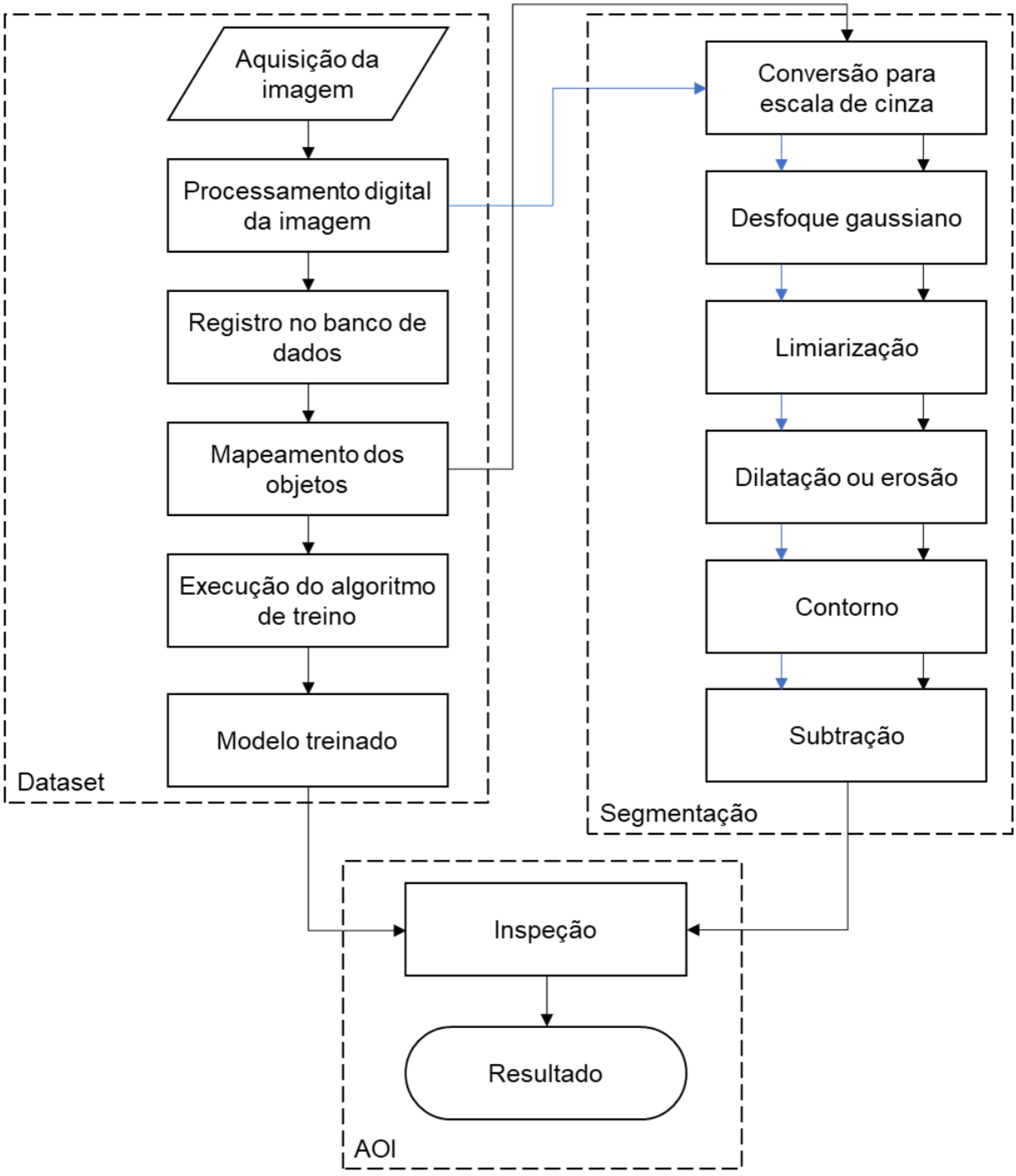
Diante das soluções apresentadas, é resumido o seguinte fluxo para o desenvolvimento do projeto, apresentado na Figura 4.

Figura 4 - Fluxograma geral do sistema AOI




O procedimento mais preciso também pode ser visto na Figura 5.

Figura 5 - Fluxograma completo do sistema AOI




O bloco “Dataset” mostra o processo que o algoritmo de *deep learning* realiza para aprender a identificar os objetos de referência. A explicação mais completa se encontra em 3.2.7.

Na “Segmentação”, há duas rotas, onde são captadas de duas etapas diferentes do “Dataset”. Os efeitos de transformação morfológicos são os mesmos para ambas, com as ações independentes entre si até a “Subtração”, onde é feita a

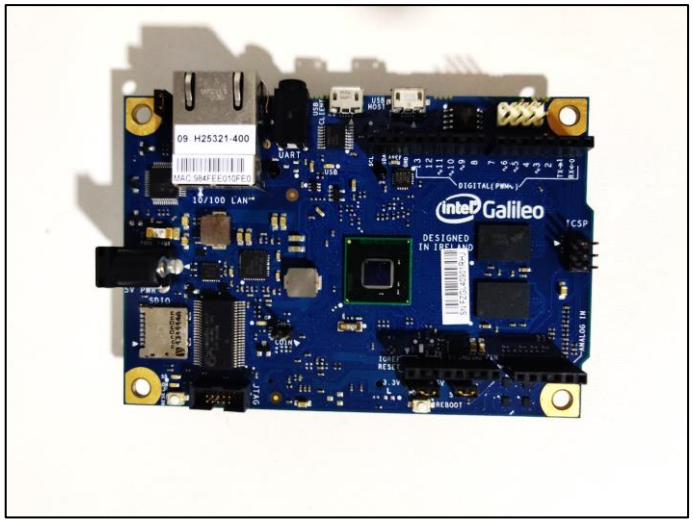
comparação das duas fontes tratadas para verificar se os objetos da imagem foram devidamente segmentados. Caso negativo, o processo de “Inspeção” realiza a detecção de falta de cadastro do modelo treinado ou falha na identificação, seja pela qualidade da imagem ou pelo fator de erro existente no código, diminuindo sua precisão.

### *3.2.5 Captação de imagens*

A aquisição de imagens é a etapa inicial para o funcionamento de todo o algoritmo. Portanto, é vital que a captura tenha o mínimo de qualidade para que os resultados não se tornem destorcidos. Assim, essa qualidade pode ser alcançada com um dispositivo fotográfico com alta capacidade de obtenção de imagem, assim como uma câmera DSLR, que permite selecionar a lente mais apropriada para o ambiente. Este que também deve estar numa situação ideal de iluminação e limpeza.

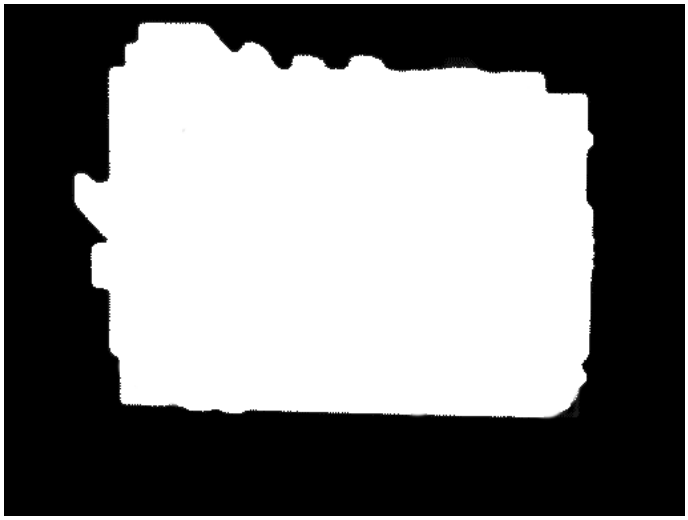
É interessante que se construa um envoltório iluminado e livre de sujeiras e poeiras ao redor do local onde se captura, visto que, segundo Mahalingam, Gay e Ricanek (2019, tradução nossa), “o sistema de iluminação assegura constante luminosidade assim evitando sombras e reflexões especulares na placa”. Isso é corroborado com exemplos de captura de uma amostra de um Intel Galilleo mostrados na Figura 6 e na Figura 2, uma com iluminação inadequada e a outra adequada, respectivamente. Em seguida, na Figura 7, foi efetuado novamente o processo de criação de contorno, explicado em 3.2.2, verificando a presença indesejada da sombra que precede os erros de análises.

Figura 6 - Imagem de placa com excesso de sombras




Fonte: Registrada pelo autor

Figura 7 - Processo de contorno com captura da sombra excessiva



Fonte: Registrada pelo autor

Pramerdorfer e Kampel (2015) disponibilizaram, em licença livre para uso acadêmico, diversas capturas feitas pelo procedimento da Figura 1. Elas foram base do estudo deste trabalho, visto que cumpre os requisitos explicados nesta seção e traz uma quantidade expressiva de imagens capturadas (cerca de 360).

### *3.2.6 Identificação de componentes*

Para este trabalho, limita-se a avaliação da identificação apenas de circuitos integrados, pelos seguintes motivos:

a) Resistores e indutores possuem baixa taxa de detecção nas literaturas estudadas, o que demanda, para não haver reprocessamento (*recall*), um algoritmo mais robusto e processadores mais potentes, o que não foi escopo do trabalho.
b) O formato desses componentes pode se confundir com os próprios CIs, o que também diminuiria a confiabilidade dos dois tipos de componentes.
c) Se atribuir a identificação de erros em circuitos impressos, os formatos de algumas falhas, como a falta ou excesso de cobre nos trilhos da PCB, podem ser confundidos com alguns dos itens que compõem a placa.

Assim, a fim de evitar o enviesamento de possíveis resultados, decide-se calcular a incidência de apenas um componente.

### *3.2.7 Testes e treinos*

Para o aprendizado de máquina nesse trabalho, toma-se como base o projeto desenvolvido por Ding et al[2] (2019). Este foi baseado em uma estrutura baseada no algoritmo Faster R-CNN (REN et. al, 2015) adaptado para a biblioteca TensorFlow[3].

O funcionamento dessa etapa depende da configuração correta dos drivers mostrados em Material, pois a demanda pela GPU é alta, devido ao processo convolucional da rede neural (CNN), que se utiliza de camadas para encontrar regiões com possíveis objetos.

Para o cálculo de precisão dos detectores, assim como o Faster R-CNN utilizado, é recorrente utilizar uma medida denominada de mAP (*mean Average Precision*, média Precisão Média, em uma tradução livre). O mAP relaciona o grau de precisão pelo índice de revocação (*recall*). Pode-se conceituar precisão como uma medida que verifica quanto é a precisão da predição. Assim, é definido matematicamente como

$$Precisão = TP/(TP + FP) \quad \text{Eq. 3}$$

[2] Código base: https://github.com/Ixiaohuihuihui/Tiny-Defect-Detection-for-PCB
[3] https://www.tensorflow.org/

Considerando $TP$ o positivo verdadeiro (*true positive*) e $FP$ o falso positivo (*false positive*). Ou seja, a precisão é a razão entre a detecção dos verdadeiros valores positivos sobre o total de positivos detectados.

A revocação, também chamado de sensibilidade (COVÕES, 2019), é dado por:

$$Revocação = TP/(TP + FN) \quad \text{Eq. 4}$$

Sendo $FN$ o falso negativo. Assim, o *recall* é a razão entre a detecção dos verdadeiros positivos sobre o total de positivos disponíveis.

Consolidando os dois conceitos, vale uma ilustração da relação entre si. Supondo uma amostra com 100 itens realmente positivos e outros itens, realmente negativos, com quantidade indiferente, conforme Tabela 1 e Tabela 2. Seleciona-se um item com predição correta, ou seja, um $TP$, sendo os demais itens ignorados, assim sendo $FN$. Verifica-se uma precisão de 1 (máxima), enquanto há uma revocação de 0,01 (baixa). À medida que a filtragem for menos seletiva, retendo mais informações de predição positiva, a depender do sistema de seletividade escolhida com a devida confiabilidade, os valores de revocação aumentam, enquanto a precisão tende a se fragilizar se a classificação não for robusta, conforme o exemplo de um algoritmo arbitrário da Tabela 2.

Tabela 1 – Matriz confusão com alta precisão

| | Predição positiva | Predição negativa |
|---|---|---|
| Real positiva | 1 | 99 |
| Real negativa | X | X |

Tabela 2 - Matriz confusão com alta revocação

| | Predição positiva | Predição negativa |
|---|---|---|
| Real positiva | 80 | 20 |
| Real negativa | X | X |

Na Discussão, presente na Seção 4, será mostrado o resultado da relação precisão e *recall*, ratificando a exemplificação com a robustez do código de forma gráfica.

### *3.2.8 Implementação de segmentação de objetos em imagens*

Os objetos que foram identificados em 3.2.6 e obtidos pelo treino configurado 3.2.7 foram gerados como identificadores únicos, sendo gravados no estilo

```
x, y, w, h, ocr (se tiver)
```

Sendo:

- x: posição horizontal do componente CI;
- y: posição vertical do CI;
- w: comprimento do CI;
- h: altura do CI;
- ocr: reconhecimento óptico do texto (OCR), com as gravações marcadas em alguns CIs.

A partir dos valores gravados, é possível verificar quantos PCBs possuem quantos CIs e, desses, qual a área ocupada por eles. Para saber como calcular a dimensão de comprimento, é necessário verificar a densidade de pixels. Para o *dataset* utilizado, são 222 pixels por polegada, portanto o fator de conversão, é cerca de 87,4 pixels por centímetro (1 pol = 2,54 cm).

# 4 RESULTADOS E DISCUSSÃO

Feito o processo definido pela Figura 5, alguns resultados foram encontrados e algumas contribuições puderam ser feitas a partir das análise dos gráficos e das Figuras a seguir.

Para o treino das imagens, foi criado um arquivo padrão para a biblioteca TensorFlow com extensão `.weights`, responsável por armazenar os padrões dos objetos agregados. Foram geradas cerca de 6200 amostras por meio do processo Faster R-CNN, que convoluciona diversas vezes a imagem, tornando a verificação robusta. Assim, foram gerados seis arquivos com a extensão mencionada, com 197 MB cada. Essa etapa durou cerca de 4,5 horas, com os equipamentos devidamente configurados conforme 3.1.

Após finalizado treino, foi executado o algoritmo dedicado ao registro e à gravação das identificações de cada circuito impresso, no formato disponível em 3.2.8. Para verificação se o sistema está de acordo com a veracidade, realizaram-se alguns testes por comparação ao controle humano, verificando se os números encontrados pela gravação correspondiam ao valor verdadeiro positivo. Alguns resultados podem ser vistos nas Figura 8 a Figura 12. Por fim, compilaram-se os resultados, trazendo as estatísticas nas Figura 13, Figura 14 e Figura 15.

Nas Figuras com a impressão dos PCBs, foi possível verificar que o algoritmo conseguiu verificar placas e CIs de dimensões variadas, desde formatos quadrados (os mais comuns) até formatos com cortes específicos, como na Figura 9. Os circuitos integrados foram destacados graficamente com um contorno vermelho.

Figura 8 - PCB com 3 CIs

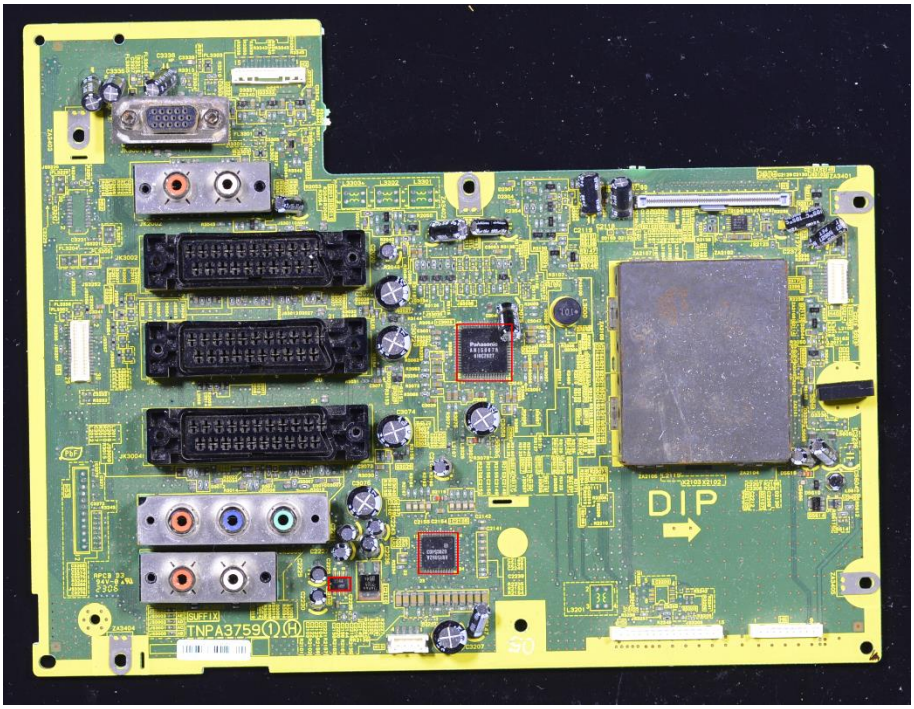



Figura 9 - PCB com 3 CIs

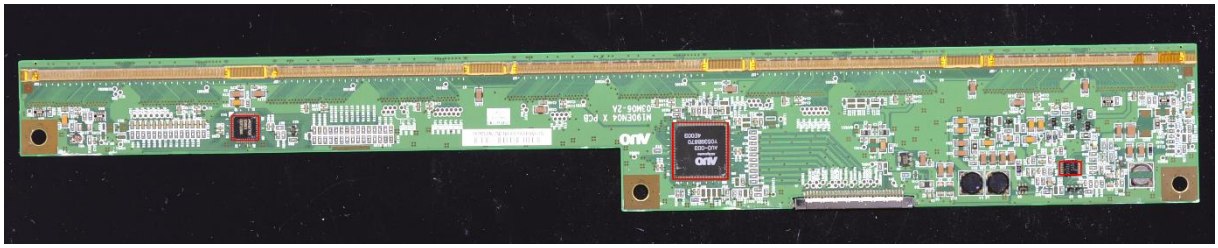


Figura 10 - PCB com 4 CIs

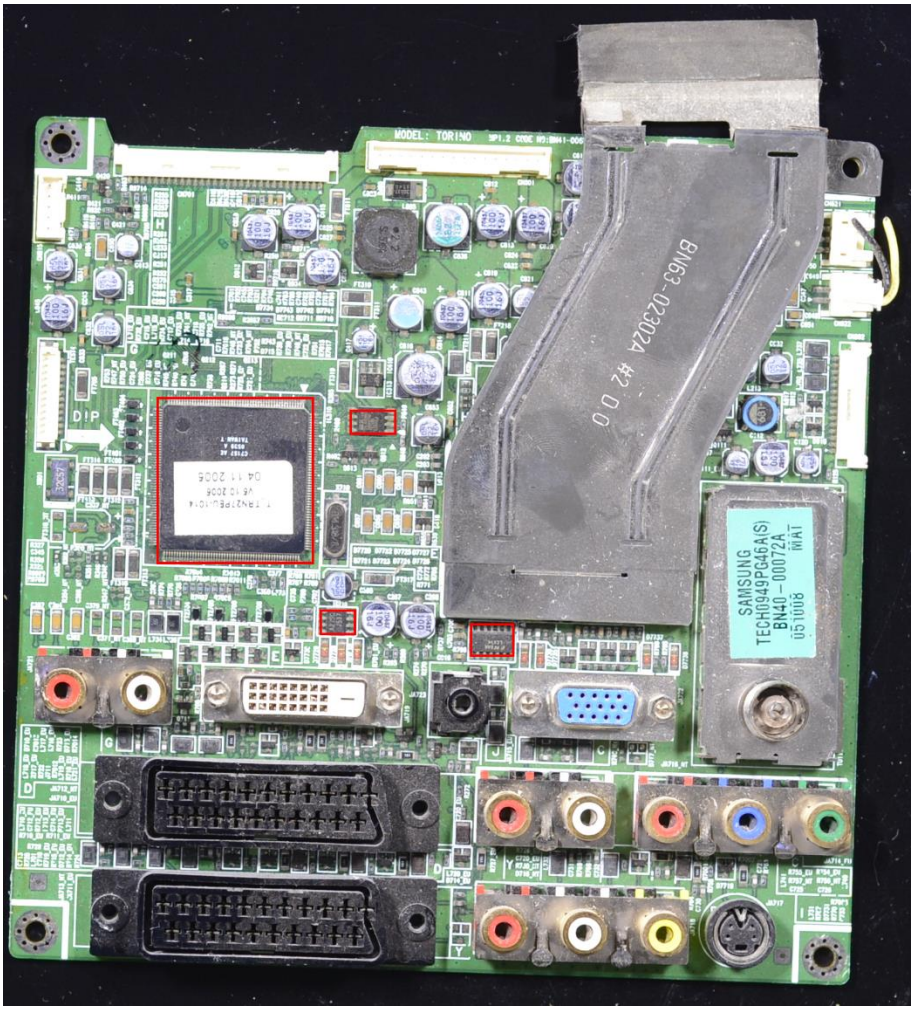


Figura 11 - PCB com 16 CIs

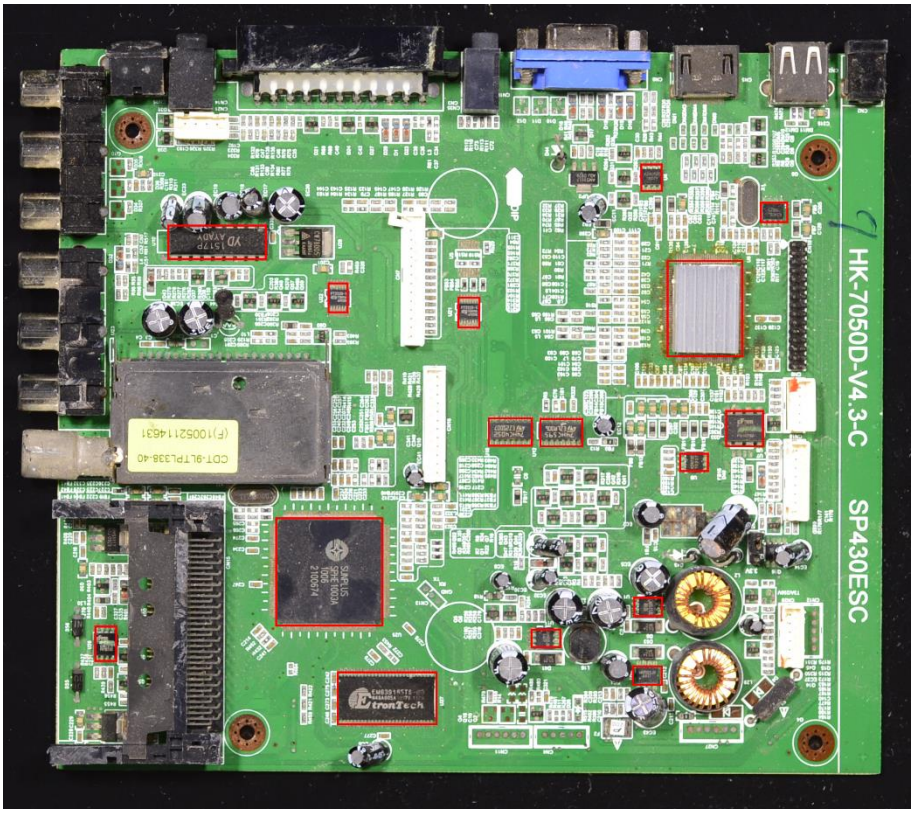
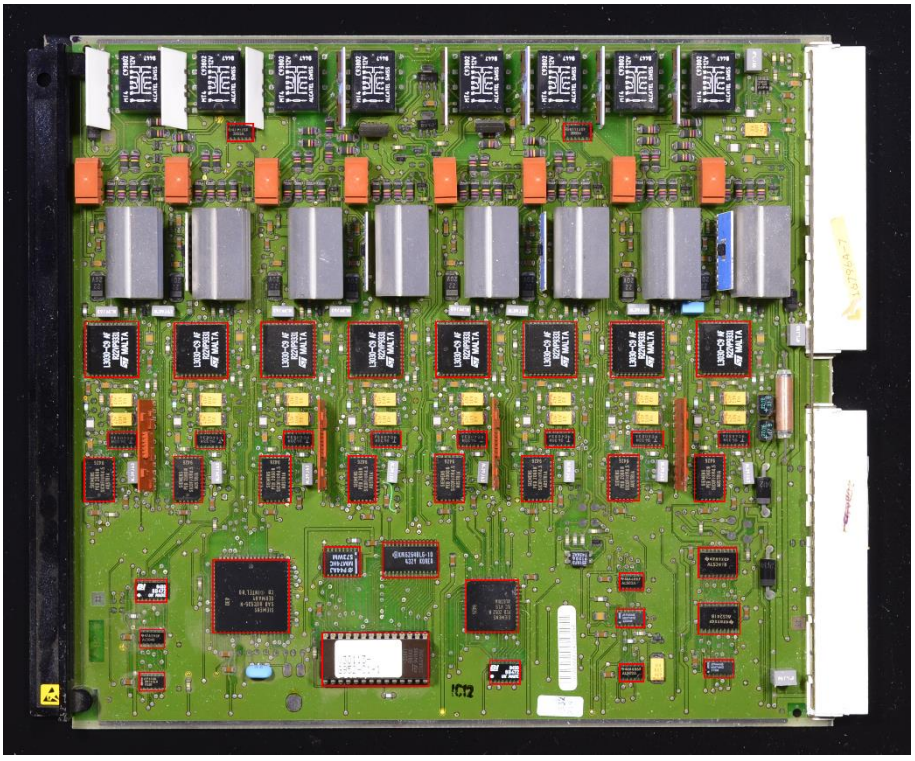


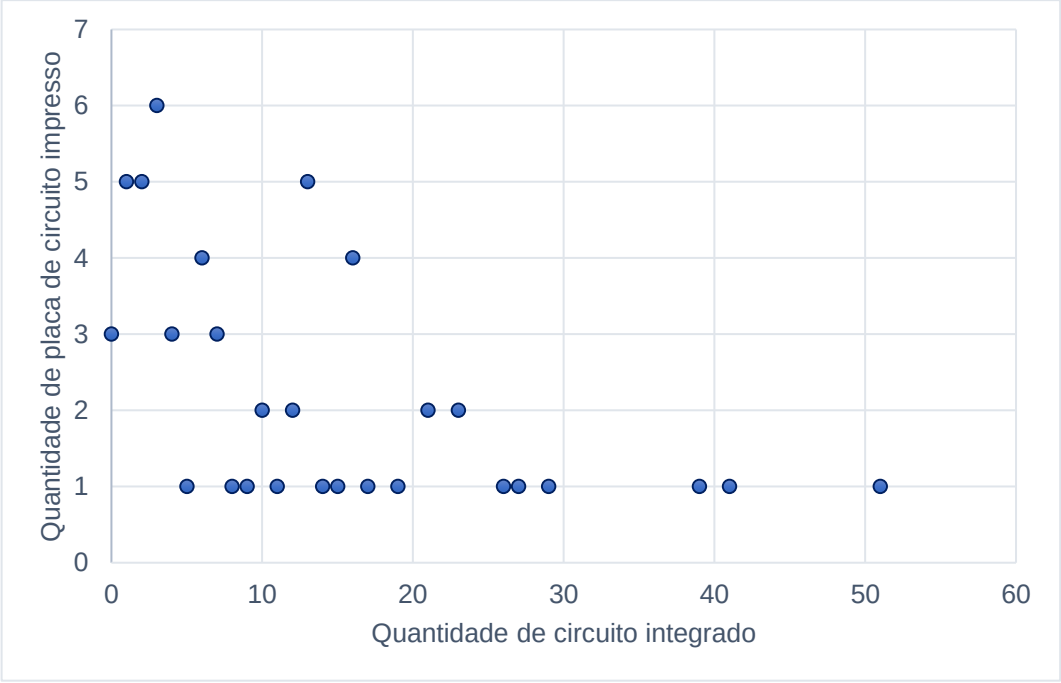

Figura 12 - PCB com 41 CIs



Feita a checagem à vista humana, verificaram-se as relações existentes entre quantidade e tamanho de CIs em diferentes PCBs. Na Figura 13, encontrou-se a concentração de unidades de placa de circuito impresso dada a presença de circuitos integrados em cada uma. No gráfico, apresentam-se as cem primeiras amostras do *dataset*.

Pôde-se verificar que, embora a amostra de imagens fosse feita com captura de PCBs de tamanhos aleatórios, há uma tendência de maior número de placas com presença menor de 20 CIs. Isso pode ser justificado pelo fato de haver menos necessidade, em um contexto eletrônico geral, que se possua diversos CIs, dada a menor complexidade do circuito e do desempenho esperado pela placa. Por outro lado, para usos específicos de alguma atividade, circuitos com maior complexidade são necessários, portanto, presenciaram-se ao menos dez placas com mais de 20 circuitos integrados agregados.

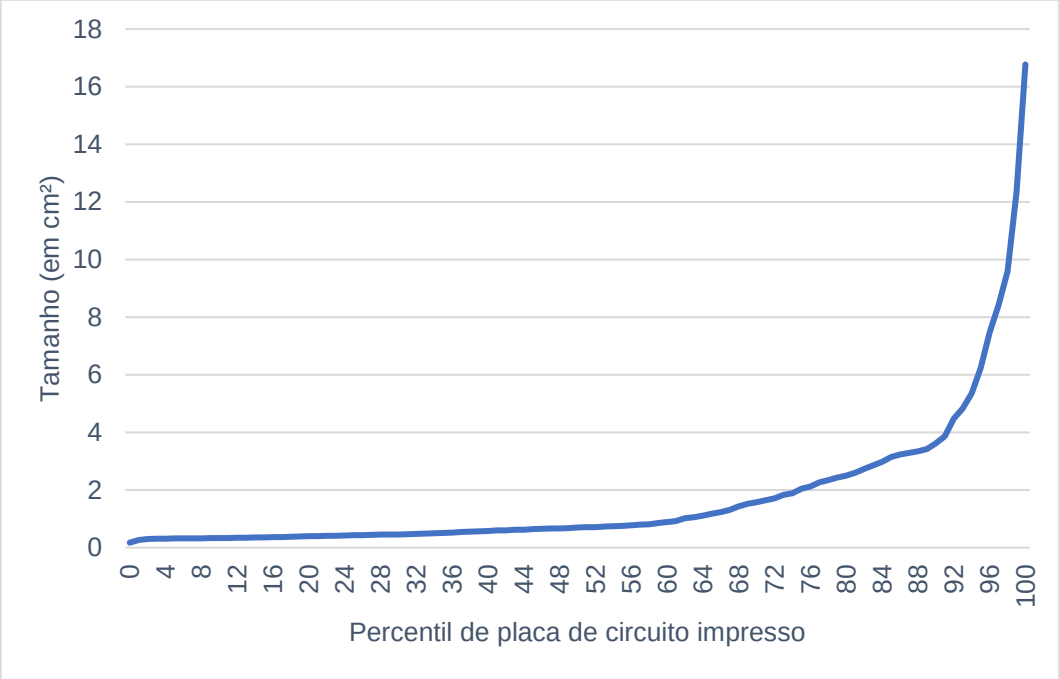
Figura 13 - Relação quantitativa de concentração de CIs em PCBs




A Figura 14 apresenta que, no total de amostras disponibilizadas, cerca de 92% possuíam todos os CIs com menos de 4 $cm^2$ e a partir desse percentual, os tamanhos se elevavam exponencialmente, chegando em componentes de cerca de 17 $cm^2$. Os números convergem com os resultados apresentados pela Figura 13, uma vez que, para uso geral, sem grande demanda de processamento, os CIs podem ser de

tamanho miniaturizado. Novamente, em casos específicos, há necessidade de uma montagem com um circuito integrado de área mais elevada.

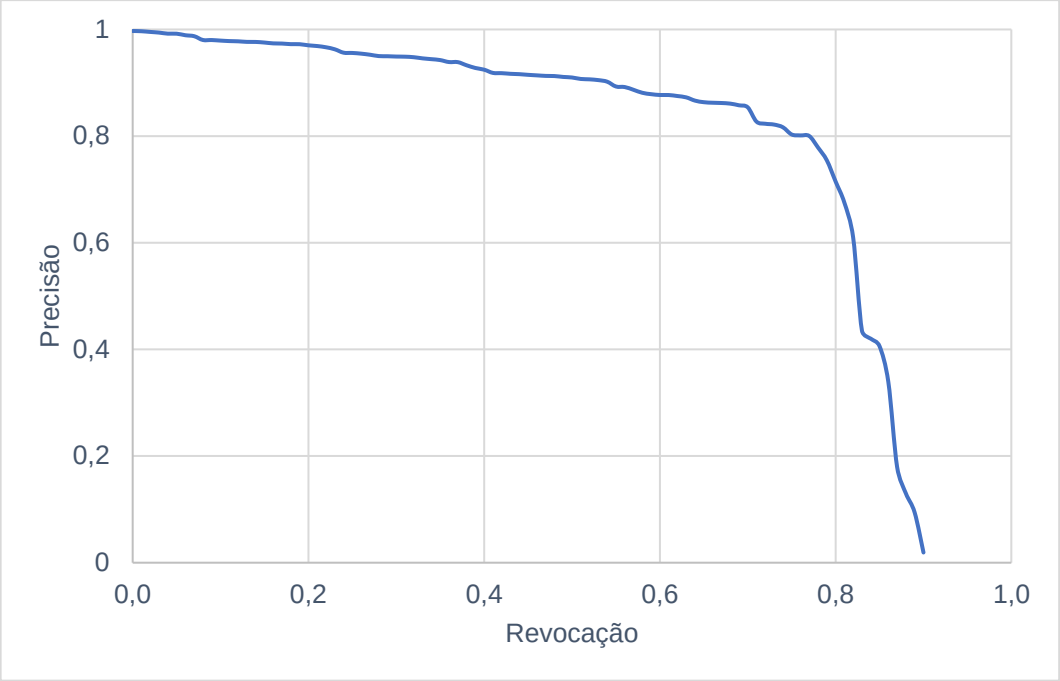
Figura 14 - Relação do tamanho dos CIs pelo percentil da quantidade de PCBs




Finalmente, a Figura 15 inclui a medição da relação de precisão e *recall*. O algoritmo se manteve estável, com uma precisão adequada de até 0,8, até cerca de um *recall* de 0,7, quando se percebeu que a análise deixou de ficar confiável a partir desse valor. Revisitando a literatura, em específico, em 2.2.4, pôde-se ver alguns comportamentos semelhantes ao encontrado e outros com uma acurácia maior, com queda da curva no *recall* mais próximo de 1,0. Isso, provavelmente, ocorreu devido a alguma configuração de captura de imagem, na qual o algoritmo interpretou indevidamente e considerou como falso positivo ou falso negativo.

Entretanto, o resultado encontrado demonstra solidez na contabilização, com a possibilidade de melhoria da execução com o controle mais refinado em alguma das etapas do bloco “Dataset”, da Figura 5, para aprimorar os resultados em maiores *recalls*.

Figura 15 - Performance do Faster R-CNN



# 5 CONCLUSÃO

Conclui-se com êxito a execução deste trabalho, com grande aquisição de conhecimento do funcionamento do aprendizado profundo de máquina, voltado para a visão computacional. Conseguiu-se identificar, sem o contato direto humano, a quantidade de circuitos integrados presentes em PCBs e quais suas áreas.

Houve uma extensa pesquisa exploratória para verificar a complexidade das redes neurais convolucionais. Os aprimoramentos que forem feitos em cada algoritmo nessa área são de majestosa importância, visto a ascensão da Indústria 4.0. E a comunidade relacionada a essa área vem aumentando, disponibilizando cada vez mais conteúdos para gerar novos *insights*.

Há possibilidades de melhora para alcançar o estado da arte do algoritmo proposto no trabalho, como a implementação de mais componentes, melhora na eficiência mAP para maiores *recalls*. Isso pode ser alcançado com a importação de mais amostras de componentes e placas de circuito impresso, que aumentam a capacidade de entendimento do computador.

## REFERÊNCIAS

ASIF, A. M. A. M. et al. An Overview and Applications of Optical Character Recognition. **International Journal of Advance Research In Science And Engineering**, v. 3, p. 261-274, jun. 2014.

AWS. Formula 1 Case Study. **AWS**, 2018. Disponivel em: <https://aws.amazon.com/pt/solutions/case-studies/formula-one/>. Acesso em: 31 out. 2019.

BENEDEK, C. Detection of Soldering Defects in Printed Circuit Boards with Hierarchical. **Pattern Recognition Letters**, Budapeste, v. 32, n. 13, p. 1535-1543, 2011.

BRASIL. Lei nº 12.305, de 2 de agosto de 2010. **Institui a política nacional de resíduos sólidos**, Brasília, DF, 2 ago. 2010. Disponivel em: <http://www2.mma.gov.br/port/conama/legiabre.cfm?codlegi=636>. Acesso em: 1 set. 2019.

COVÕES, T. F. **Aprendizado de Máquina 2019.1: Avaliação de Classificadores**. Universidade Federal do ABC. Santo André, p. 26 p. 2019.

DALAL, N.; TRIGGS, B. Histograms of Oriented Gradients for Human Detection. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)**, v. 2, p. 886-893, jun. 2005.

DING, R. et al. TDD-net: a tiny defect detection network for printed circuit boards. **CAAI Transactions on Intelligence Technology**, v. 4, n. 2, p. 110-116, abr. 2019.

GIRSHICK, R. **Fast R-CNN**. Microsoft Research. [S.l.]. 2015.

GIRSHICK, R. et al. **Rich feature hierarchies for accurate object detection and semantic segmentation**. UC Berkeley. Califórnia. 2013.

HOOF, L. V. Basic principles of PCB assembly: Through-Hole vs Surface Mounted. **EPR**, 2019. Disponivel em: <https://eprpartner.com/through-hole-vs-surface-mounted/>. Acesso em: 31 ago. 2019.

HUANG, W.; WEI, P. A PCB Dataset for Defects Detection and Classification, 2019.

KRIZHEVSKYAND, A.; SUTSKEVER, I.; HINTON, G. ImageNet Classification with Deep Convolutional Neural Networks. **Neural Information Processing Systems**, v. 25, jan. 2012.

LETA, F. R. et al. **Discussing Accuracy in an Automatic Measurement System using Computer Vision Techniques**. 18th International Congress of Mechanical Engineering. Ouro Preto: COBEM 2005. 2005.

LETA, F. R.; FELICIANO, F. F.; MARTINS, F. P. R. **Computer vision system for printed circuit board inspection**. ABCM Symposium Series in Mechatronics. [S.l.]: ABCM. 2008. p. 623-632.

LI, J. et al. Application Research of Improved YOLO V3 Algorithm in PCB Electronic Component Detection. **Applied Sciences**, v. 9, set. 2019.

LI, W.; ESDERS, B.; BREIER, M. **SMD segmentation for automated PCB recycling**. 2013 11th IEEE International Conference on Industrial Informatics. Bochum: [s.n.]. jul. 2013. p. 65-70.

LIN, T.-Y. et al. **Feature Pyramid Networks for Object Detection**. [S.l.]. 2016.

LIU, W. et al. SSD: Single Shot MultiBox Detector. **Lecture Notes in Computer Science**, p. 21–37, 2016.

MAHALINGAM, G.; GAY, K. M.; JR., K. R. **PCB-METAL:** A PCB Image Dataset for Advanced Computer Vision Machine Learning Component Analysis. 2019 16th International Conference on Machine Vision Applications (MVA). [S.l.]: [s.n.]. 2019. p. 1-5.

NORVIG, P.; RUSSELL, S. **Artificial Intelligence:** A Modern Approach. 1. ed. Nova Jersey: Prentice Hall, 1995. 934 p.

PARKER, J. R. **Algorithms for image processing and computer vision**. 2. ed. Indianápolis: Wiley Publishing, 2011.

PEDERSOLI, M.; VEDALDI, A.; GONZÀLEZ, J. A coarse-to-fine approach for fast deformable object detection. **Pattern Recognition**, v. 48, jul. 2011.

PERASSO, V. O que é a 4ª revolução industrial - e como ela deve afetar nossas vidas. **G1**, 22 out. 2016. Disponivel em: <http://glo.bo/2f2lrQO>. Acesso em: 30 out. 2019.

PRAMERDORFER, C.; KAMPEL, M. **A Dataset for Computer-Vision-Based PCB Analysis**. MVA2015 IAPR International Conference on Machine Vision Applications. Tóquio: IAPR. 2015. p. 378-381.

QUEIROZ, J. E. R. D.; GOMES, H. M. Introdução ao Processamento Digital de Imagens. **RITA**, v. 13, p. 11-42, jan. 2006.

REDMON, J. et al. **You Only Look Once:** Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Long Beach, CA: [s.n.]. 2015. p. 779-788.

REN, S. et al. **Faster R-CNN:** Towards Real-Time Object Detection with Region Proposal Networks. [S.l.]: [s.n.]. 2015.

TAHA, E. M.; EMARY, E.; MOUSTAFA, K. Automatic Optical Inspection for PCB Manufacturing: a Survey. **International Journal of Scientific & Engineering Research**, v. 5, n. 7, p. 1095-1102, jul. 2014.

VIOLA, P.; JONES, M. Rapid Object Detection using a Boosted Cascade of Simple Features. **IEEE Conf Comput Vis Pattern Recognit**, v. 1, p. I-511, fev. 2001.

WANG, M.; DENG, W. **Deep Face Recognition: A Survey**. Beijing University of Posts and Telecommunications. Pequim. 2018.

WEI, Y. et al. Multi-vehicle detection algorithm through combining Harr and HOG features. **Mathematics and Computers in Simulation**, v. 155, p. 130-145, fev. 2018.