UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

TRABALHO DE CONCLUSÃO DE CURSO EM ENGENHARIA QUÍMICA

Inspeção Automática da Qualidade de Codificação em Garrafas PET utilizando Visão Computacional em Sistema Embarcado

Autor: Gabriel Araujo de Souza

Orientador: Prof. Dr. Pedro Rafael Bolognese Fernandes

Porto Alegre, novembro de 2021

Autor: Gabriel Araujo de Souza

Inspeção Automática da Qualidade de Codificação em Garrafas PET utilizando Visão Computacional em Sistema Embarcado

Trabalho de Conclusão de Curso apresentado à COMGRAD/ENQ da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Bacharel em Engenharia Química

Orientador: Prof. Dr. Pedro Rafael Bolognese Fernandes

Banca Examinadora:

Prof. Dr. Jorge Otávio Trierweiler, UFRGS.

M. Sc. Vinícius da Costa Ávila.

Porto Alegre

2021

AGRADECIMENTOS

Agradeço aos meus pais e a toda minha família, por todo suporte e apoio nesta caminhada; aos meus amigos, pelos aprendizados e pela parceria nos desafios que enfrentamos ao longo da graduação, e a todos os professores desta universidade, pelos ensinamentos e pela paciência em transmitir seu conhecimento. Agradeço, especialmente, ao professor Pedro Fernandes pela orientação concedida durante o desenvolvimento deste trabalho, que foi de suma importância.

RESUMO

A qualidade da embalagem é um fator determinante na escolha de determinado produto por um consumidor. Além da aparência adequada, os requisitos legais exigem que certas informações estejam disponíveis para o consumidor. Este trabalho tem como objetivo principal avaliar a viabilidade técnica de utilizar uma placa Raspberry Pi e câmera de baixo custo Raspberry Pi Camera num sistema inspetor automático de qualidade de codificação em garrafas PET. Este sistema consiste na captura da imagem digital, tratamento da imagem obtida e posterior realização de reconhecimento de texto visando encontrar o conjunto de caracteres definidos para cada modelo de garrafa. Caso todos os caracteres necessários sejam encontrados, a garrafa é classificada como conforme. Foram utilizados dois modelos de garrafas, e capturadas 24 imagens de cada, sendo 15 destas de exemplos de garrafa conforme e 9 com alguma não conformidade. Os testes foram divididos em duas sequências de métodos de processamento. A segunda sequência obteve 91,7% de acurácia para as garrafas do modelo A e 83,3% para o modelo B. As classificações incorretas resultam principalmente de fatores associados à iluminação e atrasos na captura das imagens. Para melhorar os resultados sugere-se realizar os testes em ambiente com iluminação mais controlada e utilizar uma placa microprocessada com maior capacidade.

Palavras-chave: *Visão Computacional*, *Sistema Embarcado*, *Processamento de Imagens*, *Inspeção Automática, Qualidade de Codificação.*

ABSTRACT

Quality of packaging is a key factor in a consumer's choice for a particular product. Besides the package appearance, legal requirements require that certain information is presented to the consumer. The main goal of this work is to evaluate the technical feasibility of using a Raspberry Pi board as an automatic quality inspector for the expiration date and batch region in PET bottles. The system consists in capturing digital images using a Raspberry Pi Camera, processing the image obtained, and performing Optical Character Recognition in order to find the set of characters defined for each model of bottle. If all necessary characters are found, the bottle is classified as compliant. Two models of bottles were used, and 24 images of each were captured, 15 of which were examples of compliant bottles and 9 were not. The tests were divided into two sequences of processing methods. The second sequence achieved 91.7% accuracy for model A bottles and 83.3% for model B. Misclassifications resulted mainly from factors associated with lighting conditions and delays in capturing images. To improve results, it is suggested to carry out the tests in an environment with better lighting conditions and to use a microprocessed board with greater processing capacity.

Keywords: *Computer Vision*, *Raspberry Pi*, *Image processing*, *Automatic Inspection.*

LISTA DE FIGURAS

SUMÁRIO

1 Introdução

A qualidade da embalagem é um fator determinante na escolha de determinado produto por um consumidor. Além da aparência adequada, os requisitos legais exigem que certas informações estejam disponíveis para o consumidor. Quando tratamos de produtos envasados em garrafas PET, aspectos como qualidade de rotulagem, de encapsulamento, de codificação e boa formação da garrafa são parte do controle de qualidade de muitas indústrias.

A codificação das garrafas PET é a região onde constam as informações de identificação do lote e do prazo de validade para produtos alimentícios. A apresentação destas informações é obrigatória de acordo com a Resolução-RDC N° 259 da ANVISA, que aprova o Regulamento Técnico sobre Rotulagem de Alimentos Embalados. Esta resolução define as formatações permitidas e salienta que todas as informações devem ser apresentadas de forma clara e legível em todos os produtos. O não cumprimento de quaisquer especificações da Resolução fica sujeito às sanções da Lei Federal N° 6437, que dispõe sobre infrações à legislação sanitária federal. O principal objetivo destas regulamentações é proteger a integridade física do consumidor, exigindo que todas as informações pertinentes ao consumo estejam dispostas na embalagem.

Para as empresas, a codificação é fundamental no processo de rastreabilidade dos produtos. Por meio do número do lote, é possível rastrear toda cadeia produtiva daquele item, e assim tomar as providências necessárias caso ocorra algum problema que exija a recolha do lote, ou efetuar investigações de anomalias constatadas por meio do Serviço de Atendimento ao Cliente (SAC). Portanto, a verificação periódica da qualidade de codificação dos produtos é uma medida adotada como parte da rotina da operação da fábrica.

Em indústrias que não possuem sistema automático, é definido um intervalo de tempo aceitável para que seja realizada a inspeção visual da codificação, visando garantir que a mesma não apresente caracteres borrados ou mesmo ausentes e com a data de validade correta. Caso seja detectada alguma anomalia, a produção é retida a partir do momento da última verificação que estava em conformidade até o momento em que a anomalia é sanada. Esta produção deverá ser retrabalhada ou descartada. Caso a falha tenha sido ausência de codificação, será necessário recolocar as garrafas na linha de produção para que passem novamente pela máquina codificadora. Se a falha for codificação borrada, ilegível, ou com a data de validade incorreta, será necessário apagar manualmente a codificação com defeito, para então passar novamente as garrafas pela codificadora da linha. A empresa deverá definir se efetuará este retrabalho ou descartará os produtos não-conformes, visto que para retrabalhar é necessário mão de obra, espaço físico, e tempo.

Atualmente, existem métodos baseados em visão computacional para determinar se determinado aspecto de um produto apresenta conformidade ou não. Essas soluções transformam uma análise qualitativa em quantitativa, apresentam imparcialidade na análise e podem ser realizadas em linhas de produção com alta velocidade, em intervalos determinados pela necessidade das empresas ou pela capacidade dos equipamentos envolvidos.

Inspeções de qualidade de tampa e rótulo, bem como nível de enchimento do produto, são exemplos de aplicações de sistemas de visão computacional consolidados no ramo. A inspeção da qualidade de codificação em garrafas PET apresenta algumas dificuldades associadas. Entre estas dificuldades, pode-se mencionar: a curvatura cilíndrica da garrafa, que dificulta o reconhecimento de caracteres; a natureza do material PET, que pode refletir a luz e causar ofuscamento de alguns caracteres; a posição da codificação, que por vezes se encontra abaixo do nível do líquido, diminuindo o contraste entre as letras e o líquido, dependendo da cor deste; e a formação de espuma, bolhas ou gotas nas paredes da garrafa, que pode comprometer a leitura dos caracteres.

1.1 Objetivos do Trabalho

1.1.1 Objetivo Geral

Tendo em vista a importância do controle da qualidade de codificação na indústria e buscando melhorar a confiabilidade desta verificação, e assim diminuir os custos envolvidos em caso de falhas, este trabalho visa investigar a viabilidade técnica de um sistema automático de inspeção da qualidade de codificação para garrafas PET de bebidas empregando placa micro processada e câmera de baixo custo.

1.1.2 Objetivos Específicos

Para atingir o objetivo geral, são propostos os seguintes objetivos específicos:

- Verificar se os dispositivos utilizados são capazes de fornecer a acurácia necessária para o processo;
- Definir uma sequência de métodos de processamento de imagens digitais adequados para melhorar a eficiência do sistema de reconhecimento de texto (OCR) empregado;
- Definir as ferramentas de OCR mais adequadas para detecção e reconhecimento do texto;
- Definir as regras para classificação da codificação como “Conforme” ou “Não-conforme”;
- Propor um método de processamento em tempo real do sistema.

2 Revisão Bibliográfica

A seguir serão apresentadas tanto informações relativas ao tema deste trabalho como referencial teórico para os métodos aplicados, e também pesquisas envolvendo o tema.

2.1 Conceitos sobre a captura de imagens digitais

2.1.1 Fundamentos de imagens digitais

Uma imagem pode ser definida como uma função contínua de intensidade luminosa, denotada f(x,y), cujo valor ou amplitude nas coordenadas (x,y) fornece a intensidade ou o brilho da imagem naquele ponto. Uma vez que a maioria das técnicas de análise de imagem é realizada por meio de processamento computacional, a função f(x,y) precisa ser convertida para a forma discreta. Para obter uma imagem digital, são necessários dois passos, a amostragem e a quantização (Pedrini e Schwartz, 2007).

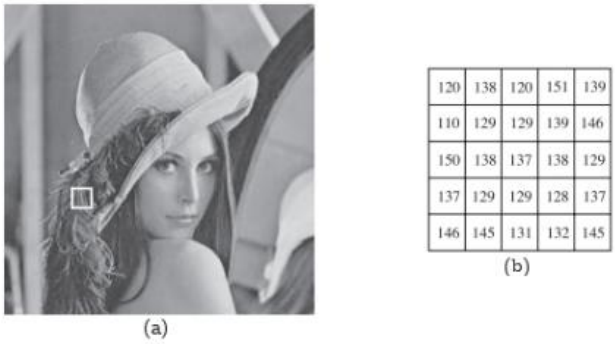
A amostragem é responsável por discretizar o domínio de definição da imagem nas direções x e y, gerando uma matriz de amostras de tamanho *M* x *N*, na qual cada elemento desta matriz é chamado *pixel*. A matriz gerada pode ser expressa como na Equação 2.1:

$$f(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(M-1,0) \\ \vdots & \ddots & \vdots \\ f(0,\ N-1) & \cdots & f(M-1,\ N-1) \end{bmatrix} \tag{2.1}$$

A quantização é responsável por determinar o número inteiro L de níveis de cinza (em uma imagem monocromática) permitido para cada ponto da imagem. O intervalo [Lmin, Lmax] é denominado escala de cinza. Por convenção, é atribuída a cor preta ao nível de cinza mais escuro, e a cor branca ao nível de cinza mais claro. É usual em processamento digital de imagens assumir que as dimensões da imagem e o número de níveis de cinza sejam potências inteiras de 2. No caso em que o número de níveis de cinza é igual a 2, a imagem é chamada binária. Estas imagens ocupam menos espaço de armazenamento e são facilmente manipuladas pelos computadores, sem requerer grande processamento (Pedrini e Schwartz, 2007).

Determinar o número de amostras M x N e de níveis de cinza L necessários para gerar uma boa imagem digital depende da quantidade de informação contida na imagem e do grau de detalhes dessa informação que é perceptível ao olho humano. Um conceito importante neste tema é a resolução espacial, que está associada à densidade de pixels na imagem. A resolução de uma imagem deve ser suficiente para atender ao grau de detalhes que devem ser discerníveis na imagem. Outro conceito importante é o de profundidade da imagem, que é definida como o número de bits necessários para armazenar a imagem digitalizada. Uma imagem com profundidade de 8 significa que ela é capaz de armazenar níveis de cinza de 0 a 255 (Pedrini e Schwartz, 2007). A Figura 1 exemplifica o conceito de representação matricial de uma imagem digital com seus níveis de cinza por pixel.

Figura 1: Representação matricial. (a) imagem original; (b) níveis de cinza correspondentes à região da imagem em destaque.

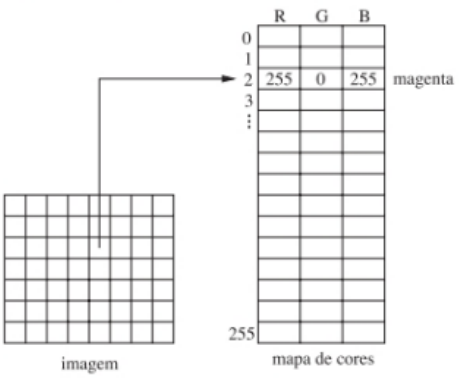


Fonte: Adaptado de Pedrini e Schwartz (2007).

2.1.2 Imagens multibanda ou multiespectrais

Uma imagem multiespectral pode ser representada como uma sequência de n imagens monocromáticas, tal que cada imagem é conhecida como banda. A maioria das cores visíveis pelo olho humano pode ser representada como uma combinação de bandas das cores primárias vermelha (R, *red*), verde (G, *green*), e azul (B, *blue*), que são utilizadas comumente para representar imagens coloridas (Pedrini e Schwartz, 2007). A Figura 2**Erro! Fonte de referência não encontrada.** exemplifica a relação entre cada pixel de uma imagem digital e suas bandas de cores RGB.

Figura 2: mapa de cores para representar uma imagem colorida.



Fonte: Adaptado de Pedrini e Schwartz (2007).

2.2 Conceitos sobre métodos de tratamento de imagens digitais

2.2.1 Conceito de máscara, ou kernel

Em processamento de imagens, existe a necessidade de realizar operações de filtragem para extrair informações de interesse na imagem. Essas operações podem ser realizadas tanto no domínio do espaço quanto no de frequência. A filtragem no domínio espacial, isto é, aquele realizado diretamente sobre o conjunto de pixels que compõe a imagem, geralmente é realizada por meio de matrizes denominadas “máscaras”. A máscara mais simples tem tamanho 3x3, e a cada posição da máscara está associado um valor numérico,

chamado de peso ou coeficiente, conforme Figura 3. A aplicação da máscara com centro na coordenada (x, y) consiste na substituição do valor do pixel na posição (x, y) por um novo valor, que depende dos valores dos pixels vizinhos e dos pesos da máscara. Os coeficientes do filtro são multiplicados pelos níveis de cinza dos pixels correspondentes e então somados, substituindo o nível de cinza do pixel central (Pedrini e Schwartz, 2007), conforme Equação 2.2.

Figura 3: máscara de 3x3 pixels com coeficientes arbitrários.

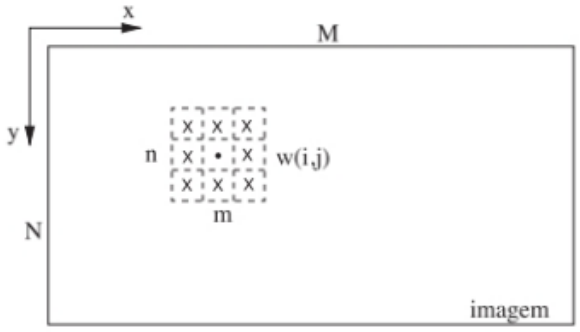
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Fonte: Adaptado de Pedrini e Schwartz (2007).

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \tag{2.2}$$

A máscara é movida para a próxima posição de pixel na imagem, e o processo é repetido até que a máscara passe por todas as posições da imagem, conforme exemplificado na Figura 4. A aplicação deste filtro em cada pixel da imagem é chamada de *correlação*. Formalmente, a correlação é definida na Equação 2.3. A *convolução* consiste em um processo similar, porém o filtro deve sofrer uma reflexão, ou rotação de 180 graus antes de ser aplicado à imagem (Pedrini e Schwartz, 2007).

Figura 4: filtragem no domínio espacial.



Fonte: Adaptado de Pedrini e Schwartz (2007).

$$w.f(x,y) = \sum_{i=[-\frac{m}{2}]}^{[\frac{m}{2}]} \sum_{j=[-\frac{n}{2}]}^{[\frac{n}{2}]} w(i,j)\, f(x+i, y+j) \tag{2.3}$$

2.3 Conceitos sobre identificação e reconhecimento de texto em imagens

2.3.1 Identificação de texto em imagens

A identificação de texto em imagens tipicamente envolve duas etapas. A primeira delas é detectar a região onde está contido o texto, obtendo as coordenadas da imagem que contém o texto, comumente chamadas de *bounding box*. A segunda etapa consiste em reconhecer os caracteres desta região, e para isso podem ser utilizados algoritmos de

Reconhecimento Óptico de Caracteres (OCR). Para ambas as etapas é comum a utilização de algoritmos baseados em redes neuronais, treinados para cada um dos objetivos.

2.4 Processos de aquisição de imagens

A seguir, serão apresentados conceitos relativos ao processo de aquisição de imagens digitais como o funcionamento de sensores, o processo de digitalização e a influência do obturador utilizado na imagem obtida.

2.4.1 Sensores e Sistema de aquisição de imagens

O processo de aquisição de imagens digitais através de sensores consiste basicamente em duas etapas principais, transformação da energia luminosa que chega ao sensor em diferença de tensão elétrica, e transformação da resposta ondulatória da tensão em informação digital. O material do sensor deve ser responsivo ao tipo de energia que se deseja detectar. Por exemplo, um fotodiodo é constituído por materiais de sílica, que produzem uma tensão proporcional à quantidade de luz que chega ao sensor (Gonzalez e Woods, 2008).

Câmeras digitais tipicamente utilizam arranjos de sensores em formato de uma matriz 2D para aquisição das imagens. A energia proveniente de uma fonte luminosa é refletida em objetos da cena, e em seguida passa pela primeira parte do sistema de aquisição de imagem, que consiste em uma lente óptica que projeta a cena no plano focal da lente. O arranjo de sensores, situado no plano focal, produz uma tensão elétrica proporcional à integral da energia recebida em cada sensor. Circuitos digitais e analógicos convertem essa tensão em um sinal analógico, que então é digitalizado por outra parte do sistema de aquisição de imagem. O resultado final é a aquisição de uma imagem digital (Gonzalez e Woods, 2008). Na seção do Apêndice A, a Figura A.1 mostra o esquema de funcionamento de um arranjo de sensores, e a Figura A.2 exemplifica o processo de aquisição de imagens digitais.

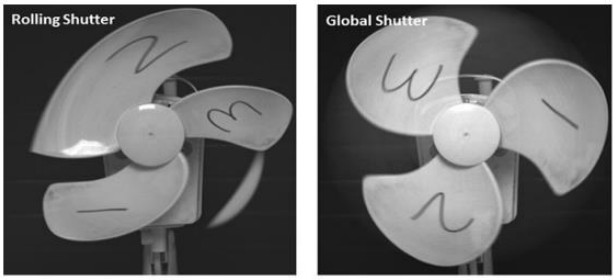
2.4.2 Rolling Shutter e Global Shutter

Para a aquisição de uma imagem digital é necessário definir o tempo de exposição à luz para capturar uma imagem. Este tempo de exposição é determinado pela velocidade do obturador, ou *shutter speed*, e pode ser definido como a quantidade de tempo em que o obturador permanece aberto durante o disparo de uma fotografia. Assim, a utilização de velocidades mais elevadas do obturador resulta em menores tempos de exposição e vice-versa (Coates e Juvan-Beaulieu, 2020).

O método de captura *global shutter* consiste em permitir a exposição à luz, armazenando a energia de cada sensor no seu respectivo acumulador, e, após o tempo determinado, interromper o acúmulo de energia em todos os sensores simultaneamente. Este processo pode ser encontrado em sensores do tipo CCD (*Charge Coupled Device,* ou Dispositivo de Carga Acoplada*)*, e em alguns sensores do tipo CMOS (*Complementary Metal-Oxide-Semiconductor*, ou semicondutor metal-óxido complementar). No método *rolling shutter*, o processo de interromper o acúmulo de energia nos sensores não é feito simultaneamente, e sim, linha a linha do arranjo. Esta forma de leitura das cargas acumuladas dos sensores pode causar o chamado efeito *rolling shutter*, caso o objeto capturado esteja em movimento e a velocidade do obturador não seja pequena o suficiente para assegurar o congelamento da

imagem (Coates e Juvan-Beaulieu, 2020). A Figura 5 apresenta um exemplo do efeito *rolling shutter*.

Figura 5: exemplo da aquisição de duas imagens digitais utilizando os métodos *rolling shutter* e *global shutter,* respectivamente, para o mesmo tempo de exposição.

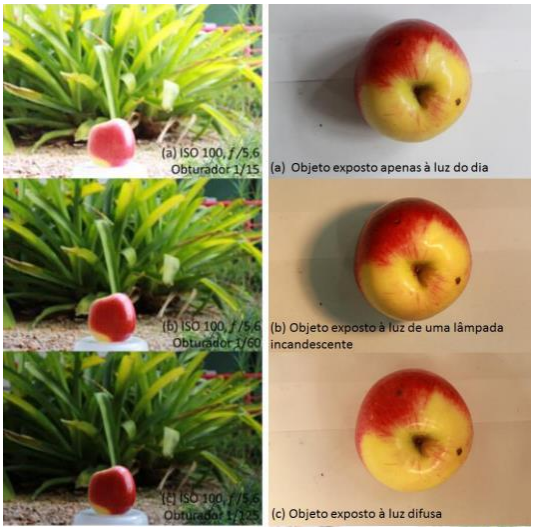


Fonte: Adaptado de Coates e Juvan-Beaulieu (2020).

A pesquisa desenvolvida por Silva (2020) mostra a influência dos parâmetros de ajuste de uma câmera semiprofissional nas imagens adquiridas. Entre os parâmetros testados, foi avaliada a influência da velocidade do obturador na captura de imagens estáticas, e observou-se que o aumento na velocidade do obturador causa um escurecimento nas imagens adquiridas, à medida que o tempo para captação de fótons é menor. Além disso, velocidades menores contribuem para a estabilização da imagem, auxiliando na minimização de tremidos e borrões.

Outro cenário testado na pesquisa envolve as diferenças na natureza da luz incidente. Observou-se que a distância entre a fonte de iluminação e o objeto altera o tipo de sombra gerada. Quanto mais perto a fonte estiver do objeto, mais bem definida é a sombra. Por fim, observou-se que, ao utilizar uma folha de papel alumínio como anteparo, é possível direcionar fótons às regiões do objeto que não recebem iluminação direta, gerando o efeito de iluminação difusa, que é responsável por causar uma diminuição no tamanho da sombra obtida, ou até mesmo não gerar sombra alguma. A Figura 6 exemplifica os efeitos de variação de *shutter speed* e variação nas formas de incidência de luz no objeto.

Figura 6: Efeito das variações testadas. À esquerda mostra-se o efeito de escurecimento das fotos conforme diminuição do *shutter speed.* À direita mostra-se o efeito das formas de incidência de luz na sombra obtida na imagem.



Fonte: Adaptado de Silva (2020).

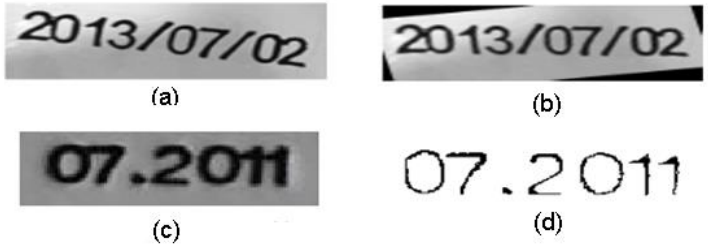
2.5 Aplicações em Reconhecimento de Codificação

Esta seção é destinada a apresentar alguns estudos que estão relacionados com o objetivo deste trabalho, seja através da utilização de ferramentas técnicas semelhantes, seja com o uso de Raspberry Pi e algoritmos de tratamento de imagem e OCR. A partir desta revisão, foram selecionados alguns dos métodos utilizados no desenvolvimento do trabalho, bem como definidos os principais parâmetros de ajuste para os tratamentos de imagem utilizados.

2.5.1 Leitura de caracteres de codificação

Zaafouri, Sayadi e Fnaiech (2015) propõe as etapas para a construção de um sistema de visão para reconhecimento dos caracteres de data de validade dos produtos industriais. O modelo foi testado em produtos variados, com diversos formatos, com foco exclusivo na identificação das datas de validade. São propostos quatro estágios de processamento: pré-processamento da imagem, segmentação dos caracteres, extração das características e reconhecimento dos caracteres. A etapa de pré-processamento consiste em converter a imagem do espaço de cores RGB para o espaço de escala de cinza. Em seguida, é realizada a correção de ângulo, a operação de limiariazação e aplicação da operação morfológica de erosão, responsável por diminuir a espessura dos dígitos, buscando desconectar caracteres. As operações de correção de ângulo e aplicação de limiarização com erosão são mostradas na Figura 7.

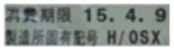
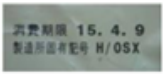
Figura 7: Tratamentos na imagem. (a) Original. (b) Correção de ângulo. (c) Imagem com caracteres conectados. (d) Limiarização e erosão.



Fonte: adaptado de Zaafouri et al (2015).

Em Hosozawa *et al.* (2018), é desenvolvida uma aplicação para uso em *smartphones* com objetivo de identificar datas de validade impressas em embalagens de alimentos industrializados. Os autores propõe que os usuários da aplicação utilizem o *smartphone* para capturar uma foto da data de validade e do item em questão antes de armazená-lo em um refrigerador de itens perecíveis. Após a captura das imagens, a data de validade reconhecida e a foto capturada são armazenadas em um servidor, que se encarregará de realizar o gerenciamento de estoque dos alimentos contidos nele. Foram testados três algoritmos de OCR *open source,* e escolhido o mais adequado para a aplicação. Com base na compatibilidade de sistema operacional, tipos de caracteres reconhecidos e documentação disponível, optou-se pelo OCR Pytesseract. Foi estudada a influência do tamanho da região de interesse sobre a capacidade de reconhecimento de caracteres. Conforme a Figura 8, os autores mostraram que a existência de espaço ao redor da área com o texto de interesse leva a erros de reconhecimento, mesmo em imagens com bom contraste e iluminação.

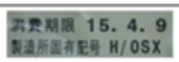
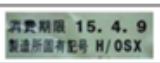
Figura 8: Comparação dos resultados obtidos para diferentes porcentagens de espaço em branco na imagem.

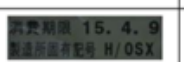
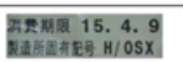
Espaço em branco	Pouco	1/2 da imagem	2/3 da imagem
Imagem de entrada			
Resultado	Reconhecido Corretamente	Reconhecido Incorretamente	Reconhecido Incorretamente

Fonte: adaptado de Hosozawa et al (2018).

As influências de contraste e brilho também foram analisadas, e foi observado que, para valores de contraste e brilho de -40%, 0% (padrão) e +40%, os resultados obtidos são idênticos, sendo reconhecidos corretamente os caracteres, conforme Figura 9.

Figura 9: Influência dos diferentes níveis de brilho e contraste no reconhecimento de caracteres.

Contraste	-40%	0%	+40%
Imagem de entrada			
Resultado	Reconhecido corretamente	Reconhecido corretamente	Reconhecido corretamente

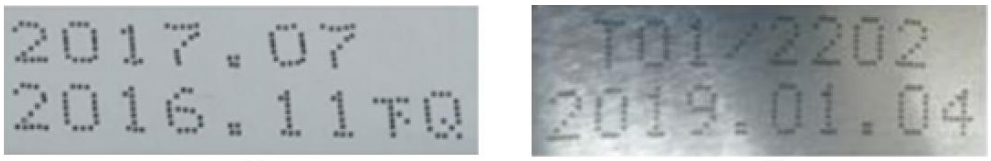
Brilho	-40%	0%	+40%
Imagem de entrada			
Resultado	Reconhecido corretamente	Reconhecido corretamente	Reconhecido corretamente

Fonte: adaptado de Hosozawa *et al* (2018).

Segundo os autores, nenhuma data de validade com caracteres pontilhados foi reconhecida corretamente, bem como no caso em que os caracteres estão em posição

diferente da horizontal. Não é mencionado o número de embalagens testadas com este tipo de fonte. Datas de validade impressas em garrafas PET e de vidro foram testadas, e é mencionado que em garrafas com menos de 5cm de diâmetro os caracteres não são reconhecidos corretamente, por conta da distorção natural causada pelo formato das garrafas. Quanto à influência da luz, os autores mencionam que mesmo pequenas quantidades de luz refletida pela superfície da embalagem ocasionam erro no reconhecimento de caracteres. A Figura 10 demonstra o caso da fonte pontilhada e da influência do reflexo da luz. A Tabela 1 apresenta uma compilação de problemas encontrados e soluções propostas pelos autores.

Figura 10: (a) Data de validade com fonte pontilhada não reconhecida corretamente. (b) Data de validade sujeita a influência de reflexos de luz, cujos caracteres não foram reconhecidos corretamente.



(a) (b)

Fonte: adaptado de Hosozawa *et al* (2018).

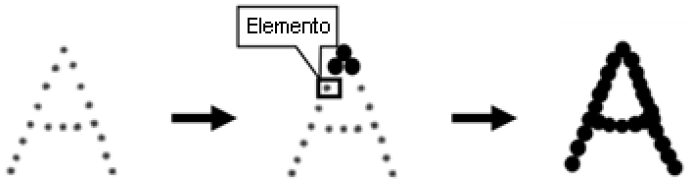
Tabela 1: relação entre problemas e soluções propostas para melhorar o reconhecimento de caracteres.

Problema	Solução Proposta
(1) Imagens incluindo grandes espaços sem texto sem reconhecidos com erros	Cortar a imagem rente a zona de interesse
(2) Todos os caracteres fora da posição horizontal são reconhecidos com erro	Rotacionar para a horizontal
(3) Algumas cores de fundo e de caracteres são reconhecidas com erro	Converter o fundo e as letras para preto e branco
(4) Fontes pontilhadas são reconhecidas com erro	Conectar os pontos
(5) Fontes com linhas finas são reconhecidas com erro	Espessar as linhas dos caracteres
(6) Caracteres muito distorcidos são reconhecidos com erro	
(7) Imagens com reflexão são reconhecidas com erro	

Fonte: adaptado de Hosozawa *et al* (2018).

Tendo em vista que a proposta da pesquisa é o desenvolvimento de um aplicativo *Android*, os itens (1) e (2) são resolvidos diretamente no aplicativo, onde é possível rotacionar e cortar a imagem como o usuário desejar. O problema (3) é resolvido com a limiarização, e os problemas (4) e (5) com o método de dilatação, conforme demonstra a Figura 11. Os autores não propõem uma solução para os problemas (6) e (7).

Figura 11: Demonstração da técnica de dilatação utilizada para conectar os pontos dos caracteres.



Fonte: adaptado de Hosozawa *et al* (2018).

O trabalho desenvolvido por Gong *et al.* (2018) visa identificar as datas de validade de embalagens industriais através das etapas de detecção da localização do texto e posterior utilização de algoritmos de OCR para reconhecimento. As imagens utilizadas possuem texto em diversas orientações, as embalagens são variadas e, portanto, possuem padrões diferentes de codificação, mudando a forma como esta é apresentada, assim como cores das letras e cores de fundo. Para reconhecer a região de interesse, foi realizado um ajuste fino em uma rede neuronal de detecção de texto, conhecida como EAST (Zhou et al., 2017). O ajuste consiste em utilizar uma rede já existente e treiná-la com um conjunto de dados específico. O algoritmo foi treinado com 800 imagens, das quais 70% foram utilizadas para treino e 30% para teste. Este ajuste permitiu que o único texto identificado nas imagens seja a data de validade, conforme Figura 12. Quanto à detecção da região da data de validade, foi obtida uma taxa de acerto de 98% sobre as 240 imagens de teste. Para reconhecimento de caracteres, foi utilizada a biblioteca Pytesseract. Os autores constataram que podem ocorrer erros de reconhecimento nos casos em que o texto está desfocado ou borrado, e não foram apresentados dados quanto à acurácia deste reconhecimento.

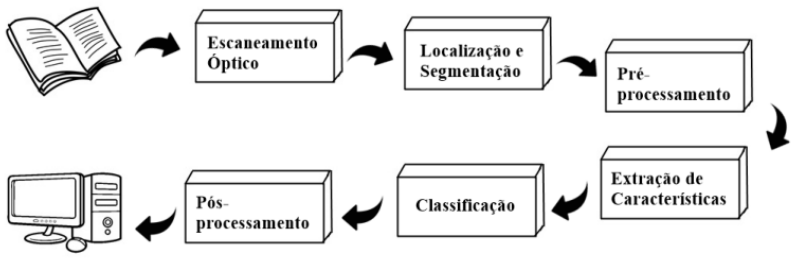
Figura 12: Exemplo de imagens utilizadas no treinamento do modelo. (a) Texto detectado antes do ajuste fino. (b) Texto detectado após o ajuste.

Fonte: Adaptado de Gong *et al* (2018).

2.5.2 Estudo de comparação entre algoritmos de OCR

Cunha (2018) contextualiza a história do desenvolvimento de algoritmos de OCR, desde o surgimento da tecnologia, mudanças nos processos, principais desafios enfrentados, até chegar à forma como é utilizado hoje, apoiando-se principalmente em redes neuronais. O estudo explica que um sistema de OCR tipicamente aplica as seguintes técnicas descritas no esquema da Figura 13.

Figura 13: Esquema de técnicas típicas envolvidas no reconhecimento de caracteres.



Fonte: Adaptado de Cunha (2018).

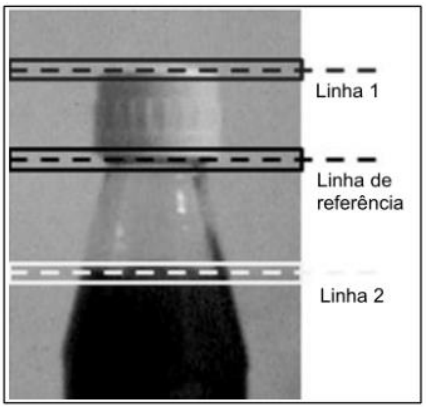
Em Cunha(2018), foram analisadas 18 ferramentas de OCR disponíveis no mercado. Com base nos critérios de disponibilidade, suporte a linguagens de programação, formatos de imagens e custo, foram selecionadas nove para serem utilizadas nos testes propostos pelo estudo: Tesseract; Google Cloud Vision API, GOCR, ABBYY Cloud OCR SDK, OCR Document Haven On Demand, OCR.Space API, New OCR API, Microsoft Vision API, OCR Web Service REST API, SemaMediaData OCR e Cloudmersive. O estudo mensura a acurácia das nove ferramentas de OCR aplicadas a cinco imagens, sendo elas: um documento auxiliar da nota fiscal eletrônica (DANFE), uma nota de Empenho pública, uma Portaria do Ministério da Educação, uma imagem da letra do Hino Nacional e uma imagem de um poema de Fernando Pessoa, que utiliza palavras e acentuações pouco usuais atualmente. Cada imagem foi testada nas resoluções de 100, 300 e 600dpi, e uma versão em foto. Segundo o autor, as ferramentas que obtiveram as melhores acurácias no conjunto de amostras foram a OCR Web Service, Tesseract, OCR.Space API, ABBYY Cloud Vision API, Google Cloud Vision API e Microsoft Vision API. Para as amostras em versão foto, nota-se que as ferramentas OCR Web Service e Google Cloud Vision API mostraram-se consistentemente como as que obtêm melhor acurácia.

2.5.3 Estudos com a utilização de Raspberry Pi para aquisição de imagens de objetos em movimento visando detecção de não conformidades

O método proposto por Sharma *et al.* (2015) realiza uma inspeção em frascos PET, a fim de identificar a existência de líquido no frasco, o raio da base e do topo, o nível de enchimento, e a colocação da tampa adequada. A distinção entre frasco cheio e vazio é feita com base nos métodos de remoção de ruído e aumento de contraste. Para as inspeções de topo e fundo, foram aplicados métodos de procura por circunferências. A presença de tampa e o correto nível de enchimento são definidos com base na distância entre duas linhas detectadas e de uma linha de referência, sendo que a linha de referência fica localizada entre as outras duas linhas. A distância entre os pares de linhas deve ser igual à altura da tampa, e, para o nível de enchimento, a distância deve ser igual à um valor

determinado. Se qualquer uma das linhas não for identificada, o algoritmo classificará o frasco como inadequado. A Figura 14 ilustra o método utilizado.

Figura 14: Linhas de referência utilizadas para detecção de presença de tampa.



Fonte: adaptado de Sharma *et al.* (2015).

Os autores variaram as condições de iluminação dos experimentos, constatando que uma iluminação inadequada pode gerar falhas nas detecções. A acurácia obtida foi de 100% nas inspeções de nível e presença de tampa nos casos com iluminação adequada, e 88% de acurácia com a iluminação inadequada. Os autores também mencionam que é necessário garantir que a garrafa esteja no centro da imagem capturada. Portanto, a detecção de linhas verticais é realizada, e apenas se a posição das linhas estiver na região adequada, o processamento será realizado. A distância entre a garrafa e a câmera é de aproximadamente 30 cm, e foi colocada uma tela de fundo branca para as inspeções de nível e qualidade de tampa. Não são mencionados valores de velocidade da esteira ou tempo de processamento das imagens.

Em Machado (2019), foi realizado o desenvolvimento e avaliação de um sistema para inspeção de frascos em linha de produção, utilizando uma placa Raspberry Pi 3 Model B. Foram avaliados dois modelos de frasco, um fabricado em material PEAD de cor branca com tampa azul, e outro fabricado em PET de cor verde com tampa verde. O objetivo do sistema proposto foi detectar a não-conformidade de ausência de tampa, e foi avaliado com base na velocidade da linha de produção de uma envasadora de produtos saneantes. Foi utilizado o método de iluminação direta com lâmpadas LED. A câmera utilizada foi a Raspberry Pi Camera V2. Ao todo, foram capturadas e analisadas 313 imagens de frascos brancos e 125 imagens de francos verdes, sendo que as imagens foram divididas entre frascos com e sem tampa. A velocidade nominal da linha foi de 8160 frascos/hora. O tempo de exposição utilizado foi de 1500 µs, com uma resolução de 1280x720 pixels. O autor menciona que, devido às limitações impostas pelo sistema de iluminação, não foi possível utilizar valores de tempo de exposição abaixo de 1500 µs, o que ocasionou algumas distorções nas imagens, como as causadas pelo efeito de *motion blur* e efeito *rolling shutter.* O autor também cita que as capturas apresentaram pequenas variações de posição lateral do frasco, causadas por atrasos no sistema de captura, alta velocidade da esteira de transporte e proximidade da câmera em relação ao frasco. O trabalho realiza uma comparação entre quatro diferentes técnicas de processamento de imagens, nomeadas pelo autor como método das linhas, método das áreas, método *K-means* e método ORB. Observa-se que todas as limiarizações

realizadas foram feitas com o método de Otsu, que consiste em calcular um valor de *threshold* global que maximiza a função de contraste para a imagem. Duas das técnicas aplicadas apresentaram acurácia de 100%, enquanto as outras duas apresentaram acurácia de 98%. O menor tempo de processamento médio obtido foi de 312 ms, na utilização do método das áreas, enquanto o maior tempo de processamento obtido foi de 2,07 segundos, na utilização do método *K-means*.

3 Materiais e Métodos

Nesta seção serão apresentadas as informações referentes ao material e metodologia utilizada para desenvolvimento do trabalho, incluindo o *hardware*, as técnicas de processamento de imagens, ferramentas de OCR, etc. Como testes em linha de produção industrial não foram possíveis durante a realização deste trabalho, foi montada uma bancada experimental para os mesmos.

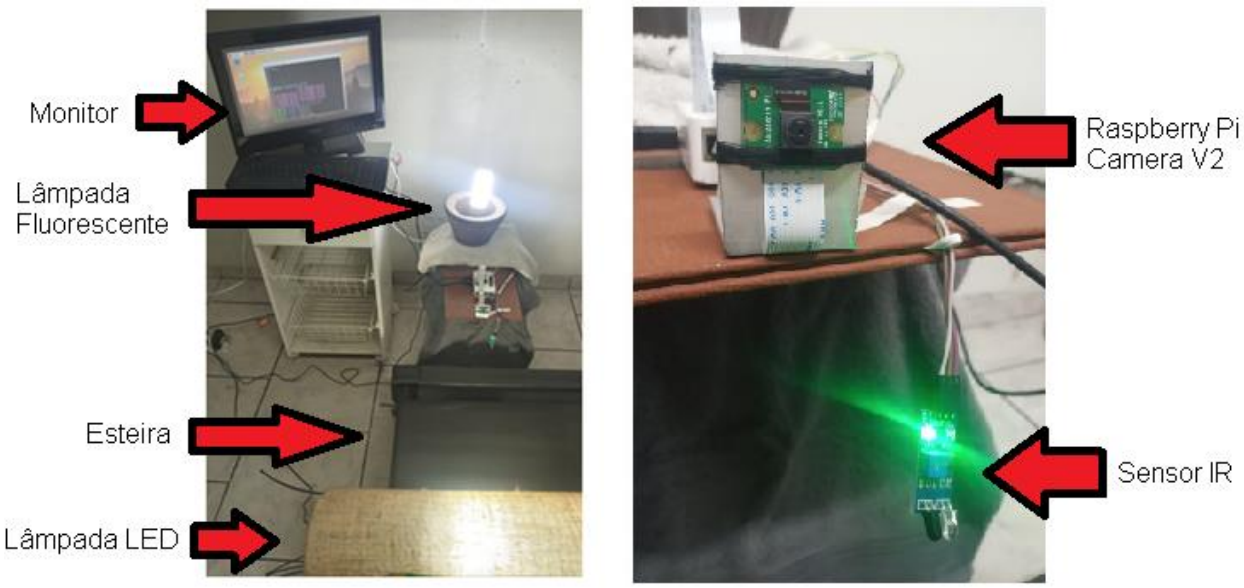
3.1 Montagem do sistema de aquisição de imagens

O sistema utilizado para os testes consiste em uma placa microprocessada Raspberry Pi Model 3B, com uma câmera Raspcamera V2 acoplada via cabo CSI e com um sensor de infravermelho conectado aos pinos da Raspberry através de cabos jumpers. A utilização desta câmera permite o ajuste de parâmetros como *shutter speed*, brilho, resolução, formato da imagem, saturação, entre outros. A especificação dos componentes é apresentada no Apêndice B, na Apêndice B

Tabela B.1, Tabela B.2 e Tabela B.3.

Para simular a linha de produção, foi utilizada uma esteira de corrida doméstica. As garrafas são colocadas manualmente no começo da esteira e o conjunto com o sensor é posicionado no final da esteira. Quando a presença da garrafa é detectada pelo sensor, a foto é capturada e processada. A distância entre a câmera e a garrafa na esteira é de aproximadamente 5cm. Foram utilizadas duas lâmpadas alinhadas à posição da câmera, uma atrás da garrafa e outra na frente. A lâmpada traseira é de LED com fluxo luminoso de 2250lm e a dianteira é fluorescente com 2040lm. O sistema montado pode ser visto na Figura 15.

Figura 15: Montagem do sistema de captura de imagens.



Fonte: Autor

A esteira utilizada possui um sistema de modos de velocidade próprio, os quais foram nomeados de “1” a “5”, sendo “1” corresponde a menor velocidade e “5” à maior. Foi feita uma marcação na esteira e a velocidade foi calculada com base no tempo necessário para completar determinado número de voltas em cada modo. A correspondência entre modos e velocidades obtidas é mostrada na Tabela 2. Neste trabalho, a velocidade utilizada para todas as amostras foi de 0,40 m/s.

Tabela 2: relação entre modos da esteira e velocidade em m/s.

MODO	VOLTAS	TEMPO(s)	VELOCIDADE (m/s)
1,5	10	58,24	0,40
2	10	42,35	0,55
2,5	12	41,51	0,68
3	13	37,26	0,82
3,5	15	37,22	0,95
4	18	39,06	1,08
4,5	18	34,56	1,22
5	20	34,77	1,35

3.2 Desenvolvimento do algoritmo de processamento de imagens

Foram realizadas duas etapas de testes de sequências e métodos de processamento de imagens. Para a captura das imagens utilizadas nos testes, foi desenvolvido um algoritmo responsável apenas pela aquisição das fotos, onde são ajustados os parâmetros *shutter speed*, *brightness* e resolução. Este algoritmo é executado na Raspberry Pi, e a captura da foto está condicionada ao recebimento do sinal de presença do sensor de infravermelho utilizado. O parâmetro *shutter speed* utilizado foi de 800 µs. Após a aquisição das fotos, as mesmas foram extraídas da Raspberry e descarregadas em um computador onde foram processadas com outro algoritmo, responsável por efetuar o tratamento das fotos e a avaliação da qualidade da codificação. As etapas utilizadas no tratamento das fotos foram determinadas com base nas recomendações existentes na literatura, descritas na Seção 2.5 sobre pré-tratamento para OCR. As etapas de detecção e reconhecimento de texto foram realizadas com auxílio de ferramentas de OCR, enquanto os tratamentos das fotos utilizaram a biblioteca OpenCV. Todos os algoritmos desenvolvidos utilizaram a linguagem de programação Python na versão 3.8.0. Os resultados obtidos foram analisados, e a partir deles foram realizados ajustes necessários para melhorar a qualidade da classificação dos códigos. Após a definição do método mais adequado, seguiu-se para o desenvolvimento de uma rotina capaz de efetuar o processamento em tempo real, cujo detalhamento está descrito no Apêndice C.

3.3 Projeto dos experimentos

O sistema de inspeção foi testado em dois modelos distintos de garrafa PET. O modelo A apresenta a codificação em região logo acima do rótulo, sem curvatura no texto. O modelo B apresenta a codificação próxima à tampa e com uma leve curvatura no texto. Os modelos são apresentados na . Foram adquiridas cinco garrafas de cada modelo, todas de lotes diferentes, para garantir uma maior variabilidade nos caracteres presentes na codificação.

Cada garrafa foi testada no sistema três vezes, totalizando 15 amostras de cada modelo. Além destas, foi utilizada três garrafas de cada modelo com alguma das não conformidades a seguir: falta de codificação, falta de uma letra, letra borrada. Cada garrafa foi passada três vezes pelo sistema, totalizando nove amostras defeituosas. Os resultados foram expressos utilizando matrizes de confusão e as métricas definidas nas Equações (3.1) a (3.3).

Figura 16: Exemplo de codificação dos modelos. (a) Modelo A. (b) Modelo B.

(a) (b)

Fonte: Autor.

3.4 Ferramentas de OCR utilizadas

3.4.1 Tesseract

O algoritmo Tesseract começou a ser desenvolvido entre os anos de 1985 e 1994 pela Hewlett Packard (HP), na linguagem de programação C, sendo reescrito para C++ em 1998. Em 2015 o projeto se tornou *open-source*, e atualmente é mantido pela Google sob a licença Apache 12. Dentre os fatores que a tornam amplamente utilizada, podemos citar: quantidade de documentação disponível, qualidade nos resultados, suporte a integração com diversas linguagens de programação, e capacidade de reconhecer mais de 100 idiomas, além de permitir uso livre. A versão mais recente é a 5.0.0 (Tesseract, 2021).

3.4.2 Google Vision Cloud API

É um conjunto de serviços em nuvem que envolve reconhecimento de imagens, como detecção de marcas e rótulos, reconhecimento facial, reconhecimento óptico de caracteres, reconhecimento de objetos e pontos de referência. A API de reconhecimento de texto é capaz de detectar grande quantidade de idiomas, além de fornecer integração e suporte às linguagens C#, GO, Java, Node.JS, PHP, Python e Ruby. Possibilita realizar até 1000 requisições mensais gratuitamente para a API, mediante criação de conta de usuário na plataforma Google Cloud (Google, 2021a).

3.4.3 Easy OCR

Desenvolvida pela Jaided AI, empresa fundada em 2020, tem o intuito de ser uma ferramenta gratuita, *open-source*, fácil de utilizar e de ajustar conforme cenários específicos. Possui uma série de parâmetros ajustáveis, como a linguagem a ser reconhecido, tamanho dos caracteres, nível de confiança mínimo para detecção, caracteres permitidos, distância mínima entre *bounding boxes*. A ferramenta permite a utilização para detecção de texto e para reconhecimento, individualmente ou conjuntamente, e é capaz de encontrar caracteres em qualquer ângulo, embora seja recomendado que estejam na horizontal (JaidedAI, 2021).

3.5 Biblioteca OpenCV

A biblioteca Open Source Computer Vision Library (OpenCV) começou a ser desenvolvida pela Intel no ano de 2000, nas linguagens de programação C/C++, e atualmente é de uso livre para escopo acadêmico e comercial, sob a licença BSD Intel. É utilizada para o desenvolvimento de software na área de Visão Computacional, e possui módulos de processamento de imagens, vídeos, estruturas de dados, álgebra linear, entre outros. Conta com mais de 350 algoritmos para as mais variadas funções como filtros de imagens, reconhecimento de objetos e faces, detecção de movimento, entre outras, e extensa documentação no site oficial, bem como diversos exemplos de uso em fóruns especializados. Possui suporte às linguagens Java, Python e Visual Basic, e a versão estável mais recente na data de realização desta pesquisa é a 4.5.2 (OpenCV, 2021).

3.6 Web Sockets

A utilização de Web Sockets permite o estabelecimento de uma sessão de comunicação bidirecional interativa entre o navegador de um usuário e um servidor. Esta API permite que as mensagens trocadas entre servidor e cliente sejam orientadas a eventos, isto é, ambas as partes podem enviar mensagens a qualquer momento, dada a ocorrência de um evento específico. A tecnologia é baseada no protocolo de comunicação TCP, e foi projetada para utilização em *browsers* HTML5, embora seja possível utilizar diretamente dentro de aplicações que tenham alguma biblioteca de suporte à tecnologia, como PHP, Node.js, Java, Ruby, Python, entre outras (Mozilla, 2021).

3.7 Avaliação de Desempenho

Para a avaliação de desempenho do algoritmo de classificação de conformidade, será utilizada uma matriz de confusão, também chamada de matriz de erro, conforme Figura 17. Essa matriz classifica as predições de sistema com saída binária em quatro categorias: verdadeiro positivo (VP), verdadeiro negativo (VN), falso positivo (FP) e falso negativo (FN).

Figura 17: Matriz de confusão genérica.

		Classificação atual	
		P	**N**
Classificação prevista	**P**	**VP**	FP
	N	FN	**VN**

Fonte: Adaptado de Google (2021).

Para se utilizar a matriz, é necessário definir qual classe será considerada como positiva e qual será a negativa. No escopo deste trabalho, será adotada a convenção de que a classificação “Conforme” será a classe Positiva, e “Não-conforme” será a Negativa. Os valores VP correspondem à situação em que o algoritmo classifica uma garrafa como conforme quando ela é realmente conforme, e os valores VN são análogos para os casos não-conformes. Os valores FP ocorrem quando a garrafa é classificada como conforme, mas

deveria ser classificada como não-conforme, e os valores FN ocorrem quando a garrafa é classificada como não-conforme, mas deveria ser classificada como conforme. A partir destas classificações, define-se as seguintes métricas de avaliação de desempenho:

$$Acurácia = \frac{VP+VN}{VP+VN+FP+FN} \tag{3.1}$$

$$Sensibilidade = \frac{VP}{VP+FN} \tag{3.2}$$

$$Especificidade = \frac{VN}{VN+FP} \tag{3.3}$$

A *Acurácia* é considerada uma das métricas mais simples e importantes. Ela avalia o percentual de acertos, ou seja, a razão entre a quantidade de predições corretas e o total de entradas. A *Sensibilidade* avalia a capacidade de detectar corretamente os resultados classificados como positivos, enquanto a *Especificidade* faz o mesmo para os resultados classificados como negativos. Considerando o escopo deste trabalho, entende-se que classificar uma garrafa não-conforme como conforme (FP) é mais prejudicial que classificar uma conforme como não-conforme (FN), portanto, espera-se obter um alto valor de especificidade (Google, 2021b). Deve-se observar que os problemas de falso negativo podem ser reduzidos, ao menos num certo nível, por estratégias adequadas de análise sequencial das codificações.

3.8 Critério de conformidade de codificação

Para avaliar a qualidade da codificação, foi desenvolvido um algoritmo que verifica se a quantidade de letras e números adequada foi encontrada, bem como se os caracteres fixos foram encontrados. Os caracteres fixos e variáveis de cada modelo são os seguintes:

- ➔ Modelo A: os caracteres “VAL”, “PET-PCR” e “L:PS” são fixos, enquanto todos os dígitos são variáveis. Os caracteres fixos devem ser sempre encontrados.

- ➔ Modelo B: Notou-se que os caracteres “VAL” e “FAB” são fixos, enquanto os últimos três caracteres da primeira linha são variáveis, e todos os dígitos são variáveis. Os caracteres fixos devem ser encontrados.

Para os caracteres comuns a todas as garrafas de mesmo modelo, foram adicionados critérios condicionais que aceitam a troca de algumas letras por outras que são semelhantes e podem apresentar difícil reconhecimento pelo OCR, prejudicando desnecessariamente o método. As trocas permitidas pelo algoritmo são mostradas na Tabela 3.

Tabela 3: equivalências permitidas para os caracteres

Permutações Permitidas
V, v, U, u, W, N
A, R
P, F
B, E
S, 8, 9, 3
C, O

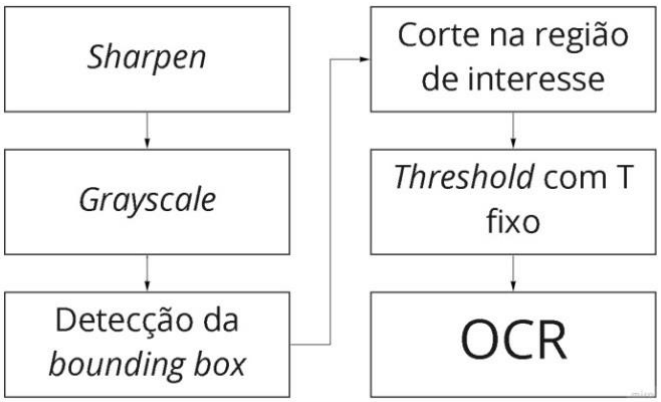
4 Processamento das imagens

No decorrer do trabalho, foram realizadas duas sequências de tratamentos. A primeira sequência teve caráter exploratório, servindo como base para a construção do algoritmo final. A partir dos resultados desta, foi proposta uma segunda sequência, visando melhorar algumas das falhas encontradas. As sequências de tratamentos utilizadas são explicadas a seguir.

4.1 Combinação Inicial de Métodos

A primeira sequência testada aplicou os métodos descritos na Figura 18. O método *sharpen* é utilizado para aumentar o contraste em uma imagem digital. A Figura 19 mostra o resultado da utilização deste tratamento. Nota-se que o contraste entre os caracteres e o fundo da imagem torna-se mais evidente na imagem (b). Em seguida, é aplicada uma transformação para mapear a imagem do espaço de cores RGB para o espaço de escala de cinza, etapa que reduz significativamente o tempo necessário de processamento, pois diminui a quantidade de informação que a imagem carrega. Para realizar a transformação, foi utilizado o método "COLOR_BGR2GRAY" disponível na biblioteca OpenCV.

Figura 18: Sequência de métodos aplicada inicialmente.



Fonte: Autor.

Figura 19: Aplicação do método *Sharpen*. (a) Original. (b) após o *Sharpen.*

Fonte: Autor.

Após, é realizada uma operação de detecção da localização do texto , visando obter as coordenadas da *bounding box*. Os testes realizados mostraram que o método é capaz de

detectar corretamente a localização do texto, porém, em alguns casos, são encontradas mais de uma *bounding box* ou as coordenadas encontradas são muito próximas aos caracteres das bordas, gerando problemas no reconhecimento que será feito posteriormente.

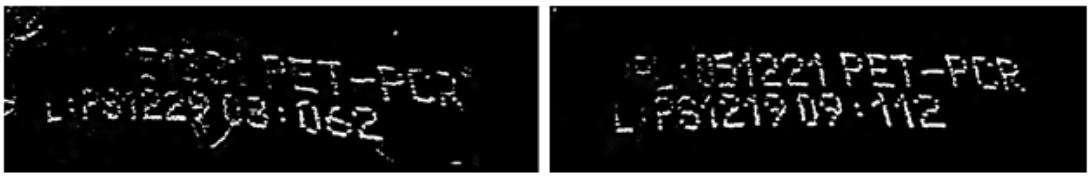
De posse das coordenadas do texto de interesse, é realizado um corte nesta região, seguido da operação de *limiarização*, isto é, a transformação da imagem para o espaço de cores preto ou branco, buscando obter o máximo contraste possível entre caracteres e fundo da imagem. Os testes realizados utilizaram parâmetros fixos para o valor de *threshold* (*T*) do método, conforme exemplificado na Figura 20.

Figura 20: Demonstração do efeito do valor *threshold*. (a) T = 140 (b) T = 160 (c) T = 180.

Fonte: Autor.

Ao testar a qualidade da leitura de texto na Figura 20, a versão com *T* = 160 obteve 100% dos caracteres reconhecidos corretamente, enquanto a versão com *T* = 140 falhou em reconhecer alguns caracteres, como os dígitos “1” e “6”. A versão com *T* = 180 não foi capaz de discernir o número “2”, nem a letra “F” devido à interferência de regiões escuras captadas. Assim, para esta sequência foi utilizado o valor de *T* = 160. Os resultados forneceram 33,3% de acurácia para o modelo A e 20,8% para o modelo B. Foi observado que, ao utilizar um valor fixo de *T* nas amostras, pequenas variações na iluminação das garrafas provocaram grandes variações no reconhecimento do texto, o que resultou numa acurácia baixa. A Figura 21 exemplifica alguns casos em que não foi possível identificar corretamente alguns dos caracteres.

Figura 21: casos em que o uso de um valor fixo de *T* não foi adequado para separar corretamente os caracteres do fundo da imagem.

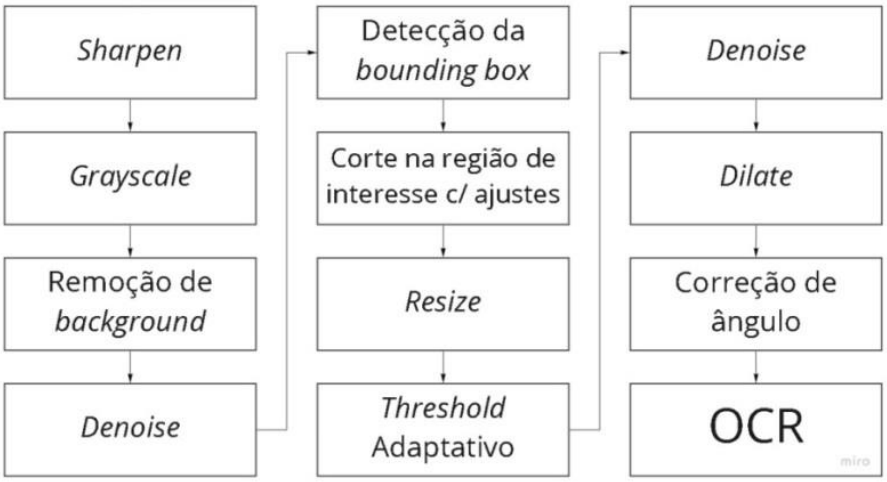


Fonte: Autor

4.2 Combinação Otimizada de Métodos

A partir dos problemas encontrados, foram realizados ajustes na sequência anterior, obtendo-se uma nova sequência, conforme mostra a Figura 22.

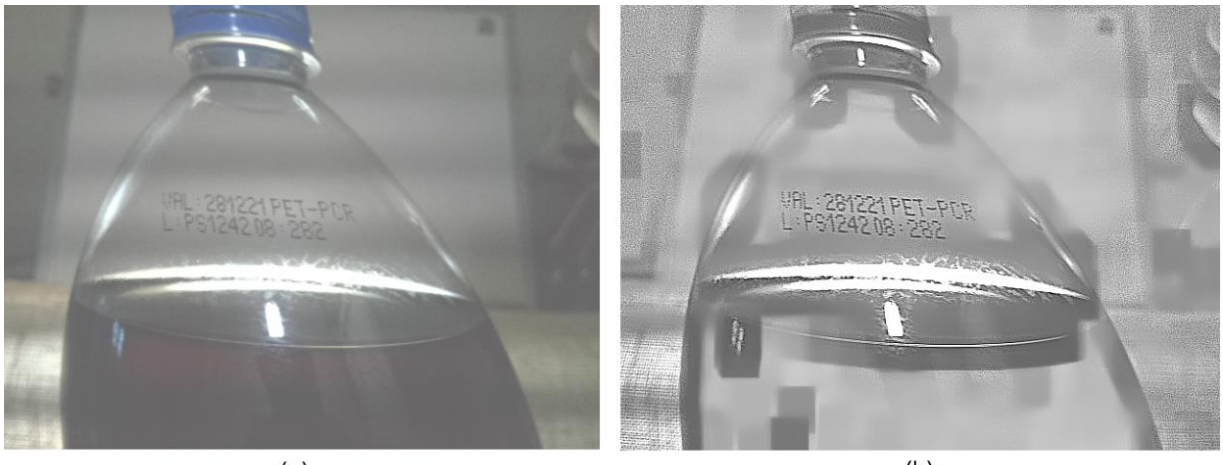
Figura 22: sequência de métodos utilizada após os ajustes realizados.



Fonte: Autor.

A nova sequência mantém a utilização dos métodos *sharpen* e a transformação para *grayscale.* Além destes, foram inseridas as operações de remoção de *background*, responsável por ofuscar a imagem nas regiões de baixo contraste, e *denoise*, responsável por eliminar o ruído com base na intensidade dos pixels da vizinhança. O sucesso de ambos os métodos depende do tamanho do *kernel* especificado. No primeiro, deve ser condizente com o tamanho do texto na imagem, e no segundo, com o tamanho dos espaços normalmente encontrados entre caracteres. Os efeitos destes quatro tratamentos podem ser vistos na Figura 23.

Figura 23: (a) Original. (b) Tratada com método *sharpen,* transformada para *grayscale,* e aplicação de remoção de background e *denoise.*

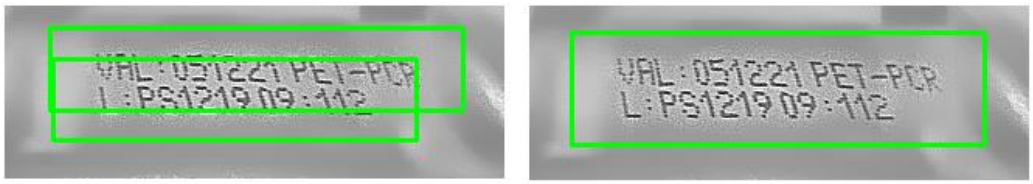


(a) (b)

Fonte: Autor.

O método de detecção de texto foi ajustado para incluir uma margem de segurança às coordenadas encontradas, sendo esta de 12 pixels para a direção vertical e 25 pixels para direção horizontal. Além disso, foram empregados alguns parâmetros opcionais, como a distância mínima entre duas *bounding boxes* para que elas sejam consideradas idênticas. Com o uso desses parâmetros e ajustes no método, foi possível obter uma *bounding box* única que engloba o texto corretamente em todas as imagens, conforme exemplificado na Figura 24, onde o retângulo em verde é formado pelas 4 coordenadas encontradas pelo método *detect text*.

Figura 24: Demonstração dos ajustes realizados. (a) Sem especificação de distância mínima. (b) Com utilização de parâmetros para distância mínima.

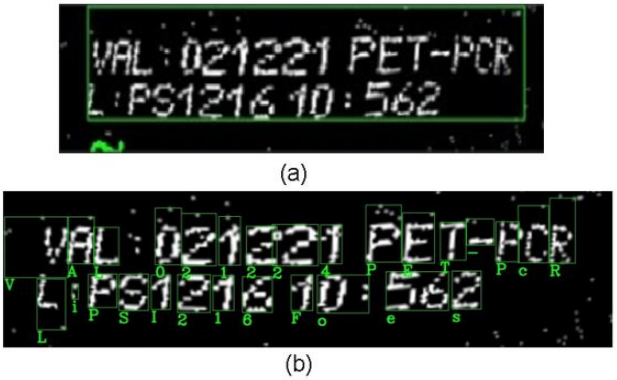


(a) (b)

Fonte: Autor.

A partir da localização do texto, é realizado um corte na imagem, seguido de uma operação de reescalonamento. Esta operação é realizada pois os testes mostraram que o tamanho dos caracteres é importante, ao passo que um mesmo algoritmo deixa de reconhecer alguns caracteres quando estes estão abaixo do tamanho mínimo recomendado. Este efeito pode ser visto na Figura 25, onde os caracteres em verde representam os valores reconhecidos pelo OCR Pytesseract.

Figura 25: Efeito do tamanho da imagem no reconhecimento de caracteres. (a) Original. (b) Aumentada em quatro vezes.



(a)

(b)

Fonte: Autor.

No caso em que a imagem com tamanho original é utilizada, nenhum caractere é identificado. Na imagem aumentada, é possível identificar 22 de 25 caracteres corretamente. Este efeito mostrou-se especialmente importante para o Pytesseract, embora os OCRs EasyOCR e Google Vision API também apresentem melhores resultados com imagens reescalonadas.

O próximo tratamento é realizado apenas nas garrafas do modelo B, pois nestas o texto de interesse apresenta uma leve inclinação em formato de arco, que diminui o desempenho dos algoritmos de OCR. Portanto, é realizado um tratamento de distorção em arco no sentido contrário ao da inclinação da imagem original, sendo informado como argumento para o método de distorção apenas o ângulo desejado, que foi definido em 30°. A Figura 26 mostra um exemplo de uma garrafa modelo B após a distorção em arco, e o texto reconhecido pelo Google Vision API.

Figura 26: utilização da distorção em arco. (a) Imagem original. (b) Imagem após aplicação do método, com ângulo de 30°.

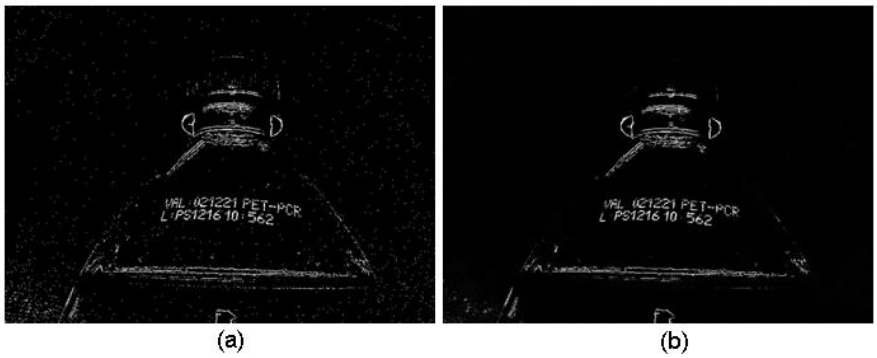
Fonte: Autor.

A partir da nova imagem, centrada na região de interesse e aumentada em quatro vezes, é realizado o tratamento de *threshold* adaptativo. Este tratamento permite que pequenas mudanças na iluminação, como reflexos na garrafa, ou a ocorrência de gotas na região da codificação, afetem menos o resultado final do OCR, pois é calculado um valor de *T* que maximiza o contraste para cada região da imagem, de acordo com o tamanho de *kernel* definido.

O baixo valor de *shutter speed* necessário para capturar a imagem sem borrões ocasiona alguns efeitos indesejáveis em todas as imagens, como regiões horizontais escuras que se repetem periodicamente ao longo da imagem. Estas regiões são mais escuras nas bordas da imagem, pois é onde há menor incidência de luz, já que a iluminação utilizada foi disposta com foco no centro das fotos. O uso de *threshold* adaptativo também apresenta uma vantagem em relação a este efeito, pois, caso uma dessas faixas escuras ocorra diretamente sobre a região da codificação, separar o texto de interesse do *background* torna-se muito difícil com o uso de um único valor *T* para toda a imagem. A Figura 27(a) exemplifica o efeito do *threshold* adaptativo numa imagem sem corte, para facilitar a visualização do tratamento.

Finalmente, é realizada novamente uma operação de *denoise* para retirar eventuais pontos de ruído, como mostra a Figura 27(b). Após, é realizada a operação morfológica de dilatação. Esta operação é responsável por tornar mais salientes algumas partes dos caracteres de interesse, além de permitir que os pontos que compõem a codificação possam se conectar uns aos outros, melhorando o desempenho do OCR utilizado. O efeito desta operação é exemplificado na Figura 28. Nota-se que com o *kernel* 5x5, começa a haver sobreposição de partes dos caracteres, o que afeta negativamente o reconhecimento de caracteres, conforme descrito em Zaafouri, Sayadi e Fnaiech (2015). Por fim, realiza-se uma correção de ângulo, conforme Figura 29. O ângulo é calculado como o arco-tangente da inclinação obtida pela diferença de altura entre o pixel branco mais à esquerda da imagem, e o pixel branco mais à direita.

Figura 27: Demonstração dos métodos de *threshold* Adaptativo e *denoise* em imagem sem corte. (a) Threshold Adaptativo. (b) *denoise*.

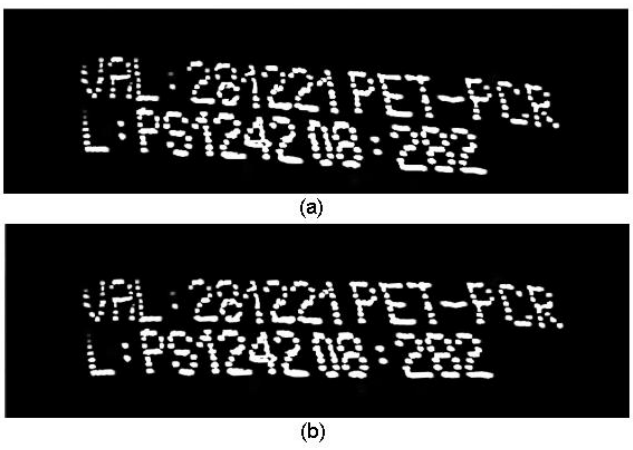


Fonte: Autor.

Figura 28: Demonstração dos tratamentos de dilatação. (a) Sem tratamento. (b) *Kernel* 3x3 (c) *Kernel* 5x5.

Fonte: Autor.

Figura 29: Exemplo do método de correção de ângulo. (a) Original. (b) Ângulo corrigido.



Fonte: Autor.

5 Resultados e Discussão

Os resultados obtidos com a sequência otimizada foram expressos em forma de matriz de confusão, bem como foram calculadas as métricas de Acurácia, Sensibilidade e Especificidade para ambos os modelos de garrafa testados, conforme apresentado na Figura 30.

Figura 30: resultados obtidos para os modelos A e B.

MODELO A

		Classificação Esperada	
		P	N
Classificação Realizada	P	13	0
	N	2	9

Acurácia	91,7%
Sensibilidade	87%
Especificidade	100,0%

MODELO B

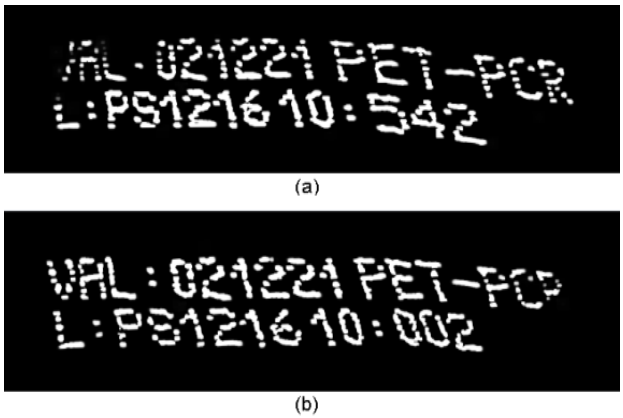
		Classificação Esperada	
		P	N
Classificação Realizada	P	11	0
	N	4	9

Acurácia	83,3%
Sensibilidade	73%
Especificidade	100,0%

Fonte: Autor.

Pode-se observar que a classificação do modelo A obteve acurácia superior à do modelo B, e ambos obtiveram especificidade de 100%, isto é, nenhuma garrafa não-conforme foi classificada como conforme. A classificação incorreta de duas garrafas do modelo A se deu por conta da não detecção dos caracteres posicionados nas extremidades da primeira linha. A iluminação ambiente não foi suficiente para fornecer o contraste adequado para estes caracteres, o que fez com que o método de *threshold* adaptativo, aliado ao método seguinte de *denoise*, considerasse parte destes caracteres como fundo da imagem. A Figura 31 mostra as imagens das garrafas do modelo A classificadas incorretamente pelo algoritmo. Nota-se que em (b) não foi possível reconhecer o caractere “R”, mesmo permitindo a equivalência do mesmo por “P”, cenário que já havia sido previsto. Este problema referente aos caracteres das extremidades poderia ser resolvido utilizando uma câmera posicionada a uma distância maior. Porém, para isto, seria necessário também uma qualidade de imagem superior, pois quanto maior a distância entre câmera e texto, menor a resolução da área de interesse. Outra solução é melhorar as condições de iluminação, diminuindo a ocorrência de sombras, e utilizar luz difusa, conforme sugerido em Silva (2020).

Figura 31: Imagens classificadas incorretamente pelo algoritmo. (a) Não reconheceu “V”. (b) Não reconheceu “R”.



Fonte: Autor.

Quantos aos cenários de não-conformidade, todos foram classificados corretamente. Os algoritmos de OCR se mostraram extremamente sensíveis a pequenas mudanças de ângulo e iluminação. Dado que todos os caracteres do modelo A são fixos, qualquer leve borrão é suficiente para classificar como não-conforme. No conjunto de imagens deste modelo, nenhum dígito foi reconhecido erroneamente, embora exista a chance de uma má codificação ocasionar em um falso positivo, isto é, o OCR não reconhecer o dígito correto, porém reconhecer outro dígito qualquer, assim satisfazendo as condições de conformidade.

As imagens do modelo B classificadas incorretamente tiveram motivos semelhantes aos que levaram às classificações incorretas do modelo A, acrescidas de ocorrências em que um dígito foi reconhecido como um caractere. As regras propostas permitiram a troca de alguns caracteres por outros, inclusive por dígitos, mas não foi permitida a troca de nenhum dígito por caractere. Na segunda linha de texto do modelo B, que possui 10 dígitos, ocorreu a troca de um dígito “0” pelo caractere “Q”, bem como do dígito “7” pelo caractere “T”, conforme Figura 32. Neste modelo, devido ao fato de que os últimos três caracteres da primeira linha são variáveis, é importante salientar que há chance de que ocorra a troca de um caractere por outro, ocasionando uma classificação Conforme. Nos testes realizados, em todos os casos que houve troca de caracteres, a resposta esperada era de conformidade, não ocasionando falsos positivos.

Figura 32: Imagens classificadas incorretamente. (a) Troca de “0” por “Q”. (b) Troca de “7” por “T”.

Fonte: Autor.

Além destas, outras duas imagens do modelo B foram classificadas incorretamente, ambas sendo da mesma garrafa, conforme Figura 33. Nota-se que o caractere “F” não está completo, bem como os caracteres “1” e “L” que aparecem em sequência. Este caso mostra claramente a influência do tamanho do *kernel* escolhido, pois a distância entre a parte superior de dois caracteres “L” ou entre a parte inferior de dois dígitos “1” é maior do que entre outros pares de caracteres. Sendo assim, uma provável solução é aumentar o tamanho do *kernel* de *denoise*.

Figura 33: Imagens classificadas incorretamente pelo algoritmo.

Fonte: Autor

A aceitação da troca de alguns caracteres mostrou-se fundamental, especialmente a troca de “V” por “v”, bem como a correção de ângulo, proposta por Zaafouri, Sayadi e Fnaiech (2015). A técnica de dilatação proposta em Hosozawa *et al* (2018) também foi extremamente relevante. O *threshold* adaptativo em conjunto com *denoise* mostrou-se uma solução razoável para o problema da limiarização.

Apesar da posição da câmera e do sensor serem fixas, as imagens capturadas apresentaram pequenas variações na posição da garrafa. Isto pode ter sido causado pela baixa capacidade de processamento da Raspberry Pi utilizada, conforme discutido em Machado (2019). Foi observado que nos casos em que a garrafa não é capturada no centro da imagem, ocorre com maior frequência a não detecção da primeira ou última letra da codificação, provavelmente por conta da menor iluminação disponível nestas regiões. Uma solução que minimiza este problema é desenvolver uma rotina que verifica a posição da garrafa antes de efetuar o processamento, conforme sugerido em Sharma *et al* (2015). A detecção das *bounding boxes* foi realizada com sucesso em todas as imagens, mesmo quando a garrafa não está centralizada, tanto para a ferramenta EasyOCR quanto para a Google Cloud API.

6 Conclusões e Trabalhos Futuros

O objetivo proposto para este trabalho envolve o estudo da viabilidade técnica de se empregar o sistema proposto em uma linha de produção. Portanto, foram realizados testes com dois modelos de garrafas PET, contando com exemplares que apresentam não conformidades. Foram definidas duas propostas de sequência de tratamentos, sendo que a primeira obteve 33,3% e 20,8% de acurácia para os modelos A e B, respectivamente, e a segunda proposta obteve 91,7% e 83,3%, respectivamente, mostrando que as mudanças realizadas foram significativas. Além disso, foi proposta uma regra para avaliação de conformidade, que permite a troca de alguns caracteres específicos, e foi desenvolvido um algoritmo capaz de processar as imagens em tempo real utilizando *web sockets*.

Este trabalho demonstrou que o sistema de baixo custo proposto é capaz de obter uma especificidade de 100% na velocidade testada das garrafas. Assim, este sistema poderia auxiliar na diminuição das possibilidades de que um problema de codificação seja percebido tardiamente, ocasionando custos de não qualidade, ou que esta garrafa chegue ao mercado, sendo passível de oferecer riscos ao consumidor, gerar multas para a empresa, e danificar a sua imagem pública. No entanto, alguns ajustes são necessários na sequencia de etapas de tratamento, que pode, inclusive, depender do tipo de garrafa. Além disso, as velocidades testadas das garrafas estão abaixo daquelas praticadas na indústria, o que é um fator importante para o funcionamento do sistema de verificação. Uma solução para este problema seria o uso de uma câmera de maior qualidade, bem como uma placa microprocessada mais robusta e com especificações técnicas melhores.

Conclui-se que, para a utilização do sistema proposto em um ambiente de produção real, seriam necessários ajustes a fim de evitar que a verificação humana tivesse que ser acionada com frequência. Nestes casos, no entanto, seria possível a implementação de algoritmos de resiliência contra falsos negativos, isto é, caso haja uma detecção de não conformidade, poderia se aguardar a confirmação da situação de não-conformidade em imagens subsequentes até o disparo de um alerta. Além disso, diversos outros tipos de análise em lote seriam possíveis, por exemplo, se o modelo de codificação adotado grava o horário de envase nas garrafas, poderia ser exigida a verificação de sequencia temporal nos códigos, etc.

Por conta dos diferentes modelos de codificação, sugere-se também a criação de uma interface gráfica que permita a seleção de uma receita, definindo qual sequência de tratamentos será utilizada para cada tipo de garrafa, e principalmente, para determinar o valor utilizado de *shutter speed*, dado que uma linha de produção pode operar em diferentes velocidades.

Outra sugestão para aumentar a acurácia do sistema, é confeccionar uma cabine fechada para o local onde é capturada a foto, assim sendo possível controlar melhor as condições de iluminação. Na ausência de tal aparato, um anteparo reflexivo de alumínio ou material com propriedade semelhante também pode ser utilizado, diminuindo a ocorrência de sombras.

REFERÊNCIAS

COATES, C.; JUVAN-BEAULIEU, I. How to choose between a Rolling Shutter & Global Shutter camera mode. **Miramar Communications Ltd**, 2020.

CUNHA, R. C. **Estudo Comparativo Sobre Ferramentas de Reconhecimento Óptico de Caracteres**. [s.l.] Instituto Federal de Minas Gerais, 2018.

GONG, L.; YU, M.; DUAN, W.; et al. A novel camera based approach for automatic expiry date detection and recognition on food packages. **IFIP Advances in Information and Communication Technology**, v. 519, p. 133–142, 2018.

GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing Third Edition Pearson. 2008.

GOOGLE. **Google Cloud Vision**. Disponível em: <https://cloud.google.com/vision/docs>. Acesso em: 24 jul. 2021a.

GOOGLE. **Matriz de Confusao**. Disponível em: <https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative>. Acesso em: 9 out. 2021b.

HOSOZAWA, K.; WIJAYA, R. H.; LINH, T. D.; et al. Recognition of Expiration Dates Written on Food Packages with Open Source OCR. **International Journal of Computer Theory and Engineering**, v. 10, n. 5, p. 170–174, 2018.

JAIDEDAI. **EasyOCR**. Disponível em: <https://www.jaided.ai/easyocr>. Acesso em: 7 ago. 2021.

MACHADO, U. T. **Inspeção Automática de Frascos Utilizando Visão Computacional em Sistema Embarcado**. [s.l.] Universidade de Caxias do Sul, 2019.

MOZILLA. **WebSockets**. Disponível em < https://developer.mozilla.org/pt-BR/docs/Web/API/WebSockets_API>. Acesos em: 24 jul. 2021.

OpenCV. Disponível em: <https://opencv.org/>. Acesso em: 7 ago. 2021.

PEDRINI, H.; SCHWARTZ, W. R. **Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações**. [s.l: s.n.].

SHARMA, S.; K., V. K.; GANDHI, R.; et al. Empty and Filled Bottle Inspection System. **ICTACT Journal on Image and Video Processing**, v. 06, n. 02, p. 1122–1126, 2015.

SILVA, A. J. DE J. Metodologia de ajuste de parâmetros para aquisição de imagens de boa qualidade utilizando uma câmera fotográfica semiprofissional comercial. **Revista Sítio Novo**, v. 4, n. 4, p. 202, 2020.

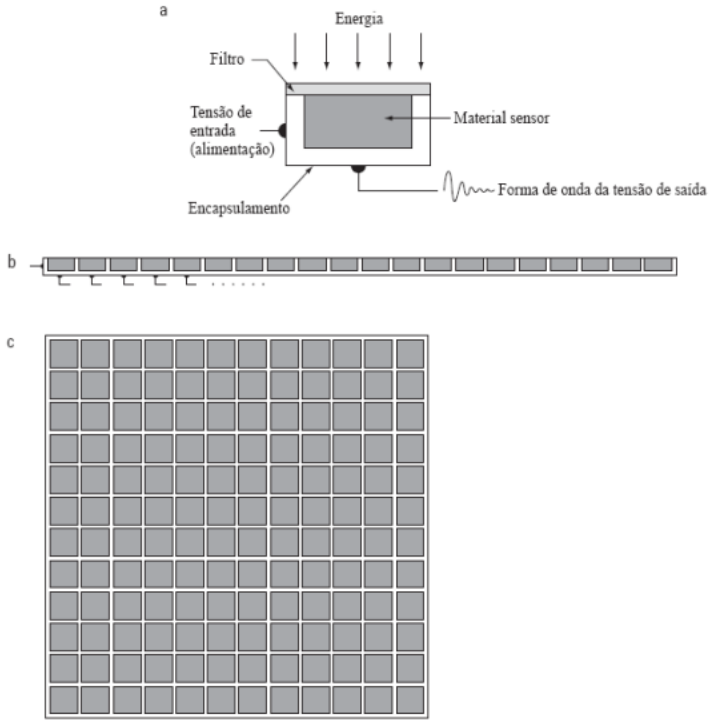
TESSERACT. **Tesseract**. Disponível em: <https://tesseract-ocr.github.io/tessdoc/>. Acesso em: 24 jul. 2021.

ZAAFOURI, A.; SAYADI, M.; FNAIECH, F. A vision approach for expiry date recognition using stretched gabor features. **International Arab Journal of Information Technology**, v. 12, n. 5, p. 448–455, 2015.

ZHOU, X.; YAO, C.; WEN, H.; et al. EAST: An efficient and accurate scene text detector. **Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017**, v. 2017- Janua, p. 2642–2651, 2017.

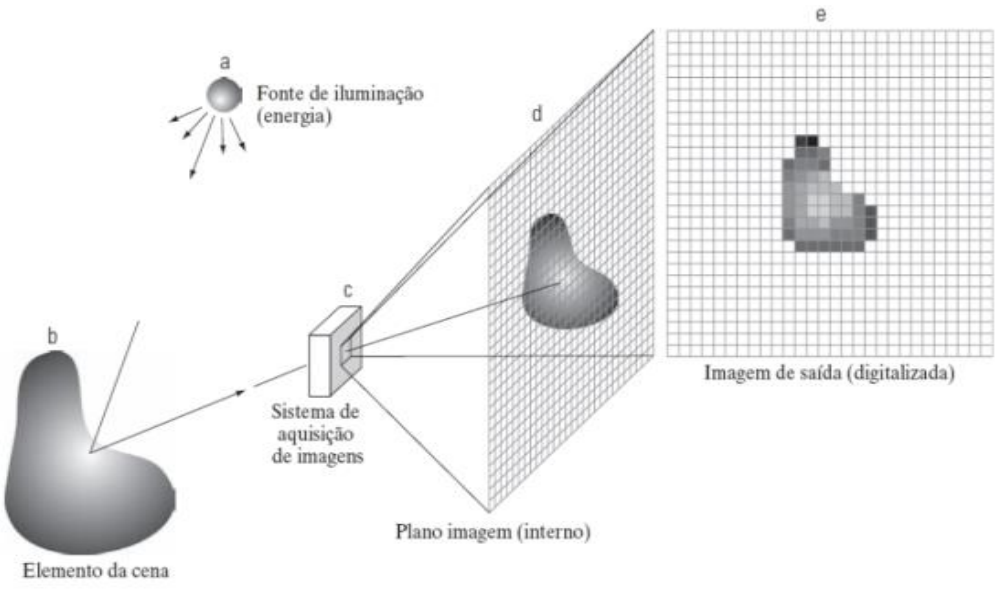
APÊNDICE A

Figura A.1: funcionamento de um arranjo de sensores. (a) Partes do sensor. (b) Uma linha do arranjo de sensores. (c) Matriz completa do arranjo de sensores.



Fonte: Adaptado de (Gonzalez e Woods, 2008)

Figura A.2: esquema do processo de aquisição de imagens digitais. (a) Fonte de energia luminosa. (b) Elemento da cena. (c) Sistema de aquisição de imagens. (d) Imagem no plano focal da lente. (e) Imagem digitalizada.



Fonte: Adaptado de (Gonzalez e Woods, 2008).

APÊNDICE B

Tabela B.1: especificações técnicas da Raspberry Pi Model 3B.

Especificações Raspberry Pi Model 3B	
Modelo da CPU	Quad Core Broadcom BCM2837 64bit
Frequência da CPU	1,2 GHz
Memória RAM	1 GB
Interfaces	40 portas digitais programáveis Interface para câmera (CSI) Conector de vídeo HDMI 4 portas USB 2.0
Conectividade	BCM43438 wireless LAN Bluetooth 4.2 BLE
Dimensões	85 x 56 x 17 mm

Tabela B.2: especificações técnicas da RaspBerry Pi Câmera V2.

Especificações Rasp Câmera V2	
Dimensão do conjunto	25 × 24 × 9 mm
Dimensão do sensor	3.68 x 2.76 mm
Resolução estática	8 Megapixels
Resolução do sensor	3280 × 2464 pixels
Tipo do sensor	Sony IMX219
Formatos de imagem	JPEG, DNG, BMP, PNG< YUV420, RGB888

Tabela B.3: especificações do sensor de IR utilizado.

Especificações Sensor IR	
Tensão de operação	3,3 a 5V DC
Distância de detecção	2 a 15cm
Dimensões	14mm x 31,1mm
Comparador	LM393

APÊNDICE C

Desenvolvimento dos algoritmos de processamento em tempo real

Após a definição do método mais adequado, seguiu-se para a etapa de desenvolvimento de uma rotina capaz de efetuar o processamento em tempo real, sem a necessidade de extrair as fotos manualmente. O sistema utiliza um algoritmo em Python que recebe o sinal de saída do sensor de IR e verifica a presença da garrafa. Caso haja detecção, é tirada uma foto, e a imagem é enviada para um computador conectado na rede local, utilizando *websockets*. Visando obter um processamento rápido da imagem, optou-se por utilizar um sistema baseado na arquitetura de cliente e servidor, onde o cliente (Raspberry) envia o arquivo da foto para o servidor (computador *desktop* na rede local). O servidor então reconhece os caracteres, avalia a conformidade com base nos critérios estabelecidos e envia o resultado obtido para a Raspberry. Também há possibilidade de salvar a imagem em disco.

Para o *desktop*, foi desenvolvido uma rotina que aguarda a conexão de um cliente, e após a conexão, aguarda o envio de um arquivo. Após receber o arquivo, efetuam-se os tratamentos na imagem e então, é enviada a resposta para o cliente. Após este processo, o código volta a esperar o envio de novas fotos, até que o processo seja interrompido. Para a Raspberry, a rotina desenvolvida começa inicializando a conexão com o servidor, e em caso de sucesso, esta fica em *stand-by* até que o sensor de IR seja disparado e a foto seja capturada. Após a captura, é realizado o envio do arquivo da foto, e então se aguarda a resposta do servidor. Quando a resposta chega, o sistema fica em *stand-by* novamente por 2 segundos, então a rotina se inicia novamente, até que o processo seja interrompido. O tempo de espera entre o envio de uma nova foto pode ser ajustado.