

RODRIGO ARAÚJO VIANNINI

POSTECH

IA PARA DEVS

COMPUTER VISION

# AULA 02

## SUMÁRIO

O QUE VEM POR AÍ? .....	3
HANDS ON .....	5
SAIBA MAIS.....	21
O QUE VOCÊ VIU NESTA AULA? .....	24
REFERÊNCIAS.....	25

EMSE

## O QUE VEM POR AÍ?

Nesta aula falaremos de OCR (Optical Character Recognition), que é uma tecnologia que permite a conversão de diferentes tipos de documentos, como arquivos PDF digitalizados, imagens capturadas por câmeras digitais ou até fotos de seu telefone, em dados editáveis e pesquisáveis. OCR é uma área da visão computacional e do processamento de linguagem natural que visa “ler” textos de imagens.

Utilizaremos duas tecnologias gratuitas: o “Tesseract”, para extrair informações de texto de imagens, e o “PaddleOCR”, o qual será utilizado para extrair textos de um PDF.

### Importância e aplicações

O OCR tem uma ampla gama de aplicações, incluindo:

- **Digitalização de documentos:** converte documentos em papel para formatos digitais, possibilitando o arquivamento e pesquisa.
- **Automação de processos:** extrai informações automaticamente de faturas, recibos e formulários.
- **Acessibilidade:** ajuda pessoas com deficiência visual a acessarem conteúdo textual em imagens.
- **Indexação e pesquisa:** facilita a busca em grandes volumes de texto digitalizado.

### História do OCR

- **Década de 1910:** Emanuel Goldberg desenvolveu uma máquina que lia caracteres e os converteu em telégrafos codificados.
- **Década de 1930:** Gustav Tauschek recebeu uma patente por seu método de reconhecimento de caracteres.
- **Década de 1950:** David H. Shepard desenvolveu o "Gismo", um dos primeiros dispositivos práticos de OCR.

- **1955:** a revista americana “Reader’s Digest” implementou OCR para leitura automatizada de texto.
- **Década de 1970:** o “Kurzweil Reading Machine”, inventado por Ray Kurzweil, podia reconhecer texto impresso em uma variedade de fontes.
- **Década de 1990:** a proliferação de computadores pessoais e scanners impulsionou o desenvolvimento de softwares de OCR, como o “Omnipage”.
- **Década de 2010:** com o desenvolvimento de redes neurais convolucionais (CNNs) o uso de deep learning começou a melhorar significativamente a precisão do OCR.
- **2018:** ferramentas de OCR open-source como Tesseract, aprimoradas com técnicas de deep learning, tornaram-se mais acessíveis e eficazes.

### **Evolução da tecnologia**

- **Deep learning:** o uso de redes neurais profundas, como as CNNs, tem melhorado significativamente a precisão do OCR.
- **GPU e TPU:** o desenvolvimento de hardware especializado, como GPUs e TPUs, permite a execução de modelos de OCR de forma mais rápida e eficiente.
- **Linguagens de programação e suas bibliotecas:** linguagens como Python e bibliotecas como OpenCV e Tesseract, facilitam a implementação e personalização de soluções de OCR.

## HANDS ON

Vamos implementar operações básicas de visão computacional utilizando a biblioteca OpenCV. Essas operações incluem carregamento, exibição, conversão para grayscale, redimensionamento, aplicação de filtros, detecção de bordas e desenho de formas em imagens. Essas operações são fundamentais para o desenvolvimento de aplicações mais complexas em visão computacional.

Mas antes, vamos falar um pouco mais sobre OCR.

### I. Componentes de um sistema de OCR

- **Aquisição de imagem:** é o processo de captura de uma imagem digital que será utilizada para OCR. Isso pode ser feito por meio de diversos dispositivos como scanners, câmeras digitais ou smartphones. A qualidade da imagem capturada é crucial para o sucesso do OCR, e deve garantir boa resolução, iluminação adequada e foco correto para que os caracteres sejam claramente distinguíveis.
- **Processamento de imagem:** refere-se a uma série de técnicas e operações aplicadas à imagem adquirida para melhorar sua qualidade e prepará-la para o reconhecimento de caracteres. Essas operações podem incluir a conversão para escala de cinza, filtragem e remoção de ruído, “thresholding” e binarização, além de outras técnicas como correção de inclinação (deskewing) e segmentação. O objetivo é otimizar a imagem para que os algoritmos de OCR possam reconhecer os caracteres de maneira precisa e eficiente.
- **Conversão para escala de cinza:** é o processo de transformar uma imagem colorida em uma imagem em tons de cinza. Cada pixel da imagem original é convertido em um valor de intensidade de cinza, variando do preto (0) ao branco (255). Isso simplifica o processamento da imagem, pois reduz a quantidade de dados e remove a influência das cores, mantendo apenas a informação de luminosidade.

- **Filtragem e remoção de ruído:** referem-se a técnicas utilizadas para melhorar a qualidade da imagem eliminando imperfeições e artefatos indesejados, como pontos, linhas e variações de intensidade que não fazem parte do conteúdo relevante. Métodos comuns incluem filtros de média, mediana e o filtro gaussiano, que suavizam a imagem e reduzem o impacto de pixels anômalos.
- **“Thresholding” e binarização:** são processos que convertem uma imagem em tons de cinza em uma imagem binária, onde cada pixel é representado como preto (0) ou branco (1). A binarização é realizada definindo um valor de limiar (threshold); pixels com intensidade acima desse limiar são convertidos em branco, enquanto os que se encontram abaixo desse limiar são convertidos em preto. Esse processo é crucial para destacar o texto ou elementos relevantes em uma imagem, facilitando a etapa de reconhecimento óptico de caracteres (OCR).
- **Reconhecimento de texto:** ou “reconhecimento óptico de caracteres” (OCR), é o processo de converter o texto de uma imagem digital em texto editável e pesquisável. Após a aquisição e o processamento da imagem, o OCR analisa os padrões de pixels que correspondem aos caracteres específicos. Utilizando algoritmos de correspondência de padrões, aprendizado de máquina e redes neurais, o OCR identifica e extrai os caracteres da imagem, transformando-os em dados textuais digitais que podem ser manipulados, armazenados e pesquisados. Este processo é fundamental para a digitalização de documentos impressos, permitindo a automatização de tarefas que envolvem grandes volumes de texto.

## II. Conceitos básicos de imagem digital

- **Pixels (abreviação de "picture elements"):** são os menores componentes de uma imagem digital. Cada pixel representa um único ponto de cor e intensidade na imagem. Em uma imagem colorida, cada pixel geralmente é composto de valores de vermelho, verde e azul (RGB) que combinados determinam a cor final do pixel. Em imagens em escala de cinza, cada pixel possui um valor de intensidade que varia do preto ao branco.

- **Resolução:** refere-se à quantidade de detalhes que uma imagem digital pode representar e é geralmente expressa pelo número de pixels na largura e altura da imagem (por exemplo, 1920x1080 pixels). Resoluções mais altas significam que a imagem contém mais pixels, permitindo maior detalhamento e nitidez. Em contextos de impressão, a resolução também pode ser medida em DPI (pontos por polegada), que indica quantos pontos de tinta são aplicados por polegada quadrada de papel.
- **Espaços de cores:** são modelos matemáticos que descrevem a forma como as cores são representadas em uma imagem digital. Eles definem um conjunto de cores que pode ser usado e as relações entre essas cores. Exemplos comuns de espaços de cores incluem:
  - **RGB (Red, Green, Blue):** utilizado principalmente para imagens em telas eletrônicas, onde as cores são criadas pela combinação de diferentes intensidades de vermelho, verde e azul.
  - **CMYK (Cyan, Magenta, Yellow, Key/Black):** utilizado principalmente em impressão, onde as cores são criadas pela combinação de diferentes quantidades de ciano, magenta, amarelo e preto.
  - **HSV (Hue, Saturation, Value):** um espaço de cores que separa a cor em matiz (tipo de cor), saturação (intensidade da cor) e valor (brilho), frequentemente usado em aplicações de edição de imagem.

### III. Representação de imagens

- **Matrizes:** em imagens digitais, uma matriz é uma estrutura bidimensional de dados onde cada elemento representa um pixel. Para uma imagem em escala de cinza, a matriz contém valores de intensidade de cinza para cada pixel, variando de 0 (preto) a 255 (branco). Para uma imagem colorida, há três matrizes (ou canais) separadas para os componentes de cor vermelho, verde e azul (RGB), cada uma com valores que variam de 0 a 255.
- **Canais RGB:** as imagens coloridas são frequentemente representadas pelo modelo RGB, onde cada pixel é composto de três valores que correspondem aos canais vermelho (R), verde (G) e azul (B). Cada canal é

uma matriz que representa a intensidade da respectiva cor para cada pixel na imagem. A combinação dos valores dos três canais determina a cor final de cada pixel. Por exemplo, um pixel com valores (255, 0, 0) seria vermelho puro.

- **Grayscale:** imagens em escala de cinza (grayscale) são imagens onde cada pixel tem um único valor de intensidade que representa a quantidade de luz, variando de preto (0) a branco (255). Ao contrário das imagens RGB, que possuem três canais, as imagens em grayscale possuem apenas um canal. Essas imagens são úteis em muitas aplicações de processamento de imagem e visão computacional, pois simplificam o processamento ao reduzir a quantidade de dados necessária.
- **CMYK (Cyan, Magenta, Yellow, Key/Black):** utilizado principalmente em impressão, o modelo de cores CMYK representa as imagens por meio de quatro canais: ciano, magenta, amarelo e preto. Cada canal indica a quantidade de cada tinta necessária para produzir a cor final em uma impressora.
- **HSI/HSV/HSL (Hue, Saturation, Intensity/Value/Lightness):** esses modelos de cores representam imagens em termos de matiz (hue), saturação e intensidade/valor/luminosidade. Esses espaços de cores são úteis em aplicações de processamento de imagem onde a manipulação direta da tonalidade e da intensidade das cores é desejada.
- **YUV/YIQ/YCbCr:** utilizados em sistemas de vídeo e compressão de imagens, esses modelos de cores separam a informação de luminosidade (Y), da informação de cor (UV, IQ, CbCr). Isso é útil para compressão de vídeo, pois a sensibilidade humana à luminosidade é maior do que à cor, permitindo uma compressão mais eficiente.
- **Lab (CIELAB):** o espaço de cores Lab é baseado na percepção humana da cor e é projetado para ser perceptualmente uniforme, significando que mudanças iguais em valores de cor correspondem a mudanças iguais na percepção visual. Ele divide a cor em três componentes:  $L^*$  (luminosidade),  $a^*$  (variação entre verde e vermelho), e  $b^*$  (variação entre azul e amarelo).



- **Representação de imagem em vetores:** diferente das representações matriciais, as imagens vetoriais utilizam formas geométricas como pontos, linhas, curvas e polígonos para representar a imagem. Isso permite que as imagens sejam escaladas para qualquer tamanho sem perda de qualidade. Formatos comuns de imagens vetoriais incluem SVG (Scalable Vector Graphics) e PDF.
- **Formatos comprimidos:** imagens também podem ser representadas em formatos comprimidos que reduzem o tamanho do arquivo. Os exemplos incluem:
  - **JPEG:** usa compressão com perdas, o que reduz o tamanho do arquivo, mas pode introduzir artefatos visuais.
  - **PNG:** usa compressão sem perdas, preservando a qualidade da imagem.
  - **GIF:** suporta animações e usa uma paleta de cores limitada, adequada para gráficos simples.
- **Representação em bases de ondas (“wavelets”):** utilizada em algumas técnicas avançadas de compressão e processamento de imagem, a transformação wavelet decompõe a imagem em componentes de frequência que podem ser processados separadamente para várias aplicações, incluindo compressão e reconhecimento de padrões.

#### IV. Ferramentas que serão utilizadas

- **Tesseract:** é uma ferramenta de OCR open-source desenvolvida pelo Google. É amplamente utilizada por sua precisão e suporte a várias línguas. Tesseract é um motor de OCR (Reconhecimento Óptico de Caracteres) de código aberto desenvolvido inicialmente pela HP e mantido pela Google. Ele é amplamente utilizado para converter texto impresso em texto digital editável. O Tesseract suporta mais de 100 idiomas e possui várias características avançadas, como:
  - **Treinamento personalizado:** permite treinar o motor com fontes e caracteres personalizados.
  - **Suporte a diferentes formatos de entrada:** pode processar imagens em formatos como TIFF, PNG, JPEG e PDF.

- **Reconhecimento de layout:** capaz de identificar e preservar o layout de documentos complexos, incluindo colunas e tabelas.
- **Integração com outras ferramentas:** pode ser combinado com outras bibliotecas de processamento de imagem, como OpenCV, para pré-processamento avançado.
- **PaddleOCR:** é uma biblioteca de OCR de código aberto, desenvolvida pela Baidu, e é parte do framework “PaddlePaddle” para aprendizado profundo. Ele é projetado para ser altamente eficiente e preciso, suportando uma ampla gama de funcionalidades:
  - **Modelos de alta precisão:** utiliza modelos baseados em aprendizado profundo que são treinados em grandes conjuntos de dados para alcançar alta precisão no reconhecimento de texto.
  - **Suporte multi-idíomas:** inclui suporte para mais de 80 idiomas, com modelos pré-treinados disponíveis para muitos deles.
  - **Detecção e reconhecimento:** capaz de detectar regiões de texto e reconhecê-lo nessas regiões separadamente, o que melhora a precisão em documentos com layouts complexos.
  - **Implementação flexível:** pode ser utilizado em diversos ambientes, desde dispositivos móveis até servidores de alto desempenho.
  - **Documentação e comunidade:** possui boa documentação e uma comunidade ativa, facilitando a integração e o desenvolvimento de soluções personalizadas.
- **Google OCR (Google Cloud Vision OCR):** Google Cloud Vision OCR é um serviço de reconhecimento óptico de caracteres fornecido pelo Google Cloud Platform. Ele oferece uma variedade de funcionalidades para a extração de texto de imagens e documentos:
  - **Detecção de texto em imagens:** pode extrair texto de imagens e documentos em diversos idiomas, incluindo textos manuscritos.
  - **Reconhecimento de layout:** capaz de identificar e manter o layout de documentos, incluindo colunas, tabelas e parágrafos.
  - **APIs RESTful:** permite fácil integração com aplicações por meio de APIs RESTful, facilitando o desenvolvimento de soluções personalizadas.

- **Suporte a múltiplos formatos de entrada:** aceita vários formatos de imagem como JPEG, PNG e PDF.
- **Machine learning e IA:** utiliza algoritmos avançados de aprendizado de máquina para aumentar a precisão do OCR.
- **Azure OCR (Azure Cognitive Services OCR):** Azure Cognitive Services OCR é parte dos serviços cognitivos da Microsoft Azure. Ele oferece funcionalidades robustas para reconhecimento de texto em imagens e documentos:
  - **Extração de texto de imagens:** suporta a extração de texto impresso e manuscrito de imagens e documentos em diversos idiomas.
  - **Reconhecimento de layout:** pode detectar e preservar o layout de documentos, incluindo formatação de texto e estruturas de tabelas.
  - **APIs RESTful:** facilita a integração com outras aplicações e serviços por meio de APIs RESTful.
  - **Suporte a diversos formatos de imagem:** compatível com vários formatos de entrada, como JPEG, PNG, BMP e PDF.
  - **Escalabilidade e desempenho:** projetado para ser altamente escalável e eficiente, adequado para aplicações empresariais de grande escala.
- **AWS OCR (Amazon Textract):** Amazon Textract é um serviço de OCR da Amazon Web Services (AWS) que vai além do reconhecimento de texto básico, oferecendo funcionalidades avançadas para análise de documentos:
  - **Extração de texto, dados e tabelas:** capaz de extrair texto, além de informações estruturadas de tabelas e formulários.
  - **Reconhecimento de texto manuscrito e impresso:** suporta a extração de texto de documentos impressos e manuscritos.
  - **APIs RESTful:** permite fácil integração com outras aplicações e serviços via APIs RESTful.
  - **Suporte a múltiplos formatos de entrada:** compatível com vários formatos de imagem como JPEG, PNG, TIFF e PDF.
  - **Segurança e conformidade:** integrado com serviços de segurança e conformidade da AWS, garantindo a proteção e privacidade dos dados.

Vamos codar...

## Instalação das bibliotecas necessárias

Para começar, vamos instalar as dependências necessárias.

OpenCV:

```
1 !pip install opencv-python
```

Figura 1 - Instalação das bibliotecas necessárias (1)  
Fonte: Elaborado pelo autor (2024)

Tesseract:

```
1 !pip install pytesseract
```

Figura 2 - Instalação das bibliotecas necessárias (2)  
Fonte: Elaborado pelo autor (2024)

```
1 !apt-get install -y tesseract-ocr
2 !apt-get install -y libtesseract-dev
3 !apt-get install -y tesseract-ocr-por
```

Figura 3 - Instalação das bibliotecas necessárias (3)  
Fonte: Elaborado pelo autor (2024)

## Carregamento e exibição de imagens

Vamos iniciar carregando uma imagem do disco e exibindo-a na tela.

Veja um exemplo usando IDEs (PyCharm ou VsCode )

```
1 import cv2
2 import matplotlib.pyplot as plt
3
4 # Carregar imagem
5 imagem = cv2.imread('caminho/para/sua/imagem.jpg')
6
7 # Exibir imagem
8 cv2.imshow('Imagem Carregada', imagem)
9 cv2.waitKey(0)
10 cv2.destroyAllWindows()
11
```

Figura 4 – Carregamento e exibição de imagens (1)  
Fonte: Elaborado pelo autor (2024)

Usando o Google Colab e carregando a imagem com o botão de upload:

```

1 from google.colab import files
2 from google.colab.patches import cv2_imshow
3 import cv2
4
5 # Upload de arquivo
6 uploaded = files.upload()
7
8 # Carregar imagem
9 imagem = cv2.imread(list(uploaded.keys())[0])
10
11 # Exibir imagem
12 cv2_imshow(imagem)
13

```

Figura 5 – Carregando a imagem com o botão de upload  
Fonte: Elaborado pelo autor (2024)

Carregando imagem sem botão de upload:

```

1 import cv2
2 from google.colab.patches import cv2_imshow
3
4 # Carregando a imagem
5 img = cv2.imread('/content/texto.png')
6
7 # Exibir imagem
8 cv2_imshow(img)
9

```

Figura 5 – Carregando a imagem sem o botão de upload  
Fonte: Elaborado pelo autor (2024)

Output:



Figura 6 – Output  
Fonte: Elaborado pelo autor (2024)

OCR:

```

1 # Aplicando o Tesseract OCR
2 texto = pytesseract.image_to_string(img, lang='por')
3 print(texto)
4

```

Figura 7 – OCR  
Fonte: Elaborado pelo autor (2024)

Output OCR:

```

Vencer é uma mistura de
luta, esforço, otimismo
e não desistir nunca.
Amby Burfoot
aa

```

Figura 8 – Output OCR  
Fonte: Elaborado pelo autor (2024)

## Conversão de imagem para grayscale

Vamos converter a imagem colorida para escala de cinza, uma operação comum em visão computacional.

```

1 # Converter para grayscale
2 imagem_gray = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
3
4 # Exibir imagem grayscale
5 cv2.imshow(imagem_gray)
6

```

Figura 9 – Conversão da imagem para escala de cinza  
Fonte: Elaborado pelo autor (2024)

Output:



Figura 10 – Output (2)  
Fonte: Elaborado pelo autor (2024)

OCR:

```

1 # Aplicando o Tesseract OCR
2 texto = pytesseract.image_to_string(imagem_gray, lang='por')
3 print(texto)
4

```

Figura 11 – OCR (2)  
Fonte: Elaborado pelo autor (2024)

Output OCR:



Figura 12 – Output OCR (2)  
Fonte: Elaborado pelo autor (2024)

## Redimensionamento de imagem

Vamos redimensionar uma imagem para padronizar o tamanho das figuras de entrada.

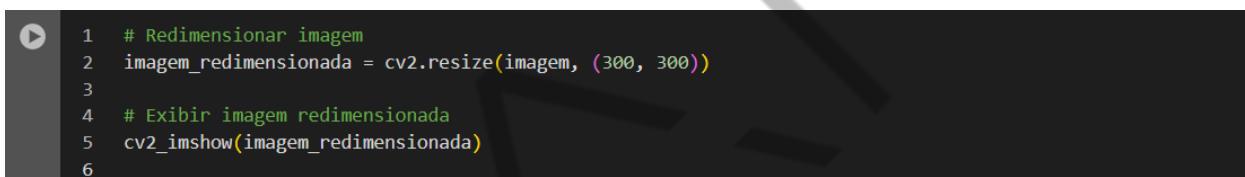


Figura 13 – Redimensionamento de imagens  
Fonte: Elaborado pelo autor (2024)

Output:

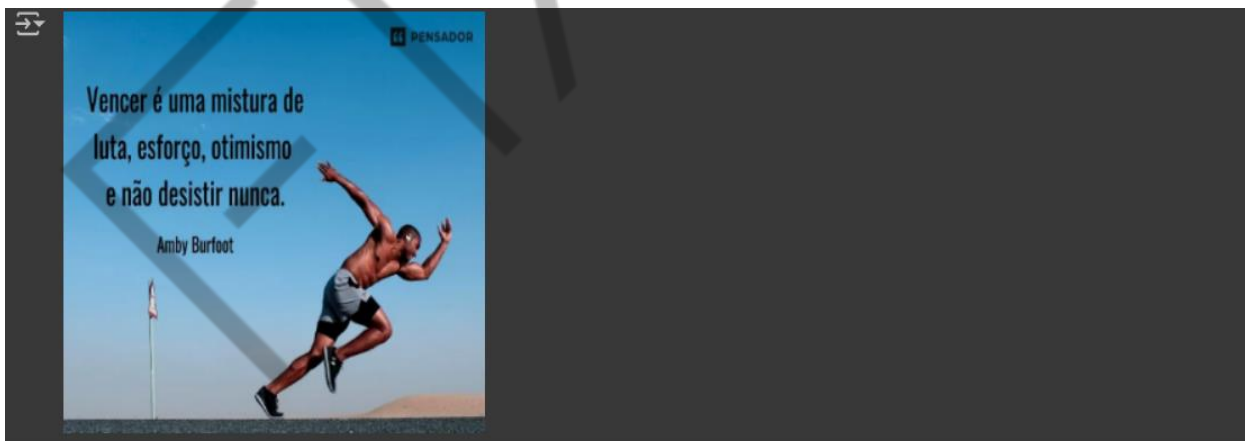


Figura 14 – Output (3)  
Fonte: Elaborado pelo autor (2024)

OCR:

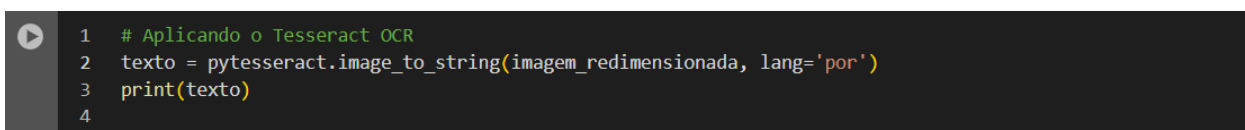


Figura 15 – OCR (3)  
Fonte: Elaborado pelo autor (2024)

Output OCR:

```

Vencer é uma mistura de
luta, esforço, otimismo
e não desistir nunca.

Amby Burtoot
f

```

Figura 16 – Output OCR (4)  
Fonte: Elaborado pelo autor (2024)

## Aplicação de filtros para suavização de imagens

Vamos aplicar a suavização de uma imagem para reduzir o ruído e melhorar a detecção de bordas.

```

1 # Aplicar filtro de suavização (blur)
2 imagem_suavizada = cv2.GaussianBlur(imagem, (15, 15), 0)
3
4 # Exibir imagem suavizada
5 cv2.imshow(imagem_suavizada)
6

```

Figura 17 – Suavização de imagem por meio da aplicação de filtros  
Fonte: Elaborado pelo autor (2024)

Output:

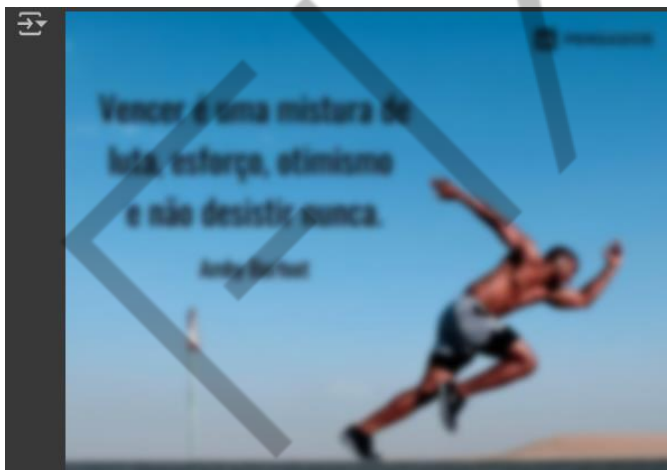


Figura 17 – Output (4)  
Fonte: Elaborado pelo autor (2024)

OCR:

```

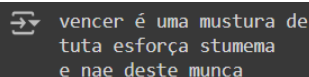
1 # Aplicando o Tesseract OCR
2 texto = pytesseract.image_to_string(imagem_suavizada, lang='por')
3 print(texto)
4

```

Figura 17 – OCR (4)  
Fonte: Elaborado pelo autor (2024)

Output OCR:



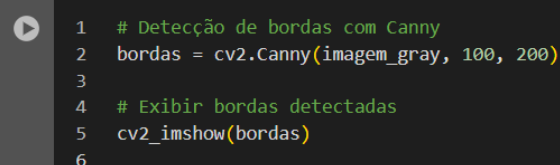


```
vencer é uma mistura de
luta, esforço, otimismo
e não desistir nunca
```

Figura 17 – Output OCR (5)  
Fonte: Elaborado pelo autor (2024)

## Detecção de bordas com Canny

Vamos iniciar a detecção de bordas, um passo crucial em muitos algoritmos de visão computacional, como detecção de objetos.



```
1 # Detecção de bordas com Canny
2 bordas = cv2.Canny(imagem_gray, 100, 200)
3
4 # Exibir bordas detectadas
5 cv2.imshow(bordas)
6
```

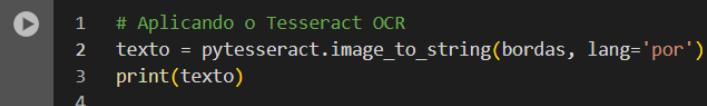
Figura 18 – Detecção de bordas com Canny  
Fonte: Elaborado pelo autor (2024)

Output:



Figura 18 – Output (5)  
Fonte: Elaborado pelo autor (2024)

OCR:



```
1 # Aplicando o Tesseract OCR
2 texto = pytesseract.image_to_string(bordas, lang='por')
3 print(texto)
4
```

Figura 19 – OCR (5)  
Fonte: Elaborado pelo autor (2024)

Output OCR:

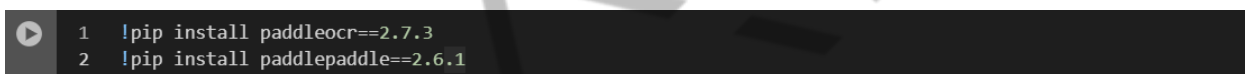


```
EE
need Ros
luta, esforço, otimismo
Ss ' a
[en oe e =.
A
as
Prod > dA
```

Figura 20 – Output OCR (6)  
Fonte: Elaborado pelo autor (2024)

Para cada tipo de imagem, pode ou não ser necessário a utilização de algum processamento. Em alguns casos, podemos até combinar mais de uma técnica. O importante é testar e encontrar a melhor opção para atender as necessidades das suas amostras. As opções acima, são apenas alguns exemplos e poderão explorar novas formas de tratamento à medida que os desafios se apresentam.

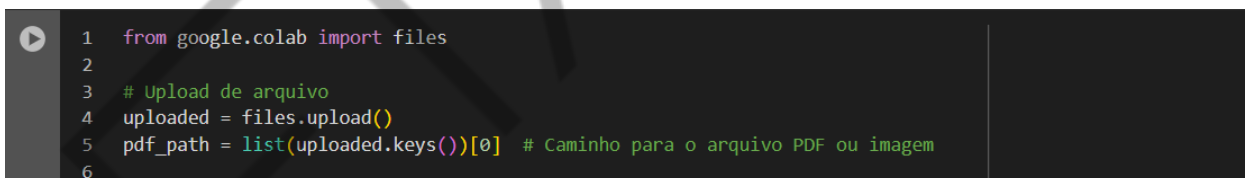
### Paddle OCR:



```
1 !pip install paddleocr==2.7.3
2 !pip install paddlepaddle==2.6.1
```

Figura 21 – Paddle OCR  
Fonte: Elaborado pelo autor (2024)

Carregando PDF com botão de upload:



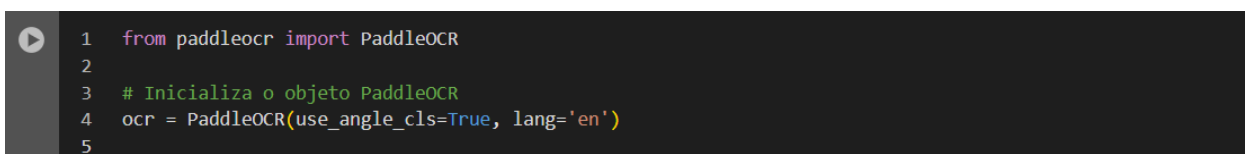
```
1 from google.colab import files
2
3 # Upload de arquivo
4 uploaded = files.upload()
5 pdf_path = list(uploaded.keys())[0] # Caminho para o arquivo PDF ou imagem
6
```

Figura 22 – Carregando PDF com botão de upload  
Fonte: Elaborado pelo autor (2024)

### Extração de texto com PaddleOCR

Vamos inicializar o objeto PaddleOCR e criar uma função para extrair texto de uma imagem ou PDF.

Inicializando o PaddleOCR:



```
1 from paddleocr import PaddleOCR
2
3 # Inicializa o objeto PaddleOCR
4 ocr = PaddleOCR(use_angle_cls=True, lang='en')
5
```

Figura 22 – Iniciando o PaddleOCR  
Fonte: Elaborado pelo autor (2024)

Função para extrair texto:

```

1 # Função para extrair texto OCR de um PDF ou imagem
2 def extract_text_from_pdf(pdf_path):
3     # Use o método `ocr.ocr()` para extrair texto de uma imagem
4     extract_ocr = ocr.ocr(pdf_path)
5
6     # Concatene todas as palavras detectadas em uma única string
7     result = ''
8     for line in extract_ocr:
9         for word in line:
10             result += word[1][0] + ' ' # O índice 0 representa o texto da palavra
11             result += '\n' # Adicione uma nova linha após cada linha de texto
12     return result
13
14 # Extraí texto do PDF ou imagem usando OCR
15 ocr_text = extract_text_from_pdf(pdf_path)
16
17 # Imprimir o texto extraído
18 print("Texto Extraído:")
19 print(ocr_text)
20

```

Figura 22 – Função Extração de Texto  
Fonte: Elaborado pelo autor (2024)

Output:

```

'enei mpla Energia e Servicos S. A 01.Sala 701. Aqwa Corporal DOCUMENTO AUXILIAR DA NOTA FISCAL DE ENERGIA ELETTRICA ELETTRONICA TIPO DE FORNECIME
NTO NDE DIAS EITURA MONOFASICO 26/03/2024 26/04/2024 31 27/05/2024 Bruno Eliseo Alcantara DANSTALACAO/ Av Bosque da Fazenda, 141 840644 Leblon -
Rio de Janeiro CEP:24350-000 CPF/ CNPJ: 112.343.433-01 NDO CLIENTE 5942541 59:13-03:00 MES/AN VENCIMENT TOTAL A PAGAF 04/2024 10/05/2024 R$264,4
8 DESCRICAO DO FATURAMENTO RBUTOSCAQUOTALR CONSUMO/kw Alouota ICMS Tnit (es) ISIPASE 3.2 TIPOS DE Kh213 0,29318 CPPRMUN 565588858 Subtotal Fatro
n 249,30 264,46 8,07 246,94 MENTOS DE MEI NMedidolt-614 P.Heraricl DataLeitor Loitura Leitura o Ajuste Sinief 01/2019 DADOS DE MEDICAO REAVISO D
E CONTAS VENCIDAS Grandezas ENEL 838300000020 644800193008 007515199076 000594254132 CPF: Bruno Eliseo Alcantara 112.343.433-01 29/04/2024 02024
04060526333 Referencia 05/2024 10/05/2024 N de controle Mensagem: 300007515199 Pague via PIX! Utilize este QR Code \n'

```

Figura 23 – Output (6)  
Fonte: Elaborado pelo autor (2024)

## Extração de informações específicas com expressões regulares

Vamos usar expressões regulares para extrair informações específicas, como CPF/CNPJ, nome do cliente, número do cliente, data de vencimento, valor total, referência de faturamento, e verificar se há um QR Code presente.

Definindo padrões de expressões regulares:

```

1 import re
2
3 # Expressões regulares para encontrar padrões específicos
4 padrao_cpf_cnpj = r"CPF/ CNPJ:\s{1}([0-9-])+"
5 padrao_nome_cliente = r"CPF:\s{1}([0-9-])+\s{1}([0-9-])+\s{1}([0-9-])+\s{1}([0-9-])+"
6 padrao_num_cliente = r"NDO CLIENTE\s{1}([0-9-])+"
7 padrao_data_leitura = r"DataLeitor\s{1}([0-9-])+\s{1}([0-9-])+\s{1}([0-9-])+"
8 padrao_data_vencimento = r"MES/AN\s{1}([0-9-])+\s{1}([0-9-])+\s{1}([0-9-])+\s{1}([0-9-])+"
9 padrao_valor_total = r"R$\s{1}([0-9-])+\s{1}([0-9-])+"
10 padrao_descricao_faturamento = r"DESCRICAO DO FATURAMENTO(.*)MENTOS DE MEI"
11 padrao_referencia_faturamento = r"Referencia\s{1}([0-9-])+\s{1}([0-9-])+\s{1}([0-9-])+\s{1}([0-9-])+"
12 padrao_qr_code = r"Pague via PIX! Utilize este QR Code"
13

```

Figura 23 – Definindo padrões de expressões regulares  
Fonte: Elaborado pelo autor (2024)

Função para extrair informações com base nos padrões:

```

1  # Função para extrair informações com base nos padrões de expressão regular
2  def extrair_informacoes(texto, padrao):
3      match = re.search(padrao, texto)
4      if match:
5          return match.group(1)
6      else:
7          return None
8
9  # Extrair informações
10 cpf_cnpj = extrair_informacoes(ocr_text, padrao_cpf_cnpj)
11 nome_cliente = extrair_informacoes(ocr_text, padrao_nome_cliente)
12 num_cliente = extrair_informacoes(ocr_text, padrao_num_cliente)
13 data_vencimento = extrair_informacoes(ocr_text, padrao_data_vencimento)
14 valor_total = extrair_informacoes(ocr_text, padrao_valor_total)
15 referencia_faturamento = extrair_informacoes(ocr_text, padrao_referencia_faturamento)
16
17 # Verificar se há QR Code presente
18 qr_code_presente = re.search(padrao_qr_code, ocr_text) is not None
19
20 # Exibir informações extraídas
21 print("Nome do Cliente:", nome_cliente)
22 print("CPF/CNPJ:", cpf_cnpj)
23 print("Número do Cliente:", num_cliente)
24 print("Data de Vencimento:", data_vencimento)
25 print("Valor Total:", valor_total)
26 print("Referência de Faturamento:", referencia_faturamento)
27 print("QR Code Presente:", qr_code_presente)
28

```

Figura 24 – Função para extrair informações com base nos padrões

Fonte: Elaborado pelo autor (2024)

Output:

```

➡ Nome do Cliente: Bruno Eliseo Alcantara
  CPF/CNPJ: 112.343.433-01
  Número do Cliente: 5942541
  Data de Vencimento: 04/2024 10/05/2024
  Valor Total: 264,48
  Referência de Faturamento: 05/2024 10/05/2024
  QR Code Presente: True

```

Figura 25 – Output (7)

Fonte: Elaborado pelo autor (2024)

## SAIBA MAIS

À medida que nos aprofundamos na introdução ao OCR (Reconhecimento Óptico de Caracteres), é importante entender que essa técnica abrange uma variedade de métodos e aplicações que vão muito além da simples extração de texto de imagens. Aqui estão algumas áreas de interesse adicionais que você pode explorar:

- **Reconhecimento de padrões:** o reconhecimento de padrões em OCR se concentra na identificação e interpretação de padrões de texto em imagens. Isso é fundamental para aplicações como leitura automática de documentos, identificação de placas de veículos e processamento de formulários.
- **Segmentação de imagem:** a segmentação de imagem divide uma imagem em regiões distintas, essenciais para isolar texto de fundos complexos, facilitando a leitura precisa em condições variadas, como documentos escaneados ou fotos de textos.
- **Visão estéreo e reconstrução 3D:** embora não diretamente ligada ao OCR, a visão estéreo e a reconstrução 3D podem complementar técnicas de OCR, especialmente em aplicações onde a compreensão da estrutura tridimensional do ambiente é necessária, como na leitura de textos em superfícies curvas.
- **Aprendizado profundo em OCR:** o aprendizado profundo, especialmente as redes neurais convolucionais (CNNs), tem revolucionado o OCR, permitindo melhorias significativas em precisão e velocidade. Modelos de aprendizado profundo são amplamente utilizados para reconhecimento de caracteres manuscritos e impressão em diversos idiomas.
- **Realidade aumentada:** o OCR desempenha um papel crucial na realidade aumentada, permitindo a tradução em tempo real de texto em ambientes do mundo real, como placas de sinalização, livros e menus, por meio de dispositivos móveis ou óculos inteligentes.

- **Aplicações em medicina e saúde:** na área da saúde, o OCR é utilizado para digitalizar prontuários médicos, interpretar prescrições manuscritas e extrair informações de exames médicos, aumentando a eficiência e reduzindo erros administrativos.
- **Visão computacional embarcada:** a visão computacional embarcada com OCR é comum em dispositivos como scanners portáteis, leitores de código de barras e sistemas de controle de inventário, permitindo a leitura automática de texto e códigos em tempo real.
- **Reconhecimento de texto em imagens:** o OCR é amplamente usado para converter documentos impressos, manuscritos ou digitados em dados digitais editáveis, facilitando a digitalização de grandes volumes de documentos, livros e arquivos históricos.
- **Análise de movimento e pose:** embora focada em movimento humano, a análise de movimento e pose pode complementar o OCR em sistemas de leitura automática, como em caixas registradoras, onde a movimentação do produto é analisada para otimizar a leitura do código de barras ou rótulo.
- **Segmentação semântica:** a segmentação semântica pode ser usada em OCR para identificar e categorizar diferentes tipos de texto em um documento, como títulos, parágrafos, rodapés e tabelas, melhorando a precisão na digitalização de documentos complexos.
- **Visão computacional para agricultura:** o OCR pode ser aplicado na agricultura para ler e interpretar etiquetas e códigos em equipamentos agrícolas, monitorar o uso de insumos e rastrear a origem e movimentação de produtos agrícolas.
- **Identificação de emoções em faces:** a identificação de emoções em faces pode complementar sistemas de OCR em interfaces de usuário, ajustando a leitura e a interpretação de textos baseadas no reconhecimento das emoções do usuário, melhorando a experiência de interação.
- **Fusão de dados sensoriais:** a fusão de dados sensoriais combina informações de diferentes sensores para melhorar a precisão do OCR, especialmente em ambientes complexos e onde múltiplos sensores são

utilizados para captar texto de diferentes superfícies e condições de iluminação.

- **Detecção de anomalias e eventos raros:** o OCR pode ser usado para detectar anomalias em documentos, como textos falsificados ou adulterados, e identificar eventos raros em grandes volumes de dados textuais, como padrões de fraude em documentos financeiros.
- **Interpretação de imagens médicas:** na medicina, o OCR pode ser usado para digitalizar e interpretar textos em imagens médicas, como anotações em radiografias e exames, auxiliando na integração de dados textuais com registros eletrônicos de saúde.
- **Simulação e realidade virtual:** o OCR pode ser utilizado em simulações e realidade virtual para interpretar e gerar textos dinâmicos dentro de ambientes virtuais, como legendas automáticas em treinamentos e jogos interativos.

Esses são apenas alguns exemplos das vastas possibilidades dentro do campo do OCR. Ao explorar esses tópicos, você poderá descobrir novas aplicações e contribuir para o avanço contínuo desta área de pesquisa e desenvolvimento.

## O QUE VOCÊ VIU NESTA AULA?

Nesta aula, conhecemos os fundamentos do OCR (Reconhecimento Óptico de Caracteres), explorando desde sua definição até as suas aplicações práticas. O OCR é uma técnica que permite a extração automática de texto de imagens e documentos digitalizados, transformando-os em dados editáveis e pesquisáveis. Durante nossa exploração, abordamos duas importantes bibliotecas para OCR: Tesseract e PaddleOCR.

EXEMPLO



## REFERÊNCIAS

BISHOP, C. M. **Pattern Recognition and Machine Learning**. Estados Unidos: Springer, 2006.

BUNKE, H.; W., P. S. P. **Handbook of Character Recognition and Document Image Analysis**. Singapura: World Scientific, 1997.

NARTKER, T. A.; NAGY, G.; RICE, S. V. **OCR Technology for Document Recognition and Data Capture**. Califórnia: SPIE Press, 2009.

NARTKER, T. A.; NAGY, G.; RICE, S. V. **Optical Character Recognition: An Illustrated Guide to the Frontier**. Estados Unidos: Springer, 1999.

## **PALAVRAS-CHAVE**

OCR. Reconhecimento de Caracteres. Processamento de Documentos. Aprendizado de Máquina.

EXEMPLO



# POSTECH