

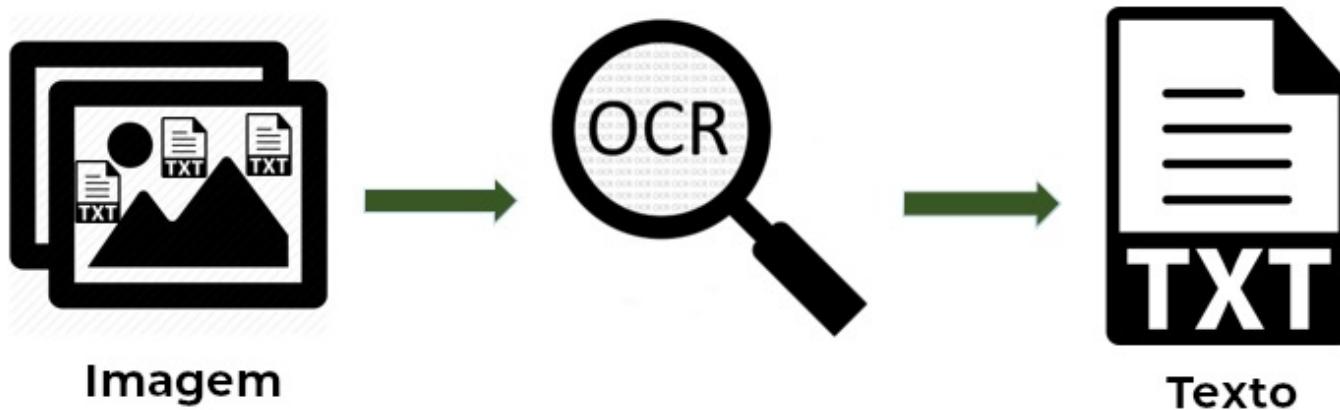
RECONHECIMENTO DE TEXTOS COM OCR E PYTHON



CONTEÚDO DO CURSO

- OCR com Tesseract
- Técnicas para pré-processamento das imagens
- OCR com EAST
- Treinamento de um OCR do zero
 - ANEXOS: Redes Neurais Artificiais e Redes Neurais Convolucionais
- EasyOCR para cenários naturais
- OCR em vídeos
- Projetos
 - Busca por termos similares (leitura de diretórios)
 - Scanner de documento + OCR
 - Leitura de placas de carros

OCR – OPTICAL CHARACTER RECOGNITION



VANTAGENS

- Otimização de tempo
- Automação
- Facilidade nas buscas
- Acessibilidade

DETECÇÃO DE TEXTOS – CENÁRIOS CONTROLADOS



Fonte: <https://www.pyimagesearch.com/2017/07/17/credit-card-ocr-with-opencv-and-python/>

DETECÇÃO DE TEXTOS – CENÁRIOS CONTROLADOS



DETECÇÃO DE TEXTOS – CENÁRIOS NATURAIS



Fonte: [Mancas-Thillou e Gosselin](#)

- Condições de iluminação
- Desfoque/borrão
- Resolução
- Ângulo de visão
- Objetos não planos
- Objetos que não são de papel
- Ruído na imagem
- Layout desconhecido
- Texto inclinado
- Problemas com letras diferentes

O que fazer nesses casos?



Fonte: <https://www.rsipvision.com/real-time-ocr/>

TESSERACT

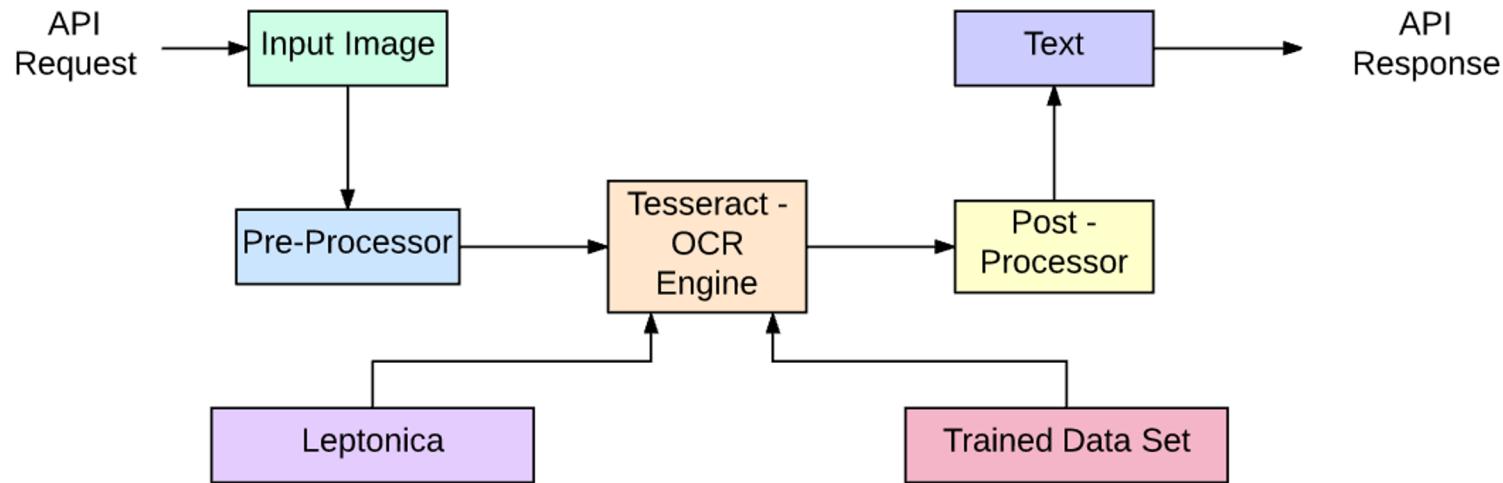
- É uma **engine** de OCR (*Optical Character Recognition* - Reconhecimento ótico de caracteres)
- Originalmente começou como um projeto de doutorado nos laboratórios da Hewlett-Packard (HP)
- Após ganhar popularidade, passou a ser desenvolvida pela HP em 1984. A equipe trabalhou até 1994
- Foi portado para Windows em 1996
- Em 2005 foi liberado na comunidade como um projeto **open source**
- Em 2006 o seu desenvolvimento começou a ser patrocinado pelo Google
- Desde 2006, é considerado até hoje uma das melhores e mais populares ferramentas de OCR
- Repositório oficial: github.com/tesseract-ocr/tesseract



TESSERACT

- A **primeira versão** do Tesseract oferecia suporte somente para Inglês
- Na **segunda versão** já havia suporte para o português brasileiro, assim como para francês, italiano, alemão, espanhol e holandês
- A **terceira versão** expandiu drasticamente o suporte para incluir idiomas ideográficos (simbólicos) como japonês e chinês, bem como idiomas de escrita da direita para a esquerda, como árabe e hebraico
- A **quarta** (e atual versão – junho de 2021) oferece suporte para mais de 100 idiomas, para caracteres e símbolos
- Pytesseract: <https://pypi.org/project/pytesseract/>

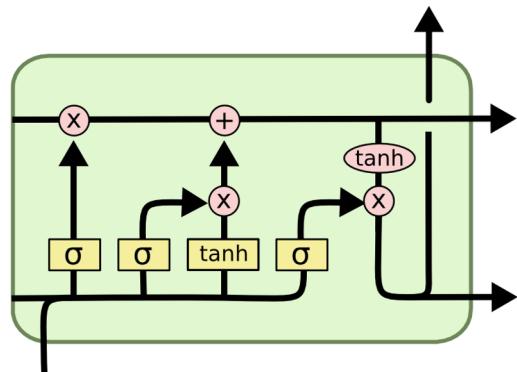
TESSERACT



Créditos da imagem: [Balaaji Parthasarathy](#)

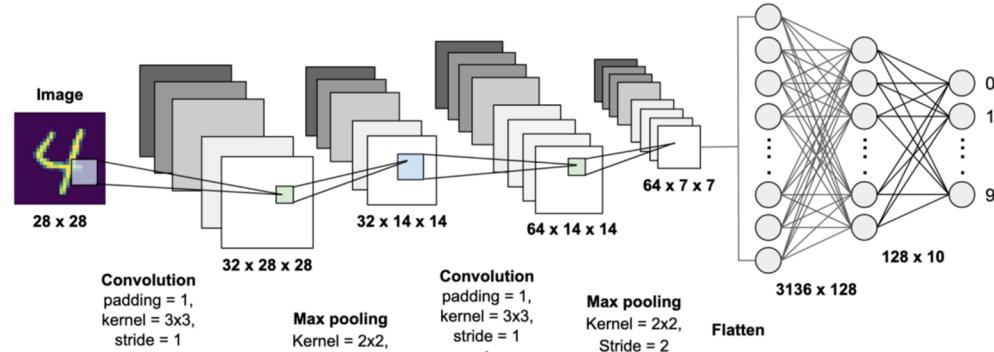
TESSERACT

Para reconhecer uma imagem contendo um único caractere, normalmente usamos uma Rede Neural Convolucional (CNN)



Estrutura de uma LSTM.

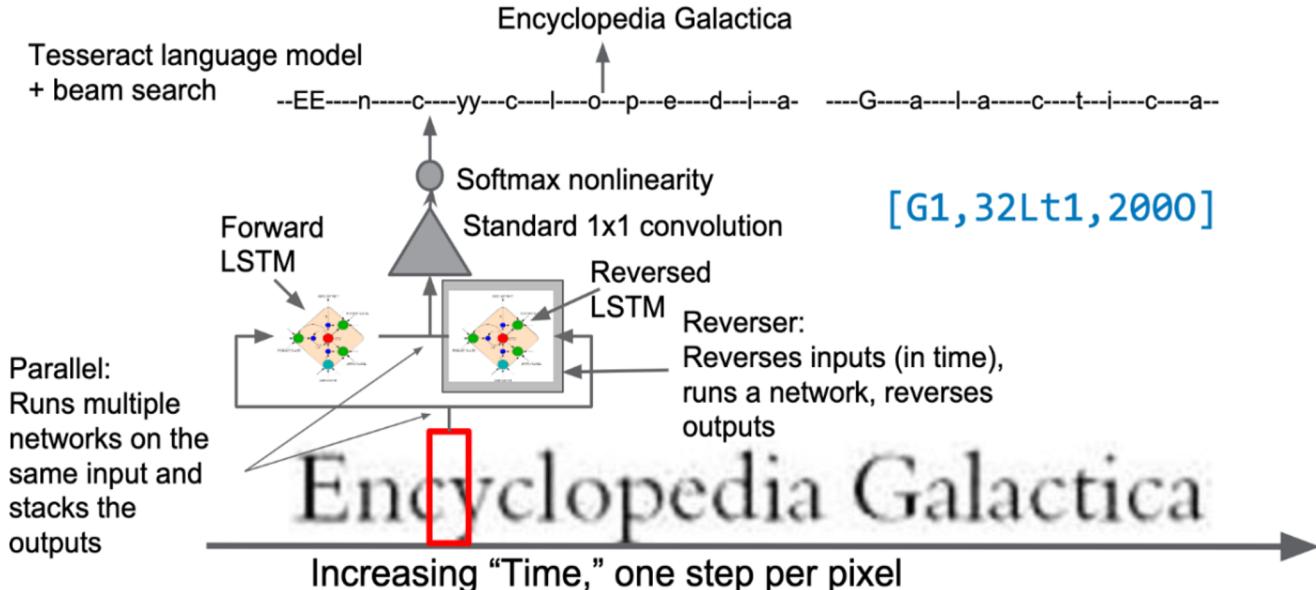
Créditos da imagem: [Colah's blog](#)



Créditos da imagem: Krut Patel

Um texto de comprimento arbitrário é uma sequência de caracteres e é mais interessante utilizar RNNs (*Redes Neurais Recorrentes*). **LSTM** (*Long short-term memory*) é uma forma popular de RNN.

TESSERACT



Google

Tesseract Blends Old and New OCR Technology - DAS2016 Tutorial - Santorini - Greece

Créditos da imagem: [Tesseract 3 OCR process paper](#)

TESSERACT

- O Tesseract tem uma função para detectar a orientação do texto na imagem, assim como o alfabeto em que ele é escrito
- Essa opção é chamada de **OSD** - *Orientation and script detection*
- Detectar se o texto da imagem está rotacionado
- Se estiver rotacionado, podemos aplicar algum tipo de pré-processamento

OUTRAS FERRAMENTAS

Além do Tesseract, há outras opções open source para OCR. Por exemplo :

- [EasyOCR](#) – é uma biblioteca que tem como proposta ser uma solução eficiente e prática que, oferece suporte para mais de 80 linguagens (até o presente momento). Sua acurácia bate de frente com os resultados do Tesseract, e sua detecção de textos nativa aplicado em cenários menos controlados demonstrou superar todas as outras soluções.
- [OCropus](#) – solução Open Source que permite fácil avaliação e reutilização dos componentes de OCR por pesquisadores e empresas. É uma coleção de programas de análise de documentos, não um sistema OCR pronto para uso. Para aplicá-lo aos seus documentos, pode ser necessário fazer um pré-processamento de imagem e, possivelmente, também treinar novos modelos.
- [Ocular](#) – funciona melhor em documentos impressos em uma impressora manual, incluindo aqueles escritos em vários idiomas. Ele opera usando a linha de comando.
- [SwiftOCR](#) – biblioteca simples e rápida escrita na linguagem Swift. Usa redes neurais para reconhecimento de imagem.

REFERÊNCIAS

Sites e referências originais

Mais sobre o funcionamento do Tesseract:

- <https://static.googleusercontent.com/media/research.google.com/pt-PT//pubs/archive/33418.pdf>
- https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/6ModernizationEfforts.pdf

Treinar seu próprio modelo para o Tesseract:

- Documentação oficial: <https://tesseract-ocr.github.io/tessdoc/tess4/TrainingTesseract-4.00.html>
- <https://www.endpoint.com/blog/2018/07/09/training-tesseract-models-from-scratch>
- <https://pretius.com/how-to-prepare-training-files-for-tesseract-ocr-and-improve-characters-recognition/>
- <https://vovaprivalov.medium.com/tesseract-ocr-tips-custom-dictionary-to-improve-ocr-d2b9cd17850b>

LIMIARIZAÇÃO - THRESHOLDING

- A limiarização, também chamada de binarização, é o método mais simples de segmentação de imagens
- Consiste em separar uma imagem em regiões de interesse e não interesse através da escolha de um ponto de corte (chamado de limiar, ou *threshold*)



LIMIARIZAÇÃO SIMPLES

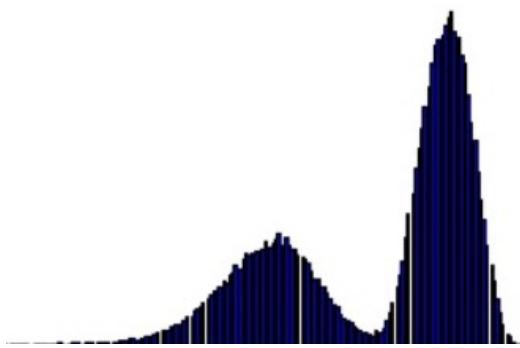
O valor da nova cor que terá o pixel é calculado de acordo com o ponto de corte (limiar ou *threshold*)

Qualquer pixel com intensidade menor ou igual ao limiar passa a ser preto

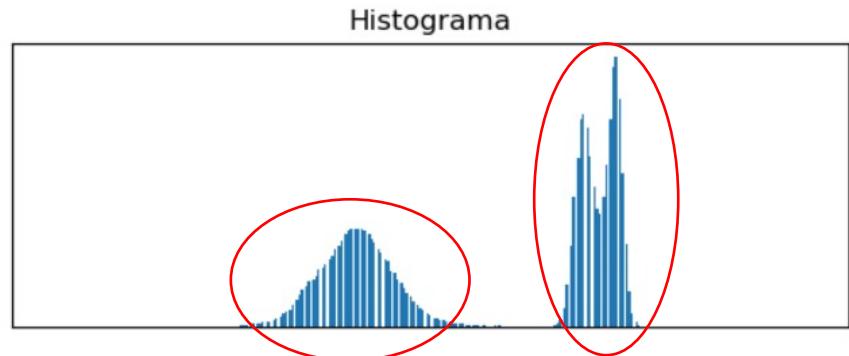
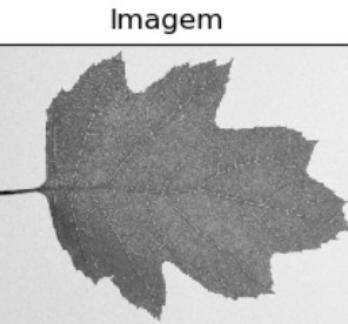
Se o pixel tiver intensidade maior que o limiar passa a ter a cor branca.



MÉTODO DE OTSU



Exemplo de um histograma de uma imagem bimodal

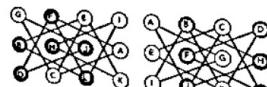


LIMIARIZAÇÃO ADAPTATIVA (GAUSSIANA)

- O limiar é calculado para pequenas regiões da imagem, sendo obtidos diferentes limiares para pequenas regiões da imagem (utilizada a média)
- Gaussiana: também utiliza o desvio padrão (considerada a variação nos tons da imagem) – utiliza convolução

Troca de moedas

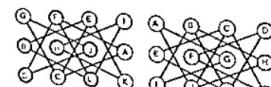
A primeira figura mostra 6 moedas de prata, A, C, E, G, I e K e 6 moedas de ouro, B, D, F, H, J e L. A sua tarefa é mover as moedas para a disposição mostrada na segunda figura. Cada movimento deve consistir em uma troca entre uma moeda de prata e uma moeda de ouro adjacente; duas moedas são adjacentes se estiverem unidas por uma linha reta. Sabe-se que o menor número de movimentos que resolve este quebra-cabeça é 17. Você consegue encontrar uma solução em 17 jogadas?



Mova as moedas da primeira posição para a segunda.

Troca de moedas

A primeira figura mostra 6 moedas de prata, A, C, E, G, I e K e 6 moedas de ouro, B, D, F, H, J e L. A sua tarefa é mover as moedas para a disposição mostrada na segunda figura. Cada movimento deve consistir em uma troca entre uma moeda de prata e uma moeda de ouro adjacente; duas moedas são adjacentes se estiverem unidas por uma linha reta. Sabe-se que o menor número de movimentos que resolve este quebra-cabeça é 17. Você consegue encontrar uma solução em 17 jogadas?



Mova as moedas da primeira posição para a segunda.

REDIMENSIONAMENTO

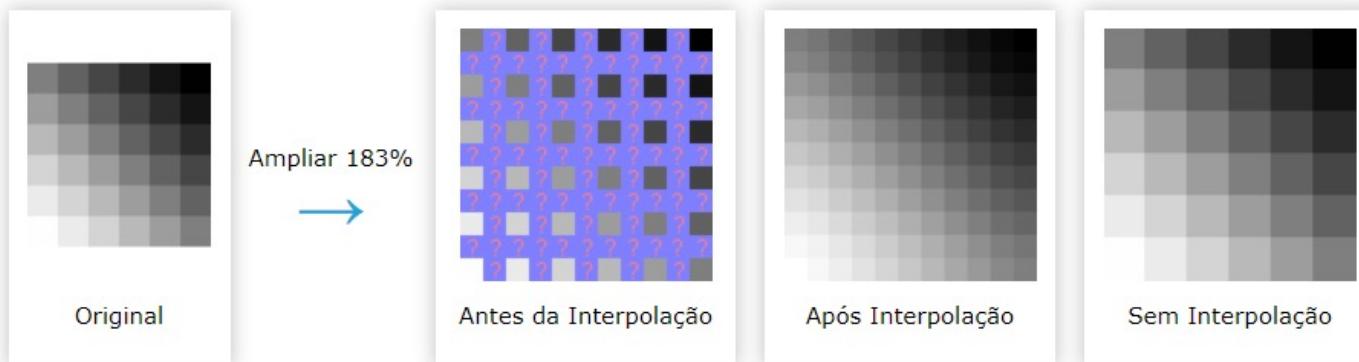
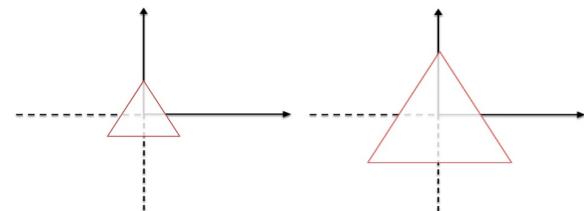
$$\begin{bmatrix} x_T \\ y_T \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Fator de escala:

>1 aumenta

<1 reduz

$s_x = s_y$ fator de escala uniforme (sem distorções)



REDIMENSIONAMENTO

Opções OpenCV

- **INTER_NEAREST** - interpolação de vizinho mais próximo. É muito usado por ser o mais rápido
- **INTER_LINEAR** - interpolação bilinear (é usada por padrão), para aumentar e também pra diminuir imagens.
- **INTER_AREA** - relação de área de pixel. Pode ser um método preferido para a redução de imagens porque fornece resultados sem moiré (efeito geralmente indesejado na imagem). Mas quando a imagem é ampliada, é semelhante ao método INTER_NEAREST
- **INTER_CUBIC** - bicúbica (4x4 pixel vizinhos). Possui resultados melhores
- **INTER_LANCZOS4** - interpolação Lanczos (8x8 pixel vizinhos). Dentre esses algoritmos, é o que apresenta resultados com a melhor qualidade

REDIMENSIONAMENTO

Comparação entre os Algoritmos de Interpolação – aumentando a imagem

original (50x50)



Region-based segmentation

With this algorithm, the image is divided into several regions. These regions are processed in an attempt to keep them as they are. This leads to a better result in blurring, but it also leads to two extreme possibilities: blur or no blur.

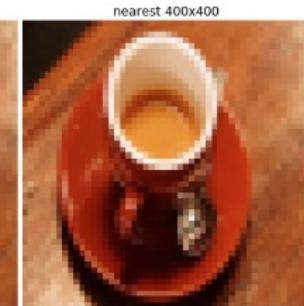
área (400x400)



Region-based segmentation

With this algorithm, the image is divided into several regions. These regions are processed in an attempt to keep them as they are. This leads to a better result in blurring, but it also leads to two extreme possibilities: blur or no blur.

nearest (400x400)



Region-based segmentation

With this algorithm, the image is divided into several regions. These regions are processed in an attempt to keep them as they are. This leads to a better result in blurring, but it also leads to two extreme possibilities: blur or no blur.

linear (400x400)



Region-based segmentation

With this algorithm, the image is divided into several regions. These regions are processed in an attempt to keep them as they are. This leads to a better result in blurring, but it also leads to two extreme possibilities: blur or no blur.

cubic (400x400)



Region-based segmentation

With this algorithm, the image is divided into several regions. These regions are processed in an attempt to keep them as they are. This leads to a better result in blurring, but it also leads to two extreme possibilities: blur or no blur.

lanczos4 (400x400)



Region-based segmentation

With this algorithm, the image is divided into several regions. These regions are processed in an attempt to keep them as they are. This leads to a better result in blurring, but it also leads to two extreme possibilities: blur or no blur.

<http://tanbakuchi.com/posts/comparison-of-opencv-interpolation-algorithms/>

REDIMENSIONAMENTO

Comparação entre os Algoritmos de Interpolação – diminuindo a imagem

original (400x400)



Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

área (50x50)



Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

nearest (50x50)



Region-based segmentation

In this step, we want to find markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

linear (50x50)



Region-based segmentation

In this step, we want to find markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

cubic (50x50)



Region-based segmentation

In this step, we want to find markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

lanczos4 (50x50)



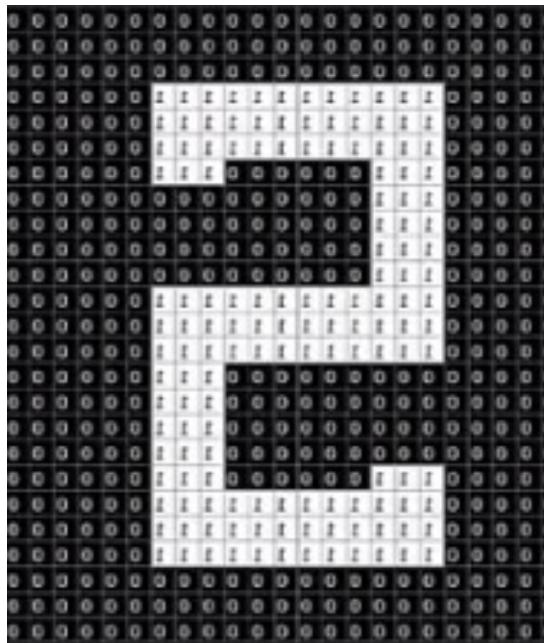
Region-based segmentation

In this step, we want to find markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

<http://tanbakuchi.com/posts/comparison-of-opencv-interpolation-algorithms/>

OPERAÇÕES MORFOLÓGICAS – EROSÃO E DILATAÇÃO



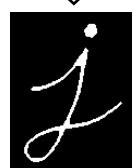
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

EROSÃO

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DILATAÇÃO

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |



OPERAÇÕES MORFOLÓGICAS – ABERTURA E FECHAMENTO

DILATACÃO

ABERTURA



DILATAÇÃO

→ EROSÃO

FECHAMENTO



REMOÇÃO DE RUÍDO COM DESFOQUE

Os filtros espaciais são implementados através de kernels (máscaras/matrizes), com dimensões ímpares. Os filtros podem ser:

- **Passa-baixa (Low-pass)** – usados para desfoque
- **Passa-alta (High-pass)** – usados para aumentar nitidez



CONVOLUÇÃO

- Convolução é uma operação matemática realizada em duas matrizes, que produz uma terceira matriz que é o resultado da operação
- A matriz primária é a imagem a ser tratada, e o tratamento dessa matriz da imagem original se dá por outra matriz chamada “núcleo” (**kernel**), ou máscara
- Dependendo dos valores do kernel é possível obter filtros de diferentes tipos, como desfoque e nitidez

| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 161 | 137 | 244 | 254 | 255 | 254 |
| 154 | 75 | 200 | 249 | 255 | 255 |
| 109 | 96 | 143 | 223 | 255 | 255 |
| 69 | 107 | 196 | 236 | 255 | 255 |
| 51 | 45 | 134 | 218 | 251 | 255 |
| 58 | 56 | 75 | 224 | 255 | 255 |

Imagen Original

$$1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 9$$

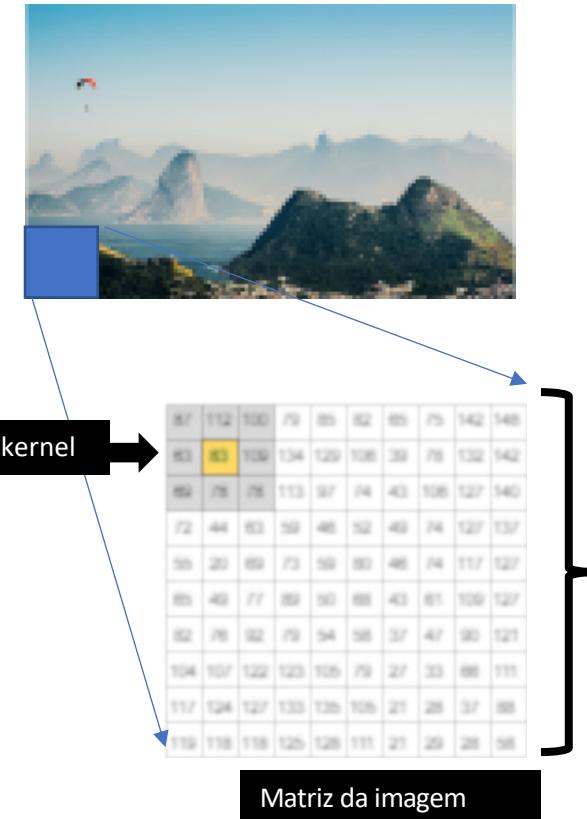
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Kernel - Media

$$\begin{aligned} & 161 + 137 + 244 \\ & 154 + 75 + 200 \\ & 109 + 96 + 143 \\ & = 1319 \end{aligned}$$

Resultado

$$\begin{aligned} & 1319 / 9 \\ & = 147 \end{aligned}$$



DESFOQUE COM MÉDIA

| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 161 | 137 | 244 | 254 | 255 | 254 |
| 154 | 75 | 200 | 249 | 255 | 255 |
| 109 | 96 | 143 | 223 | 255 | 255 |
| 69 | 107 | 196 | 236 | 255 | 255 |
| 51 | 45 | 134 | 218 | 251 | 255 |
| 58 | 56 | 75 | 224 | 255 | 255 |

Imagen Original

x

$$1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 9$$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Kernel - Média

| | | |
|-----|-----|-----|
| 161 | 137 | 244 |
| 154 | 75 | 200 |
| 109 | 96 | 143 |

=

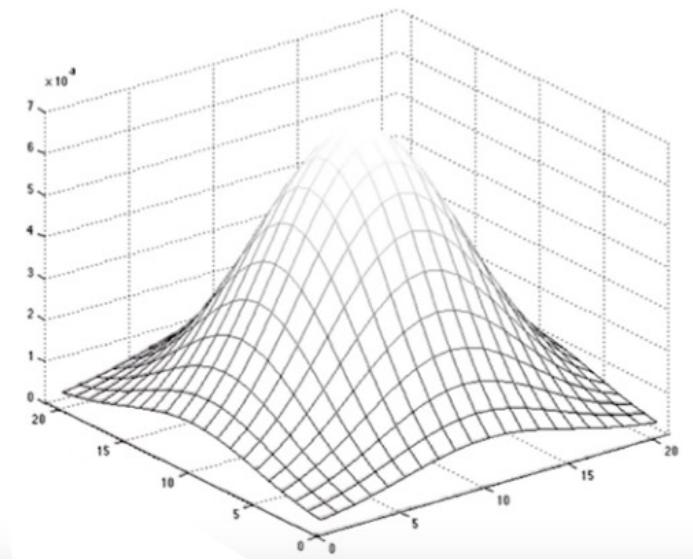
$$\begin{aligned} & 161 + 137 + 244 \\ & 154 + 75 + 200 \\ & 109 + 96 + 143 \\ & = 1319 \end{aligned}$$

Resultado

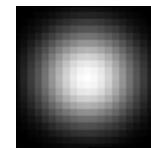
$$\begin{aligned} & 1319 / 9 \\ & = 147 \end{aligned}$$

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

DESFOQUE GAUSSIANO

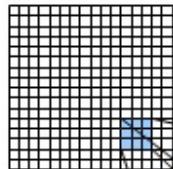


$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



Representação gráfica de um filtro gaussiano 21x21

DESFOQUE COM MEDIANA



| | | |
|-----|-----|-----|
| 101 | 69 | 0 |
| 56 | 255 | 87 |
| 123 | 96 | 157 |



Os pixels abaixo do kernel são ordenados e o valor do meio é o escolhido para ser o novo valor do pixel central (principal)

| | | | | | | | | |
|---|----|----|----|----|-----|-----|-----|-----|
| 0 | 56 | 69 | 87 | 96 | 101 | 123 | 157 | 255 |
|---|----|----|----|----|-----|-----|-----|-----|

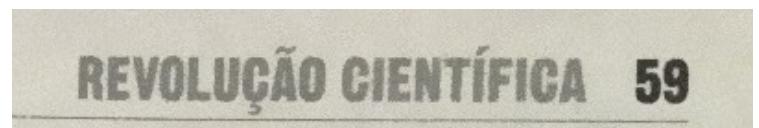
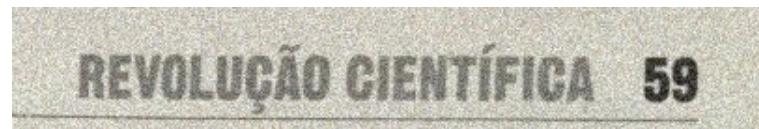
2, 2, 3, **7**, 8, 9, 9

Mediana = **7**

1, 4, 4, **5**, **6**, 7, 7, 7

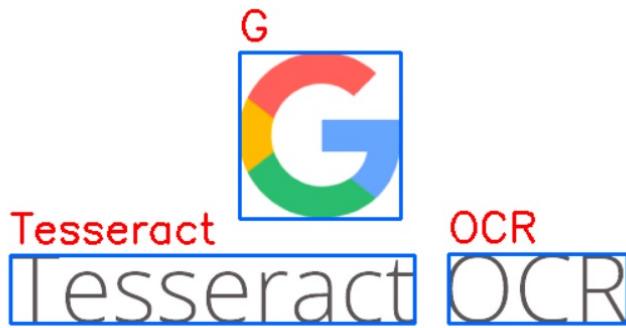
Mediana = $(5+6) \div 2$
= **5.5**

FILTRO BILATERAL



DETECÇÃO DE TEXTOS

A **deteção** do texto antes do
reconhecimento é uma etapa **essencial**



```
'conf': ['-1', '-1', '-1', '-1', 90, '-1', 74, 66],  
'height': [400, 236, 236, 92, 92, 87, 76, 87],  
'left': [0, 38, 38, 38, 38, 102, 102, 307],  
'text': ['', '', '', '', 'TESTANDO', '', '0', 'OCR...'],  
'top': [0, 79, 79, 79, 79, 228, 233, 228],  
'width': [700, 607, 607, 607, 607, 532, 77, 327],
```

Exemplo de imagens onde não haveria a necessidade de detecção previamente



EAST – DETECTOR DE TEXTOS

O **EAST** (*Efficient Accurate Scene Text detector*) é um modelo de **aprendizagem profunda (deep learning)**, publicado oficialmente em 2017 por Zhou et al.

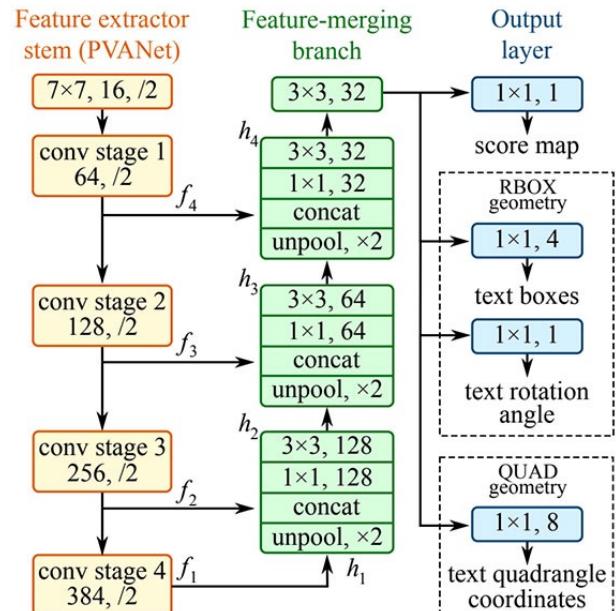
- Usa camadas convolucionais para extrair características das imagens e dessa forma **detectar** a presença de textos
- Consiste apenas **em identificar a localização do texto**. Portanto, não é um OCR que vai convertê-lo em caracteres (para isso usamos Tesseract após a detecção)
- Por utilizar uma arquitetura moderna, é uma das técnicas de detecção de texto mais precisas atualmente
- É também uma das mais conhecidas e ganhou popularidade após o release da versão 3.4.2 do OpenCV (e posteriormente a versão 4), que possibilitou implementá-la mais facilmente através do módulo DNN

EAST – DETECTOR DE TEXTO

Durante seu processamento, características em diferentes níveis de processamento são retirados e tratados matricialmente, levando à identificação de caixas delimitadoras (*bounding boxes*) onde o texto aparece na imagem

- Seu funcionamento consiste em apenas dois estágios: sua rede **FCN** (Fully Convolutional Network) e a técnica **NMS** (supressão não máxima) para suprimir caixas delimitadoras fracas e sobrepostas.

Estrutura do EAST (Fully-Convolutional Network)



Créditos: [Zhou et al.](#)

Referências

Sites e referências originais

Paper do EAST

<https://arxiv.org/abs/1704.03155v2>

Mancas-Thillou e Gosselin

https://www.tcts.fpms.ac.be/publications/regpapers/2007/VS_cmtbg2007.pdf

Repositório Tesseract OCR

<https://github.com/tesseract-ocr/tesseract>

Explicação sobre a geometria do EAST

<https://stackoverflow.com/questions/55583306/decoding-geometry-output-of-east-text-detection>