Visão Computacional: reconhecimento de texto com OCR e OpenCV

cursos.alura.com.br/course/visao-computacional-reconhecimento-texto-ocr-opencv/task/112865

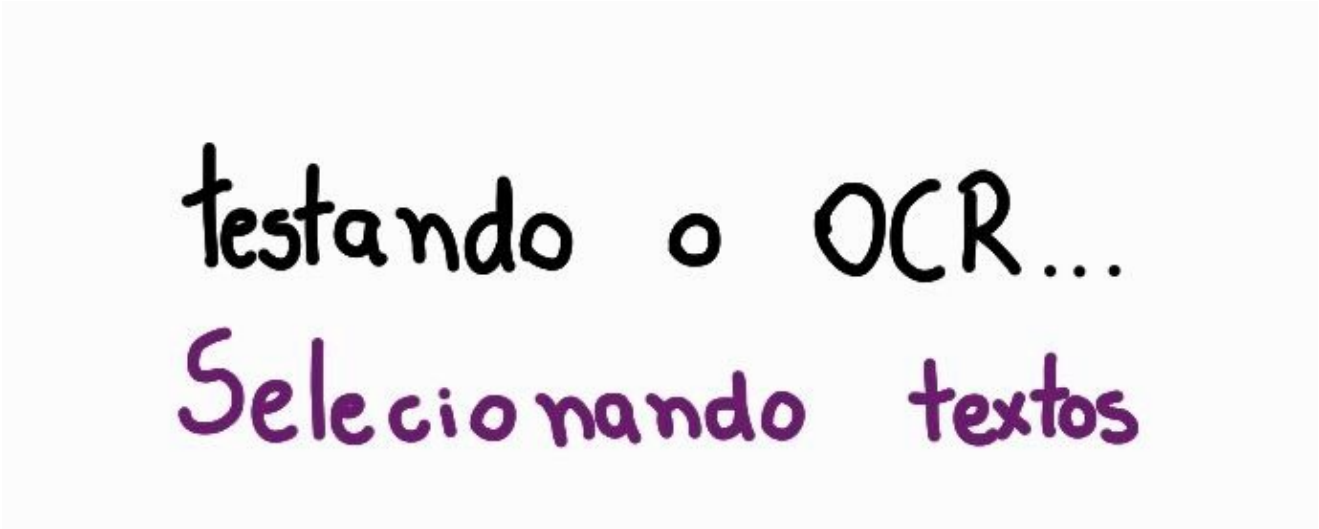
Na aula passada, estudamos um pouco sobre o funcionamento do OSD e o que ele significa. Ele é importante para compreendermos os metadados da nossa imagem e também quais são os metadados ligados ao OCR. Agora, faremos uma análise mais minuciosa com o OSD.

Para isso, vamos importar um novo módulo do PyTesseract chamado **Output**, mas, antes, adicionaremos uma nova imagem para análise, da mesma maneira que fizemos anteriormente: `img = cv2.imread` seguido do `content` da imagem, sito é, o caminho.

Neste caso, o caminho será o da "Aula3-testando.png". Em seguida, realizamos a conversão para RGB com `rgb = cv2.cvtColor()`, passando os parâmetros da imagem, `img`, e `cv2.COLOR_BGR2RGV`. Para finalizar, mostraremos a imagem que chamamos de `rgb`, com `cv2_imshow(rgb)`.

```
img = cv2.imread('/content/text-recognize/Imagens/Aula3-testando.png')
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
cv2_imshow(rgb)
```

A imagem é um texto manuscrito de duas linhas. Na primeira linha, está escrito "testando o OCR..", em preto, com "testando o" em minúsculo e "OCR" em maiúsculo. Na segunda linha, está escrito "Selecionando textos" em roxo, com "S" maiúsculo e o resto do texto em minúsculo.



Esse caso difere dos que já analisamos por ser manuscrito. Além disso, as cores das frases são diferentes e algumas letras são maiúsculas e outras minúsculas. Vamos conferir como o OCR vai se comportar. Os metadados nos permitirão compreender como o OCR está funcionando internamente, não só o resultado que nos oferece. Vamos importar o método `Output` do `pytesseract`.

```
from pytesseract import Output
```

O `Output` começa com "O" maiúsculo exatamente por ser um método. O próximo código não é muito diferente dos que já utilizamos, porém, agora temos um parâmetro a mais dentro do `.image_to_data()`.

```
config_tesseract = '--tessdata-dir tessdata'
resultado = pytesseract.image_to_data(rgb, config=config_tesseract, lang='por',
output_type=Output.DICT)
resultado
```

Na parte de resultado de texto, `resultado` é igual a `pytesseract.image_to_data()` e, dentro dos parênteses, passamos os parâmetros: a nossa imagem, `rgb`; `config=config_tesseract`, que é da pasta `tessdata` e foi definido na linha acima; o `lang='por'`, pois a leitura é em português; e, por último, `output_type=Output.DICT`, com "O" maiúsculo, porque estamos chamando o método.

Com o `output_type=Output.DICT`, receberemos uma espécie de dicionário que diz sobre o que ele está vendo na imagem. Por fim, ao invés de um fazermos `print()`, em que o resultado é o texto, isto é, o que ele leu da imagem, "testando o OCR" e "Selecionando textos", queremos saber os metadados da imagem. O resultado será:

```
 {'block_num': [0, 1, 1, 1, 1, 1, 1, 1, 1, 1],
'conf': ['-1', '-1', '-1', '-1', 63, 90, 48, '-1', 85, 77],
'height': [276, 159, 159, 73, 69, 28, 59, 62, 62, 49],
'left': [0, 92, 92, 94, 94, 348, 425, 92, 92, 474],
'level': [1, 2, 3, 4, 5, 5, 5, 4, 5, 5],
'line_num': [0, 0, 0, 1, 1, 1, 1, 2, 2, 2],
'page_num': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
'par_num': [0, 0, 1, 1, 1, 1, 1, 1, 1, 1],
'text': ['',
 '',
 '',
 '',
 'Testando',
 'o',
 'OCR..',
 '',
 'Selecionando',
 'textos'],
'top': [0, 74, 74, 74, 74, 111, 88, 171, 171, 184],
'width': [688, 502, 502, 491, 204, 28, 160, 502, 328, 120],
'word_num': [0, 0, 0, 0, 1, 2, 3, 0, 1, 2]}
```

O retorno tem uma série de valores: block_num; conf; height; left; level; line_num; text; dentre outros. Sobre a frase "testando o OCR...", ele considerou a letra "t" no início da frase como maiúscula, "Testando", porque está um pouco maior que o outro "t" no meio da palavra, "tando". Além disso, considerou apenas dois pontos, "..".

No Colab, que está no Projeto da aula, nós disponibilizamos um dicionário com a definição de cada um dos valores. O que nos interessa agora, é que a parte de `text` tem uma correlação muito alta com a de `conf`. `Selecionando` e `textos` são os dois últimos elementos e se referem aos dois últimos da fileira do `conf`, isto é, `85,87`.

Na linha que está acima de `Selecionando`, existe um espaço em branco, `'',`. Na fileira do `conf`, antes do `85`, temos `-1`. Disto, compreendemos que todos os espaços em branco são de confiança -1.

> conf = confiança da predição (de 0 a 100. -1 significa que não foi reconhecido texto)

Podemos definir um mínimo de confiança. Por exemplo, o mínimo de confiança que recebemos do OCR foi 48, então, podemos definir o mínimo como 40 ou 45. Desta forma, o OCR não mostrará os campos vazios ou -1, porque essa é uma informação que não nos interessa. Isso nos ajudará a fazer reconhecimentos melhores no futuro.

Para criarmos o mínimo de confiança, usaremos uma variável chamada `min_conf`. Para que ela fique com uma boa visualização, vamos criar um slide ao lado. Esse slide contém uma barra que podemos mover para variar o tamanho do `conf`. Mas, se definirmos o `conf` como 40, por exemplo, isto é, `min_conf = 40`, a barra andará até o valor definido.

Então, nós passaremos um parâmetro do tipo `slider`, com mínimo zero e máximo 100, conforme a confiança de predição, e podemos variá-la movendo a barra do slide. Como o nosso mínimo é 40, podemos manter, neste caso, o valor 40.

```
min_conf = 40 #@param {type: 'slider', min: 0, max: 100}
```

Na próxima aula, utilizaremos o mínimo de confiança para criarmos uma caixa delimitadora.