Visão Computacional: reconhecimento de texto com OCR e OpenCV

cursos.alura.com.br/course/visao-computacional-reconhecimento-texto-ocr-opencv/task/112867

Já conseguimos colocar nosso *bounding box* na imagem. Agora, vamos usar o `for` que construímos anteriormente e tentaremos colocar o mesmo texto "testando o OCR...", "Selecionando textos".

```
img_copia = rgb.copy()
for i in range(len(resultado['text'])):
  confianca = int(resultado['conf'][i])
  if confianca > min_conf:
    x, y, img = caixa_texto(resultado, img_copia)
cv2_imshow(img_copia)
```

Além da caixa ao redor das letras, do retorno e *print* do texto, queremos que o texto apareça de forma escrita pela máquina. Após conferir se a confiança é maior que a confiança mínima e aplicar na nossa função de `caixa_texto`, ele vai pegar, dentro do nosso resultado, o valor de texto. Esse trecho de código é bastante parecido com o da confiança, `confianca = int(resultado['conf'][i])`. Então, faremos `texto = resultado['text'][i]`.

```
img_copia = rgb.copy()
for i in range(len(resultado['text'])):
  confianca = int(resultado['conf'][i])
  if confianca > min_conf:
    x, y, img = caixa_texto(resultado, img_copia)
    texto = resultado['text'][i]
```

Ele está trazendo o valor de `texto` para o `if` e encaminhando para cada valor de `i` que for maior que a confiança mínima. A próxima etapa é colocar esse `texto` dentro da imagem de cópia, `img_copia`, que já está com o *bounding box*. Para isso, utilizaremos uma função do OpenCV, a `cv2.putText()`.

Como parâmetros da função, nós passaremos: o nome da imagem, `img_copia`; o texto que ele deve adicionar à imagem, `texto`; a posição que ele deve colocar o texto, `(x, y - 10)`, considerando que, com "y-10", ele localizará o texto um pouco acima do eixo y, para que ele não ocupe a mesma posição da linha azul, não fique em cima dela, atrapalhando a leitura; a fonte, `cv2.FONT_HERSHEY_SIMPLEX`; o tamanho do texto, `1.1`; a cor do texto, `(0,0,255)`.

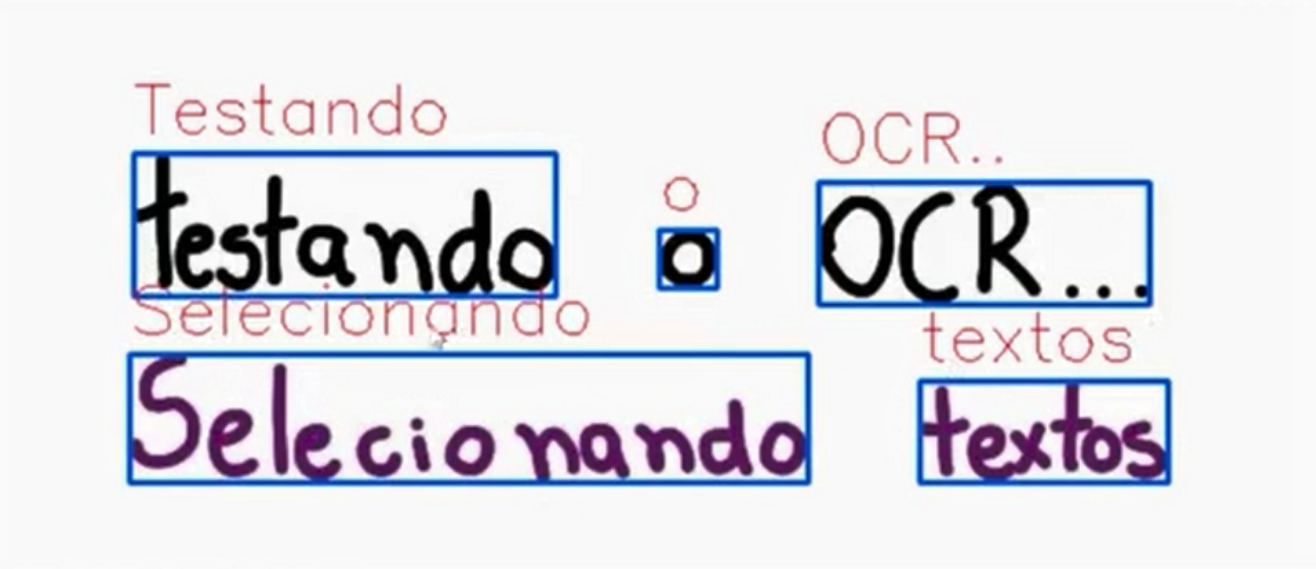
Sobre as cores, algo que precisamos conferir é se elas estão em BGR. Então, quando pesquisamos alguma cor em RGB no Google, precisamos ficar atentos, pois a leitura será ao contrário: BGR. Por exemplo, no nosso caso, o `255` será o valor de R, o G será zero e o B, também será zero. Como o `255` está no final, provavelmente a nossa cor será um vermelho bem forte.

```
img_copia = rgb.copy()
for i in range(len(resultado['text'])):
  confianca = int(resultado['conf'][i])
  if confianca > min_conf:
    x, y, img = caixa_texto(resultado, img_copia)

    texto = resultado['text'][i]
    cv2.putText(img_copia, texto, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.1, (0,0,255))

cv2_imshow(img_copia)
```

O texto já está aparecendo na imagem.

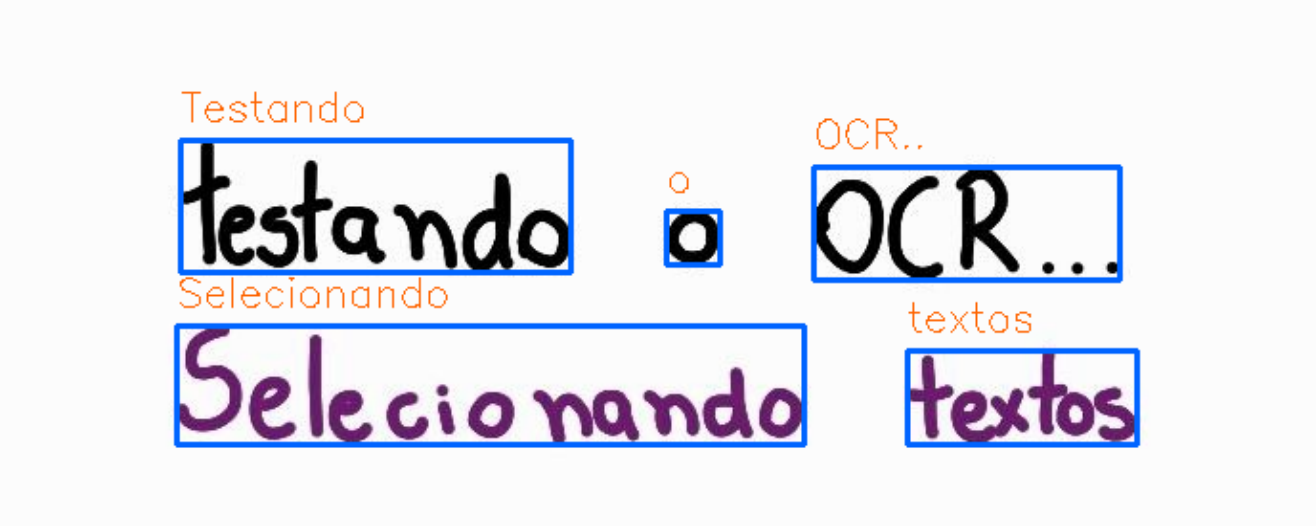


A letra está ultrapassando o *bounding box*, podemos ajustar o tamanho do texto para `0.7` e a cor, BGR, para `(0,100,255)`. O tamanho das letras está menor e a cor, mais alaranjada.

```
img_copia = rgb.copy()
for i in range(len(resultado['text'])):
  confianca = int(resultado['conf'][i])
  if confianca > min_conf:
    x, y, img = caixa_texto(resultado, img_copia)

    texto = resultado['text'][i]
    cv2.putText(img_copia, texto, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,100,255))

cv2_imshow(img_copia)
```



Então, o tamanho 0.7 está bom, porque não sobrepõe nenhum dos textos da imagem e a cor alaranjada é melhor que um vermelho tão vibrante. Assim, conseguimos escrever na nossa imagem, utilizando o OSD. Usamos as informações do dicionário na nossa imagem. Para além do *print* da imagem, conferimos as coordenadas, fizemos o *bounding box* e adicionamos texto.