Visão Computacional: reconhecimento de texto com OCR e OpenCV

cursos.alura.com.br/course/visao-computacional-reconhecimento-texto-ocr-opencv/task/112866

Já sabemos como o OSD funciona, definimos o mínimo de confiança para 40 e podemos começar a criar a nossa caixa delimitadora, **Bounding Box**. Nós vamos colocar o *bounding box* na região de interesse, que é o nosso texto. Em outras palavras, vamos desenhar uma caixa em volta do nosso texto para indicar que é ele que estamos catalogando dentro do OCR.

Para começar, vamos fazer uma função. A caixa de texto funciona usando os valores que já estudamos, por exemplo: *height*, a altura do bloco de texto detectada; *left*, a coordenada x onde inicia a caixa delimitadora; *top*, coordenada y; e *width*, largura do bloco de texto atual detectado. Portanto, essas são as coordenadas que determinam onde está o texto, isto é, onde ele começa e onde termina.

Essas delimitações viabilizam a construção da nossa caixa de texto. A função da caixa delimitadora, é:

```
def caixa_texto(resultado, img, cor = (255, 100, 0)):
  x = resultado['left'][i]
  y = resultado['top'][i]
  w = resultado['width'][i]
  h = resultado['height'][i]

  cv2.rectangle(img, (x, y), (x, y), cor, 2)

  return x, y, img
```

Primeiro, definimos uma função chamada `caixa_texto()`. Essa função receberá `resultado`, que está no código do dicionário, onde estão os valores de x e y que já estudamos.

Ela também receberá a imagem, `img`, e a cor, `cor`, que será em BGR, pois estamos trabalhando com OpenCV. Lembrando que o `255` se refere ao canal "B", o `100` ao "G" e o `0` ao "R". Por esses dados, já conseguimos prever que a imagem terá mais tons de azul que de vermelho, por exemplo, pois o vermelho marca zero e o azul, 255.

Então, vamos chamar `x`, `y`, `w` e `h`, coordenadas equivalentes aos valores do dicionário: left, top, width e height. Todos variando conforme `i`. Como podemos ter várias imagens/textos, não podemos limitar o código a um único valor. Depois, para construirmos o retângulo, vamos usar uma função do OpenCV que é a `.rectangle()`.

Portanto, chamamos essa função com `cv2.rectangle()` e, nos parâmetros, passamos: `img`; o ponto inicial, `(x,y)`; o ponto final, `(x,y)` também, porque ele fechará um retângulo; a cor, `cor`; e o *thickness*, que é o tamanho da borda, `2`. O retorno da nossa função será: `x`, `y` e `img`. Na

próxima célula, para testarmos a função, escreveremos `caixa_texto()`, passando `resultado` e `rgb`.

```
caixa_texto(resultado, rgb)
```

Deu um erro: o nome `i` não está definido. Recebemos um alerta de que estamos tentando fazer uma alteração que não existe, portanto, vamos definir o `i` e depois fazemos o `caixa_texto()`. Então, vamos apagar a célula do `caixa_texto` e criar uma nova, onde definirmemos um `for`. Nós faremos um `len()` do resultado para descobrirmos quantos textos textos, isto é, a quantidade de textos que ele tenta detectar, inclusive os valores de "-1".

```
len(resultado['text'])
```

> 10

```
for i in range()
```

Enquanto ele estiver nesse `range` do `len(resultado['text'])`, fará o nosso `for`.

```
len(resultado['text'])
```

> 10

```
for i in range(len(resultado['text'])):
```

Nós continuaremos o código construindo a parte da confiança, porque, dentre os 10 resultados, alguns são inúteis, é o caso do "-1". Não desejamos que ele faça uma caixa que não tenha nenhum valor/texto. Por isso, faremos `confianca` igual a `int()`, porque desejamos que o valor seja inteiro, e, nos parênteses, `resultado['conf']`, isto é, o resultado da confiança e o valor de `[i]`.

```
for i in range(len(resultado['text'])):
  confianca = int(resultado['conf'][i])
```

Primeiro, ele pegará o valor da confiança. Se esse valor de confiança for maior que o mínimo de confiança que nós definimos dentro do parâmetro, 40, ele segue para `x, y, img = caixa_texto(resultado, img_copia)`.

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

Também precisamos fazer uma imagem de cópia para que a nossa imagem não seja sobrescrita. É um preciosismo: estamos adicionando uma nova imagem para que não seja necessário colocar o *bounding box* dentro da imagem original. Se quisermos, podemos salvar essa cópia e ainda teremos a imagem original, disponível para ser usada em outros projetos.

Acima do código, para a cópia, escreveremos `img_copia = rgb.copy()`, sem passar nada como parâmetro, só fazendo a imagem de cópia. Ao final do código, faremos `cv2_imshow(img_copia)`.

```
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)
```

Ele nos retornou a imagem de cópia, porém ela está com pequenos pontos azuis, que são os pontos de x e y. Vamos observar o código da nossa função:

```
def caixa_texto(resultado, img, cor = (255, 100, 0)):
  x = resultado['left'][i]
  y = resultado['top'][i]
  w = resultado['width'][i]
  h = resultado['height'][i]

  cv2.rectangle(img, (x, y), (x, y), cor, 2)

  return x, y, img
```

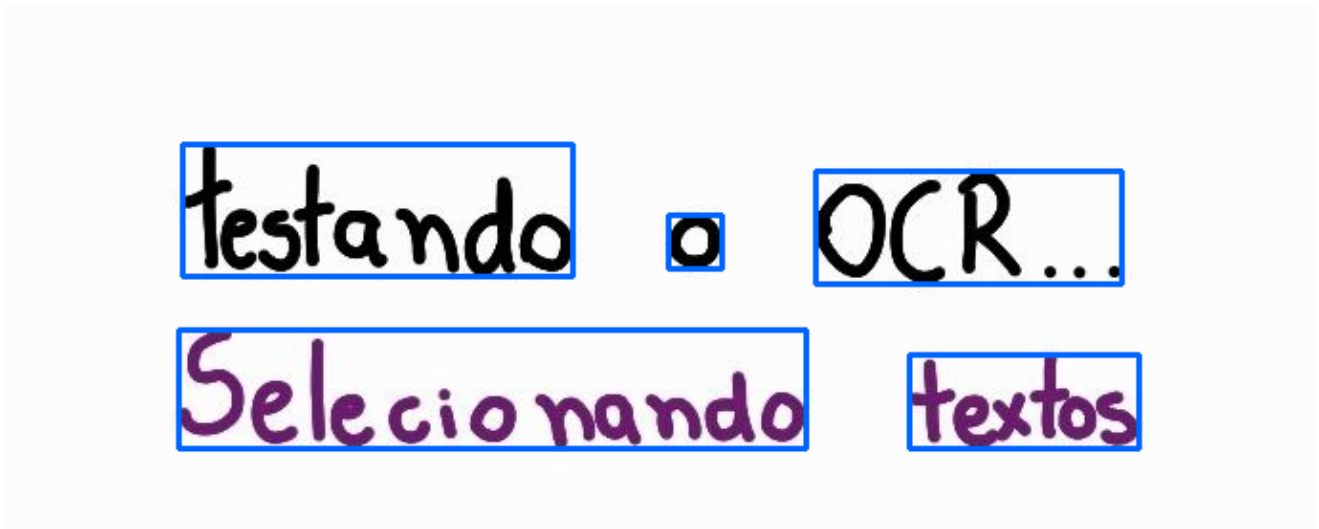
Ele está fazendo o retângulo, começando com "x" e "y". Mas, não estamos passando nada sobre o "w" e o "h". Então, precisamos corrigir para `x+w` e `y+h`, rodar a função e o código de cópia de imagem.

```
def caixa_texto(resultado, img, cor = (255, 100, 0)):
  x = resultado['left'][i]
  y = resultado['top'][i]
  w = resultado['width'][i]
  h = resultado['height'][i]

  cv2.rectangle(img, (x, y), (x+w, y+h), cor, 2)

  return x, y, img
```

Agora ele fez o *bounding box* ou **caixa de detecção de texto**.



Vamos voltar ao nosso código `img_copia` e analisá-lo:

```
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)
```

Primeiro, conferimos se a confiança é maior que a confiança mínimo - no nosso caso, a confiança mínima é 40 - e depois, aplicamos dentro da função que criamos. Assim, visualizaremos o funcionamento da função na nossa imagem.

A nossa função está correta e funcionando. Se quisermos mudar a cor do *bounding box*, basta trocar dentro da função. Nosso próximo passo é tentar adicionar um texto em cima da imagem, mas isso é assunto para a próxima aula!