Visão Computacional: reconhecimento de texto com OCR e OpenCV

cursos.alura.com.br/course/visao-computacional-reconhecimento-texto-ocr-opencv/task/112877

Depois de alguns passos, descobrimos onde estavam os termos em cada uma das imagens e precisamos e agora faremos o reconhecimento nas próprias imagens. Para isso, importaremos algumas das funções que já fizemos nas aulas anteriores, caso você não se lembre de alguma delas, basta voltar à algumas aulas e revisar. Começaremos pela fonte, Calibri.

```
fonte_dir = '/content/text-recognize/Imagens/calibri.ttf'
```

Em outra célula, importaremos a função `escreve_texto`, que fizemos para utilizar a fonte Calibri e escrever textos com a biblioteca PIL.

```
def escreve_texto(texto, x, y, img, fonte_dir, cor=(50, 50, 255), tamanho=16):
  fonte = ImageFont.truetype(fonte_dir, tamanho)
  img_pil = Image.fromarray(img)
  draw = ImageDraw.Draw(img_pil)
  draw.text((x, y-tamanho), texto, font = fonte, fill = cor)
  img = np.array(img_pil)

  return img
```

Também vamos importar o nível mínimo de confiança, que, por enquanto, deixaremos em 30.

```
min_conf = 30  #@param {type:"slider", min:0, max:100}
```

Outra função que usaremos é a `caixa_texto` para o *bounding box*.

```
def caixa_texto(i, 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
```

Para encontrarmos as palavras, usaremos um `for` bastante semelhante ao do vídeo passado, `for imagem in caminho`. Mas, ao invés de utilizarmos `texto = OCR_processa(img, config_tesseract)`, faremos um `OCR_processa()` para as imagens.

```
def OCR_processa_imagem(img, termo_pesquisa, config_tesseract, min_conf):
  resultado = pytesseract.image_to_data(img, config=config_tesseract, lang='por',
output_type=Output.DICT)
  num_ocorrencias = 0

  for i in range(0, len(resultado['text'])):
    confianca = int(resultado['conf'][i])
    if confianca > min_conf:
      texto = resultado['text'][i]
      if termo_pesquisa in texto:
        x, y, img = caixa_texto(i, resultado, img, (0,0,255))
        img = escreve_texto(texto, x, y, img, fonte_dir, (50,50,225), 14)

        num_ocorrencias += 1
  return img, num_ocorrencias
```

Então, faremos `def OCR_processa_imagem()`. Os termos que passaremos, são a imagem, o termo de pesquisa, o `config_tesseract` e o mínimo de confiança, que definimos como 30. Na próxima linha, `resultado = pytesseract.image_to_data()`, passando a imagem, o `config=config_tesseract`, o `lang='por'` e o `output_type=Output.DICT`. Neste trecho, estamos fazendo a imagem para dados.

Depois, vamos inicializar o número de ocorrências com zero. Mais abaixo, faremos o número de ocorrências `+= 1`. Com isso, ele fará uma iteração, a cada vez que a ocorrência terminar, ele faz uma iteração e volta para o laço até acabar o texto.

Em seguida, vamos criar um `range()` de zero até o tamanho do resultado `text`. Na próxima linha, faremos confiança igual ao valor do resultado da confiança em cada uma das imagens. É o mesmo que fizemos anteriormente, mas, nesse caso, com base em `i`, porque temos mais de uma imagem.

Se a confiança for maior que a confiança mínima, ele passa para a outra linha, onde texto é igual ao resultado do texto no momento `i`. Depois, se o termo de pesquisa estiver no texto, ele desenha a caixa de texto e faz a função `escreve_texto`. Então, ele vai fazer o *bounding box* e escreverá o texto em si.

O próximo passo, nós já comentamos: fazer o número de ocorrências, isto é, uma iteração de `+1` e voltar para o laço até acabar o texto. Talvez demore um pouco, exatamente por processar todas as imagens. Por fim, ele retornará a imagem e o número de ocorrências. Vamos rodar!

Nós não queremos passar uma imagem por vez, mas, sim, as quatro juntas. Sendo assim, na próxima célula vamos fazer um código que também é bastante parecido com o `for imagem in caminho`.

```
termo_pesquisa = 'learning'

for imagem in caminho:
  img = cv2.imread(imagem)
  img_original = img.copy()

  nome_imagem = os.path.split(imagem)[-1]
  print('==================\n' + str(nome_imagem))

  img, numero_ocorrencias = OCR_processa_imagem(img, termo_pesquisa, config_tesseract,
min_conf)
  print('Número de ocorrências para {} em {}: {}'.format(termo_pesquisa, nome_imagem,
numero_ocorrencias))

  print('\n')

  mostrar(img)
```

O termo de pesquisa vai continuar sendo "learning". Na outra linha, abriremos o nosso `for`. Ele vai carregar a imagem com o `cv2.imread()`. Vai criar uma cópia da imagem, o que é muito importante, principalmente neste caso, em que temos uma pasta com as imagens. Não podemos escrever na imagem original, ela poderia ser um documento, o currículo de alguém ou um comprovante de compra.

Prosseguindo, vamos passar a imagem e acessar a última posição do diretório, usando `os.path.split`. Em seguida, faremos o `print()` de vários símbolos de "igual (=)" com `\n` que serve para pular linha, com o nome da imagem. Então, estamos fazendo a separação mais o nome da imagem.

Em seguida, teremos imagem e número de ocorrências igual a `OCR_processa_imagem()` e passaremos tudo o que ele precisa: imagem, termo de pesquisa, `config_tesseract` e o mínimo de confiança. Depois, ele vai fazer o `print()` do número de ocorrências para o termo de pesquisa, que é "learning", em nome da imagem e número de ocorrências. Por fim, ele fará `mostrar(img)`.

Lembrando que o primeiro par de chaves é para mostrar o termo de pesquisa, o segundo, para o nome das imagens e o terceiro é o número de ocorrências.

No resultado, com as quatro imagens, visualizamos primeiro "artigo-termos-ML.png", que é o "Número de ocorrências para learning em artigo-termos-ML.png: 1". Ele fez o *bounding box* em vermelho na imagem, destacando a palavra "learning" e repetiu o processo para as demais imagens.

Então, conseguimos fazer o *bounding box* e também a detecção. Porém, existem alguns, ele não detectou o termo "learning" quando foi escrito em "L" maiúsculo, "Learning". Portanto, não está conseguindo detectar palavras com letras diferentes.

Ele precisa encontrar as palavras, independentemente de terem caracteres em caixa alta. Vou deixar essa implementação como desafio e logo mais eu volto para resolver com você! Até mais!