Visão Computacional: reconhecimento de texto com OCR e OpenCV

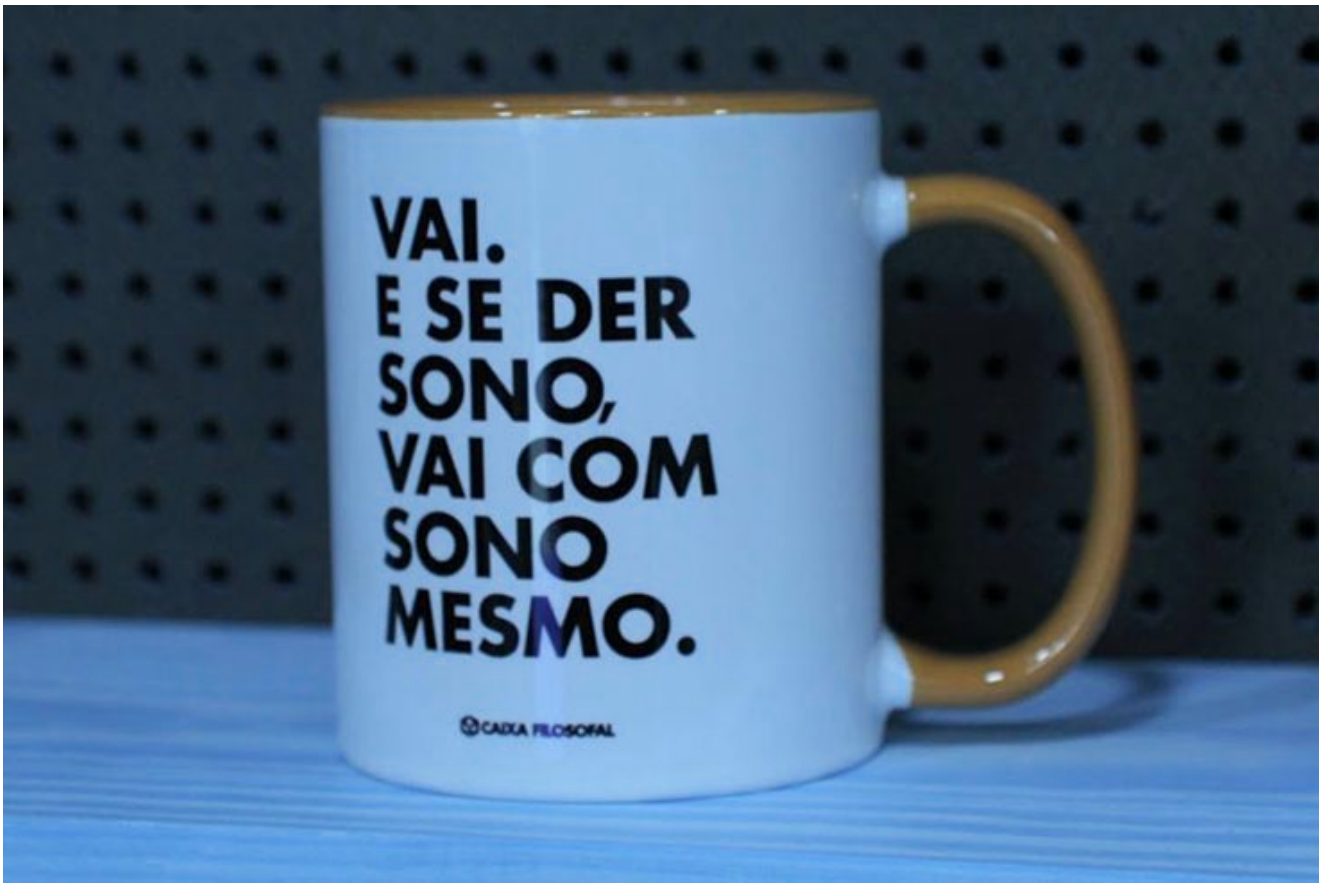
cursos.alura.com.br/course/visao-computacional-reconhecimento-texto-ocr-opencv/task/112872

Todos os casos que estudamos nas aulas anteriores eram totalmente controlados. Isto significa ter controle da luz, das sombras e visualizar os textos de forma limpa/legível. Agora, vamos conhecer um pouco mais sobre cenários naturais.

Lidamos com **cenários naturais** quando fotografamos, por exemplo, uma xícara de café da manhã ou a placa de um *outdoor*. Nestes casos, precisamos aplicar o OCR à vida real. Além disso, os cenários naturais demandam mais pré-processamento das imagens, já que estamos lidando com fotografias com mais sombra e luz.

Vamos analisar a fotografia de uma caneca que está na "Aula4-caneca2.jpg". Nós utilizaremos o código de importação das imagens que já utilizamos anteriormente: `img = cv2.imread()`, com o caminho da imagem que estamos utilizando; depois fazemos a conversão de BGR para RGB; por fim, fazemos `cv2_imshow(rgb)`.

```
img = cv2.imread('/content/text-recognize/Imagens/Aula4-caneca2.jpg')
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
cv2_imshow(rgb)
```



Se trata de uma caneca comum e a luz que incide sobre ela cria uma sombra da alça na mesa. Na frente da caneca, onde está a frase, também é possível visualizar o reflexo da luz. Não é um cenário totalmente controlado. Além disso, o texto que está escrito na caneca, "Vai, e se der sono, vai com sono mesmo." não está no papel, por isso, nosso trabalho será um pouco mais complexo.

Após carregar a imagem, vamos fazer o `config_tesseract` e, com isso, o `DICT`, que é o nosso dicionário. Provavelmente, o PSM dessa imagem será o 6, o de um bloco de texto só. Se não colocarmos o PSM 6, talvez teremos o problema discutido na aula de PSM: não vai aparecer nada ou aparecerá um texto que não faz nenhum sentido. Então, vamos definir o PSM.

```
config_tesseract = '--tessdata-dir tessdata --psm 6'
```

Agora, vamos definir o mínimo de confiança com o `slider`, que também já conhecemos.

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

Vamos deixar em 40, por enquanto, mas podemos mudar de acordo com o dicionário. Em seguida, vamos trazer a parte de `resultado`.

```
resultado = pytesseract.image_to_data(rgb, lang="por", output_type=Output.DICT,
config=config_tesseract)
resultado
```

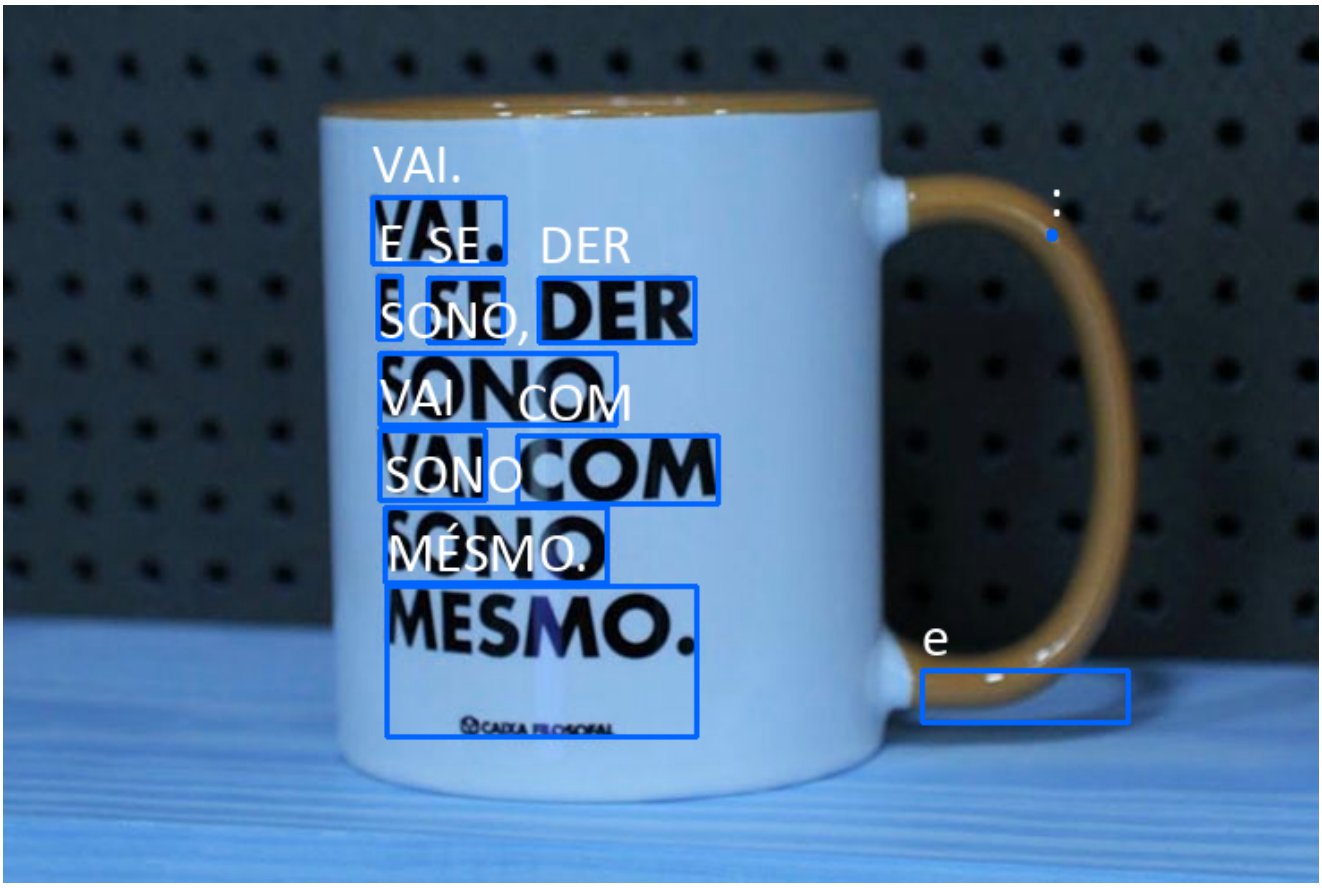
O retorno é bastante extenso e você pode conferir por completo no Projeto da aula, disponível neste link. Mas, nosso interesse é no valor de confiança mínima. Aparentemente, a confiança mínima é 48. Abaixo, é possível conferir o trecho do resultado em que esse valor aparece.

```
'-1',
  '-1',
  56,
  56,
  91,
  '-1',
  '-1',
  91,
  '-1',
  '-1',
  48,
  90,
  '-1',
  '-1',
  89,
  '-1',
  '-1',
  60,
  54],
 'height': [490,
```

Para a próxima etapa, vamos utilizar o código das imagens que fizemos na aula passada. Nós estamos utilizando a `escreve_texto()` com a fonte externa que escolhemos anteriormente, a Calibri.

```
img_copia = rgb.copy()

for i in range(0, len(resultado['text'])):
  confianca = int(resultado['conf'][i])

  if confianca > min_conf:
    x, y, img = caixa_texto(resultado, img_copia)

    texto = resultado['text'][i]
    img_copia = escreve_texto(texto, x, y, img_copia, fonte)

cv2_imshow(img_copia)
```

Nesta célula, ele está fazendo a cópia, depois entra no `for`, confere a confiança. Se a confiança for a mínima, ele constrói a caixa de texto, busca o texto, faz a imagem de cópia e, por último, o `cv2_imshow()` com a imagem de cópia. Vamos visualizar a nossa imagem.



Ele detectou o nome da marca, "caixa filosofal" que está mais abaixo da frase principal e em tamanho menor. Ao lado, perto da alça, detectou "dois pontos (:)" e uma vogal "e" que não existem e que provavelmente são algum reflexo da luz. Esse é um problema que temos em cenários naturais. Se tentarmos só colocar o OSR para rodar, sem um pré-tratamento das imagens, teremos *outliers* ou falsos positivos.

Na próxima aula começaremos a estudar como tratá-los.