Transcrição de Livros Físicos

Obs.: Caso deseje o rápido manuseio, pode utilizar do atalho Ctrl+F9 para rodar todas as células.

Introdução

Escolha do tema

Depois de entrevistar oito participantes, percebemos que aqueles que costumam ler livros físicos frequentemente precisam digitalizar partes dessas leituras para simplificar o estudo e maximizar seu tempo. Estudantes e profissionais afirmaram que frequentemente utilizam livros impressos para estudo, mas enfrentam desafios ao transcrevê-los e organizá-los no ambiente digital. Apesar da crescente adoção de formatos digitais, o acesso a conteúdos impressos ainda é essencial para muitos, seja por meio de livros de bibliotecas, materiais compartilhados ou anotações manuscritas. De acordo com os entrevistados, as principais dificuldades encontradas são:

● A transcrição manual de textos físicos, que demanda tempo e pode ser propensa a erros.

● A limitação de ferramentas disponíveis no mercado, como o Google Lens, que permite a digitalização de pequenos trechos por vez, não mantém a formatação original e pode apresentar imprecisões na transcrição.

● A dificuldade em transformar anotações manuscritas em documentos editáveis, impactando a organização e a revisão do conteúdo.

Conforme o relatado por uma das entrevistadas, um produto no mercado que realiza a digitalização de textos de forma gratuita é o Google Lens, porém, essa digitalização é somente de trechos pequenos do texto por vez, não mantém a formatação original e a transcrição nem sempre está igual ao texto original. Nesse contexto, decidimos desenvolver um leitor de texto que irá digitalizar e converter imagens de livros ou outros textos físicos, capturadas por uma webcam, para formatos como TXT, DOC ou PDF.

Contexto e Cenário de Aplicação

Nosso sistema visa principalmente simplificar a digitalização e a organização de conteúdos impressos, por meio de um sistema de processamento de imagens desenvolvido em linguagem

Python com a biblioteca morph.py e com a API OpenCV. Esse sistema será capaz de capturar imagens de textos físicos utilizando uma webcam e convertê-los para formatos digitais, de forma que auxilie no processo de estudo de estudantes, pesquisadores e profissionais que lidam frequentemente com material impresso. Dessa forma, as principais funções desse sistema serão:

● Captura de imagem de textos físicos.

● Processamento das imagens para extrair o texto.

● Exportar esse texto em formatos como TXT, DOC ou PDF.

● Manter a formatação original do texto.

Interatividade com o usuário

O usuário irá capturar uma foto do texto que deseja digitalizar, e o software criado irá converter em um documento de forma facilite a visualização, com possibilidade de fácil pesquisa, portabilidade e capacidade de rápida tradução.

Materiais e Métodos

Modelagem Funcional do SPI (MF)

O Sistema de processamento de Imagem (SPI) do programa foi modelado para transcrever os textos presentes na imagem da câmera, de forma que:

● Detecte diferentes blocos de texto

● Transcreva a imagem

● Salve em um arquivo .txt

Descrição da Implementação do SPI

A implementação foi realizada em Python, através da plataforma do Google Colab, utilizando principalmente a bilbioteca openCV para processamento da imagem. Foi também utilizada uma biblioteca de linguagem do python. O usuário captura a imagem desejada do livro ou outro item que deseje transcrever, o sistema processa a imagem de forma que facilite a leitura e então os blocos de textos são deetectados e transcritos.

Lista dos arquivos

● Projeto.ipynb - Com o projeto inteiro para execução ● Roteiro do Laboratório Experimental.ipynb - Com o roteiro e o código bem explicado para execução do usuário

Análise Técnica

O sistema foi capaz de realizar a captura de imagem e o processamento de forma correta, com a qual o texto estava de fácil leitura e detecção, entretanto a automação do reconhecimento de letras através da biblioteca pré-estabelecida do openCV, não obteve muito sucesso, sendo muito afetado pela quantidade de texto, claridade da imagem, entre outros aspectos. Desta forma, fica entendido que seria necessário uma reescrita do código realizando este reconhecimento de forma manual ou com outra biblioteca, que foge do escopo do curso.

Laboratório Experimental

Roteiro

O Roteiro do laboratório experimental pode ser encontrado abaixo.

Roteiro

O programa realizado possui o objetivo de permitir a fácil digitalização de livros físicos, de forma a facilitar seu manuseio e busca.

Ele utiliza técnicas de OCR (Reconhecimento Óptico de Caracteres) para digitalizar o conteúdo e facilitar sua leitura, busca e armazenamento. O sistema pode ser operado por qualquer usuário, mesmo sem conhecimentos técnicos, bastando seguir os passos descritos neste roteiro.

Obs.: Caso deseje o rápido manuseio, pode utilizar do atalho Ctrl+F9 para rodar todas as células.

Interface de Entrada e Saída:

- **Entrada**: Imagens contendo páginas de livros ou documentos físicos.
- **Saída**: Texto extraído da imagem, salvo em arquivos `.txt` no Google Drive ou exibido diretamente no notebook.

Conectando ao Google Drive

Primeiramente, precisará conceder a permissão de acesso ao Google Drive, no qual os arquivos serão armazenados. Basta rodar a célula abaixo, clicando na seta à esquerda.

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

Bibliotecas

Agora, realizaremos a instalação das bibliotecas necessárias para o programa.

In [ ]:

```
if 1:
    !pip install numpy
    !pip install opencv-python
    !pip install matplotlib
    !pip install scikit-image
    !pip install pillow
    !pip install pytesseract
    !pip install language-tool-python
    !sudo apt install tesseract-ocr
    !sudo apt install libtesseract-dev
    !sudo apt install tesseract-ocr-por
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(2.0.2)
Requirement already satisfied: opencv-python in
/usr/local/lib/python3.11/dist-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in
/usr/local/lib/python3.11/dist-packages (from opencv-python) (2.0.2)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
```

```
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib)
(1.17.0)
Requirement already satisfied: scikit-image in
/usr/local/lib/python3.11/dist-packages (0.25.2)
Requirement already satisfied: numpy>=1.24 in
/usr/local/lib/python3.11/dist-packages (from scikit-image) (2.0.2)
Requirement already satisfied: scipy>=1.11.4 in
/usr/local/lib/python3.11/dist-packages (from scikit-image) (1.14.1)
Requirement already satisfied: networkx>=3.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-image) (3.4.2)
Requirement already satisfied: pillow>=10.1 in
/usr/local/lib/python3.11/dist-packages (from scikit-image) (11.1.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in
/usr/local/lib/python3.11/dist-packages (from scikit-image) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in
/usr/local/lib/python3.11/dist-packages (from scikit-image) (2025.3.30)
Requirement already satisfied: packaging>=21 in
/usr/local/lib/python3.11/dist-packages (from scikit-image) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in
/usr/local/lib/python3.11/dist-packages (from scikit-image) (0.4)
Requirement already satisfied: pillow in
/usr/local/lib/python3.11/dist-packages (11.1.0)
Requirement already satisfied: pytesseract in
/usr/local/lib/python3.11/dist-packages (0.3.13)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.11/dist-packages (from pytesseract) (24.2)
Requirement already satisfied: Pillow>=8.0.0 in
/usr/local/lib/python3.11/dist-packages (from pytesseract) (11.1.0)
Requirement already satisfied: language-tool-python in
/usr/local/lib/python3.11/dist-packages (2.9.3)
Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (from language-tool-python) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from language-tool-python) (4.67.1)
Requirement already satisfied: psutil in
/usr/local/lib/python3.11/dist-packages (from language-tool-python) (5.9.5)
Requirement already satisfied: toml in /usr/local/lib/python3.11/dist-packages
(from language-tool-python) (0.10.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->language-tool-python)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests->language-tool-python)
(3.10)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->language-tool-python)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->language-tool-python)
(2025.1.31)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tesseract-ocr is already the newest version (4.1.1-2.1build1).
0 upgraded, 0 newly installed, 0 to remove and 34 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libtesseract-dev is already the newest version (4.1.1-2.1build1).
0 upgraded, 0 newly installed, 0 to remove and 34 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tesseract-ocr-por is already the newest version (1:4.00~git30-7274cfa-1.1).
0 upgraded, 0 newly installed, 0 to remove and 34 not upgraded.
```

In [ ]:

```
# download morph.py from drive
!pip install -U --no-cache-dir gdown --pre
!wget https://raw.githubusercontent.com/fzampirolli/morph/main/morph.py
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.11/dist-packages
(5.2.0)
Requirement already satisfied: beautifulsoup4 in
/usr/local/lib/python3.11/dist-packages (from gdown) (4.13.4)
Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from gdown) (3.18.0)
Requirement already satisfied: requests[socks] in
/usr/local/lib/python3.11/dist-packages (from gdown) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from gdown) (4.67.1)
Requirement already satisfied: soupsieve>1.2 in
/usr/local/lib/python3.11/dist-packages (from beautifulsoup4->gdown) (2.6)
Requirement already satisfied: typing-extensions>=4.0.0 in
/usr/local/lib/python3.11/dist-packages (from beautifulsoup4->gdown) (4.13.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (2.3.0)
```

```
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown)
(2025.1.31)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (1.7.1)
--2025-04-23 22:24:37--
https://raw.githubusercontent.com/fzampirolli/morph/main/morph.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 42991 (42K) [text/plain]
Saving to: 'morph.py'

morph.py            100%[===================>]  41.98K  --.-KB/s    in 0.01s

2025-04-23 22:24:37 (3.44 MB/s) - 'morph.py' saved [42991/42991]
```

In [ ]:

```python
import numpy as np
import cv2
from google.colab.patches import cv2_imshow #  display de imagem
from skimage import io
from PIL import Image
import matplotlib.pylab as plt
from morph import *
import pytesseract

pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'
custom_config = r'--oem 3 --psm 6'
```

Processamento de Imagem

Por conseguinte, rodaremos o código utilizado para processar a imagem, deixando-a de forma mais adequada para a leitura do texto

In [ ]:

```python
# 1. Aquisição de imagem (Webcam)
def capture_image(image_path):
  image = cv2.imread(image_path)
  # Converter a imagem para escala cinza
  image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  return image_gray
```

In [ ]:

```
# 2. Pré processamento
def process_image(image_gray):
  # melhoria de contraste (aqui poded ser útil para imagens com iluminação
irregular)
  clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
  contrast = clahe.apply(image_gray)

  # remoção de ruídos
  blurred = cv2.GaussianBlur(contrast, (5, 5), 0)  # Ideal para pré-OCR

  # Limiarização adaptativa com média ponderada gaussiana
  thresh = cv2.adaptiveThreshold(blurred, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 2)

   # Dilatamento para engrossar caracteres finos
   #aumentar a dilatção
  kernel = np.ones((1,1), np.uint8) #aumentar o kernel ou mudar o kerne
  dilated = cv2.dilate(thresh, kernel, iterations=1)

  return dilated
```

In [ ]:

```
def segment_text(image_processed):
    # 1. Pré-processamento adicional para segmentação
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    dilated = cv2.dilate(image_processed, kernel, iterations=1)

    # 2. Detecção de contornos (modo hierárquico para blocos aninhados)
    contours, hierarchy = cv2.findContours(dilated, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    # 3. Filtrar contornos por área e proporção (para descartar ruídos)
    min_area = 50
    max_area = 5000
    text_blocks = []
    segmented_img = cv2.cvtColor(image_processed, cv2.COLOR_GRAY2BGR)

    for i, cnt in enumerate(contours):
        area = cv2.contourArea(cnt)
        x, y, w, h = cv2.boundingRect(cnt)
        aspect_ratio = w / h

        # Critérios para ser considerado texto
        if (min_area < area < max_area) and (0.2 < aspect_ratio < 5):
```

```
            cv2.rectangle(segmented_img, (x, y), (x+w, y+h), (0, 255, 0), 2)
            text_blocks.append((x, y, w, h))  # Guarda coordenadas dos blocos

    # 4. Ordenar blocos da esquerda para direita, de cima para baixo
    text_blocks = sorted(text_blocks, key=lambda b: (b[1] // 20, b[0]))  #
Agrupa por linhas

    return segmented_img, text_blocks
```

In [ ]:

```
# 6. Pós-processamento (Correção de texto)
# Entrada: Texto reconhecido com possíveis erros.
# Saída: Texto refinado e corrigido em formato digital.

import language_tool_python
import re

def corrigir_texto(texto_ocr):
    tool = language_tool_python.LanguageTool('pt-BR')

    # Ajuste de formatação simples
    texto_formatado = texto_ocr.replace('\n', ' ')
    texto_formatado = re.sub(r'\s+', ' ', texto_formatado)

    # Correção ortográfica e gramatical
    texto_corrigido = tool.correct(texto_formatado)

    return texto_corrigido
```

MAIN

Por último, seguimos para o código, no qual será solicitado a permissão para utilização da sua webcam e, logo após, será necessário que tire a foto que deseja, para a transcrição do livro. Logo após o programa irá lhe dar o texto transcrito.

Procedimento Experimental

📘 Caso 1: Digitalização de uma página de livro ou documento físico fotografada com o celular

1. Execute o código abaixo.
2. Tire uma foto nítida de uma página de livro.
3. Salve a imagem no seu Google Drive.
4. Espere o programa finalizar a digitalização do texto dessa imagem.

📘 Caso 2: Digitalização de uma folha de caderno ou folha impressa

1. Execute o código abaixo.
2. Tire uma foto de uma folha de caderno ou de uma impressão.
3. Salve a imagem no seu Google Drive.
4. Espere o programa finalizar a digitalização do texto dessa imagem.

Resultados Esperados

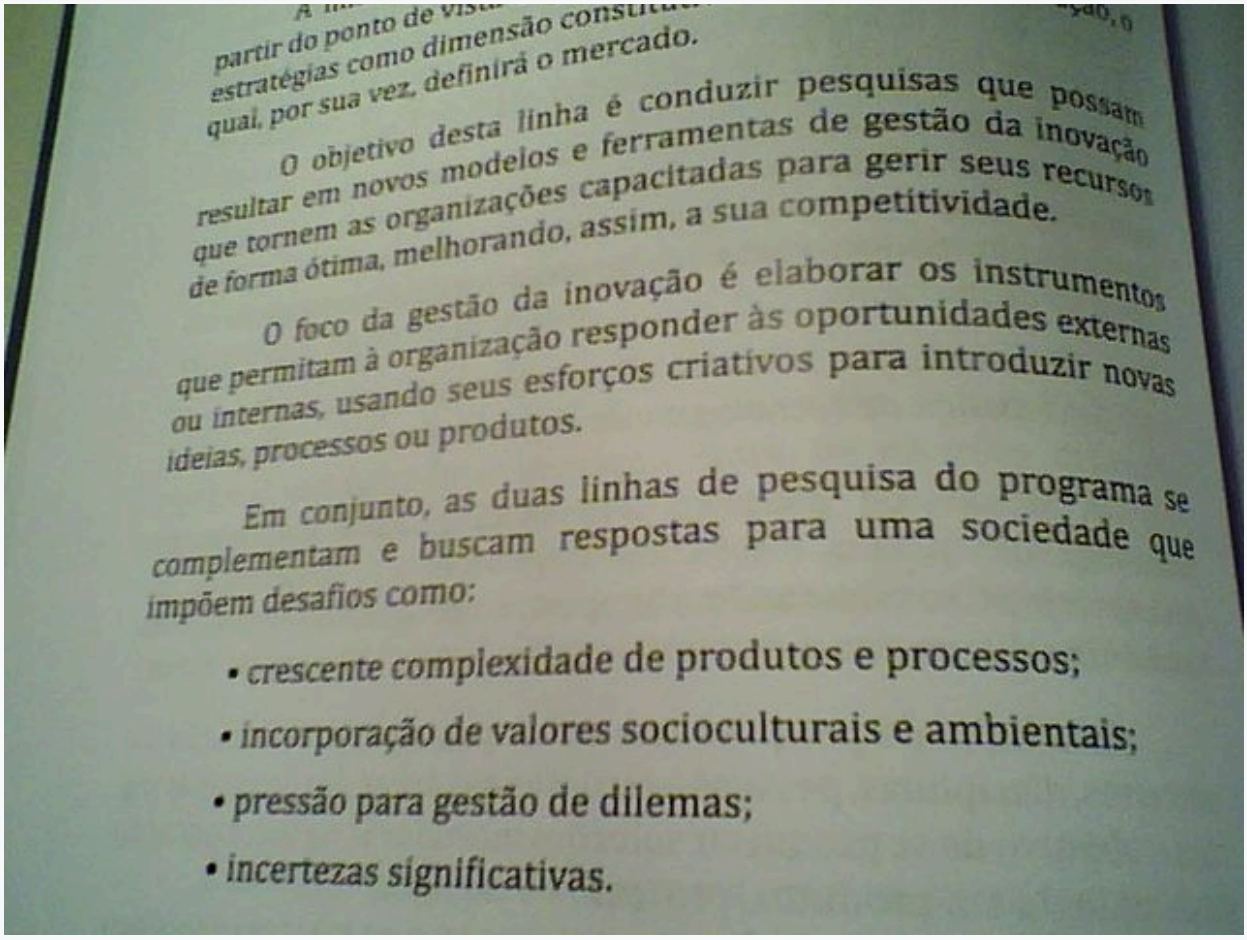
- O sistema deve retornar o texto contido na imagem com relativa precisão.
- Ruídos na imagem ou fotos tremidas podem comprometer a extração.

In [ ]:

```
# 7. Salvar no formato escolhido (txt, pdf, docx...)

if __name__ == "__main__":
  from IPython.display import Image
  import os

  image_path = "/content/drive/MyDrive/Colab Notebooks/"
  filename=os.path.join(image_path,'photo1.jpg')
  quality=0.8

  try:
    take_photo(filename, quality)

    print('Imagem gravada em {}'.format(filename))

    # Show the image which was just taken.
    display(Image(filename))

    image_gray = capture_image(filename)
    image_processed = process_image(image_gray)

    # Mostrar imagem processada
    plt.figure(figsize=(8, 6))
    plt.imshow(image_processed, cmap='gray')
    plt.title('Imagem Processada para OCR')
    plt.show()

    # Segmentação
    segmented_img, text_blocks = segment_text(image_processed)

    # Mostrar segmentação
    plt.figure(figsize=(8, 6))
    plt.imshow(segmented_img)
    plt.title('Blocos de Texto Detectados')
```

```
    plt.show()

    # Configuração otimizada para texto em parágrafos
    custom_config = r'--oem 3 --psm 6 -c
tessedit_char_whitelist="ABCDEFGHIJKLMNOPQRSTUVWXYZÇÃÂÊÁÉÍÓÚÀÜÖËÄ,.:;!?() "'

    # Reconhecimento do texto completo
    full_text = pytesseract.image_to_string(image_processed, lang='por',
config=custom_config)
    print("TEXTO RECONHECIDO:\n", full_text)

        # Reconhecimento por blocos (apenas se existirem blocos detectados)
    if len(text_blocks) > 0:
        print("\nRECONHECIMENTO POR BLOCOS:")
        print("-"*50)
        for i, (x, y, w, h) in enumerate(text_blocks):
            roi = image_processed[y:y+h, x:x+w]

            # Mostrar cada bloco (opcional)
            # plt.figure(), plt.imshow(roi, cmap='gray'), plt.title(f'Bloco
{i}'), plt.show()

            text = pytesseract.image_to_string(roi, lang='por', config='--psm
7')
            print(f"Bloco {i+1}: {text.strip()}")
        print("-"*50)

    else:
      print("\nNENHUM BLOCO DE TEXTO DETECTADO!")

    # Verificação final
    print("\nVERIFICAÇÃO:")
    print(f"Total de blocos detectados: {len(text_blocks)}")
    print(f"Tamanho da imagem processada: {image_processed.shape}")

    # Pós-processamento
    texto_corrigido = corrigir_texto(full_text)
    print("TEXTO CORRIGIDO:\n", texto_corrigido)
    with open("texto_corrigido.txt", "w", encoding="utf-8") as f:
      f.write(texto_corrigido)
      print("Texto salvo em 'texto_corrigido.txt'")


  except Exception as err:
    # Errors will be thrown if the user does not have a webcam or if they do
not
    # grant the page permission to access it.
    print(str(err))
```

Imagem capturada:



partir do ponto de vis
estratégias como dimensão constitu
qual, por sua vez, definirá o mercado.

O objetivo desta linha é conduzir pesquisas que possam resultar em novos modelos e ferramentas de gestão da inovação que tornem as organizações capacitadas para gerir seus recursos de forma ótima, melhorando, assim, a sua competitividade.

O foco da gestão da inovação é elaborar os instrumentos que permitam à organização responder às oportunidades externas ou internas, usando seus esforços criativos para introduzir novas ideias, processos ou produtos.

Em conjunto, as duas linhas de pesquisa do programa se complementam e buscam respostas para uma sociedade que impõem desafios como:

- crescente complexidade de produtos e processos;
- incorporação de valores socioculturais e ambientais;
- pressão para gestão de dilemas;
- incertezas significativas.

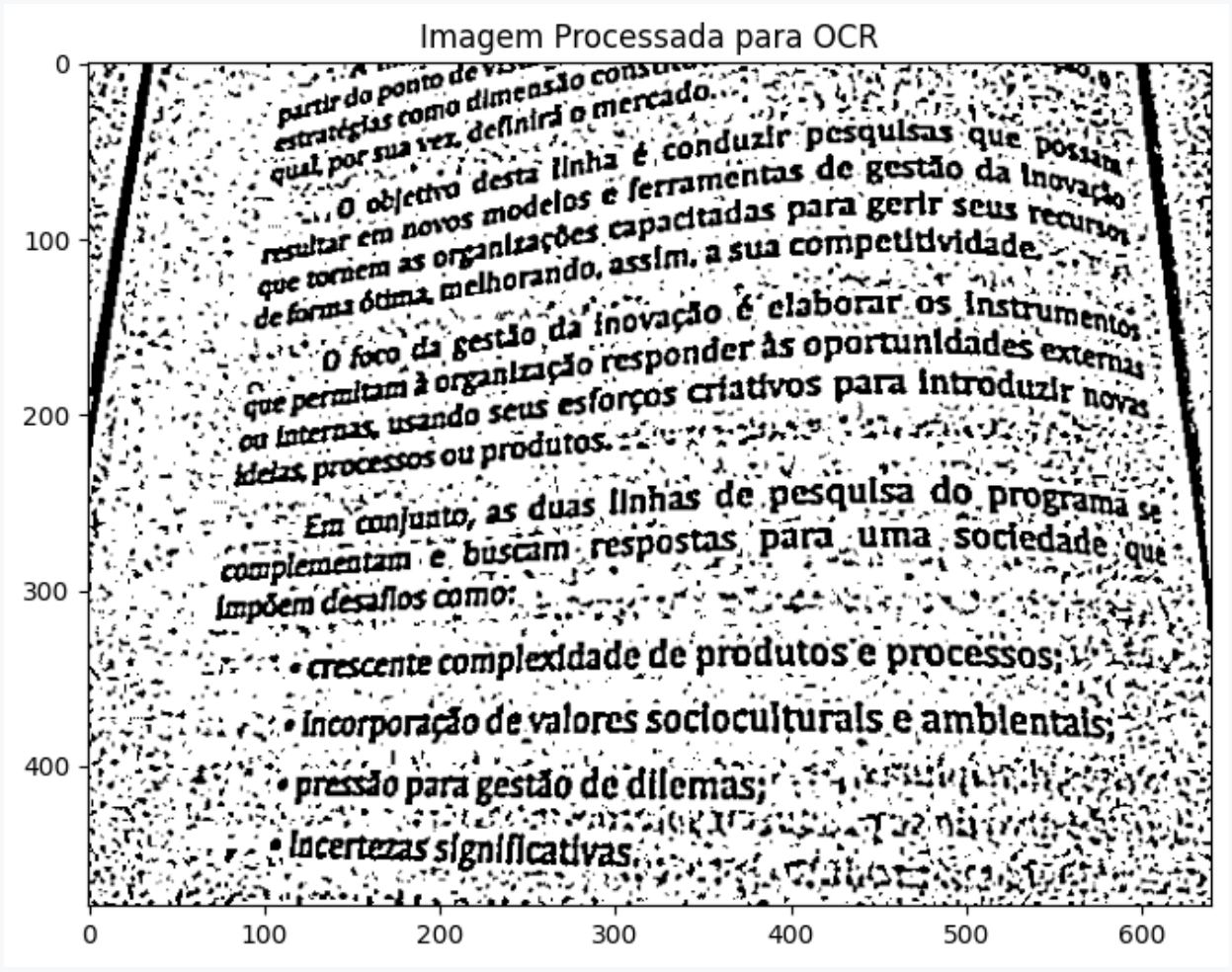
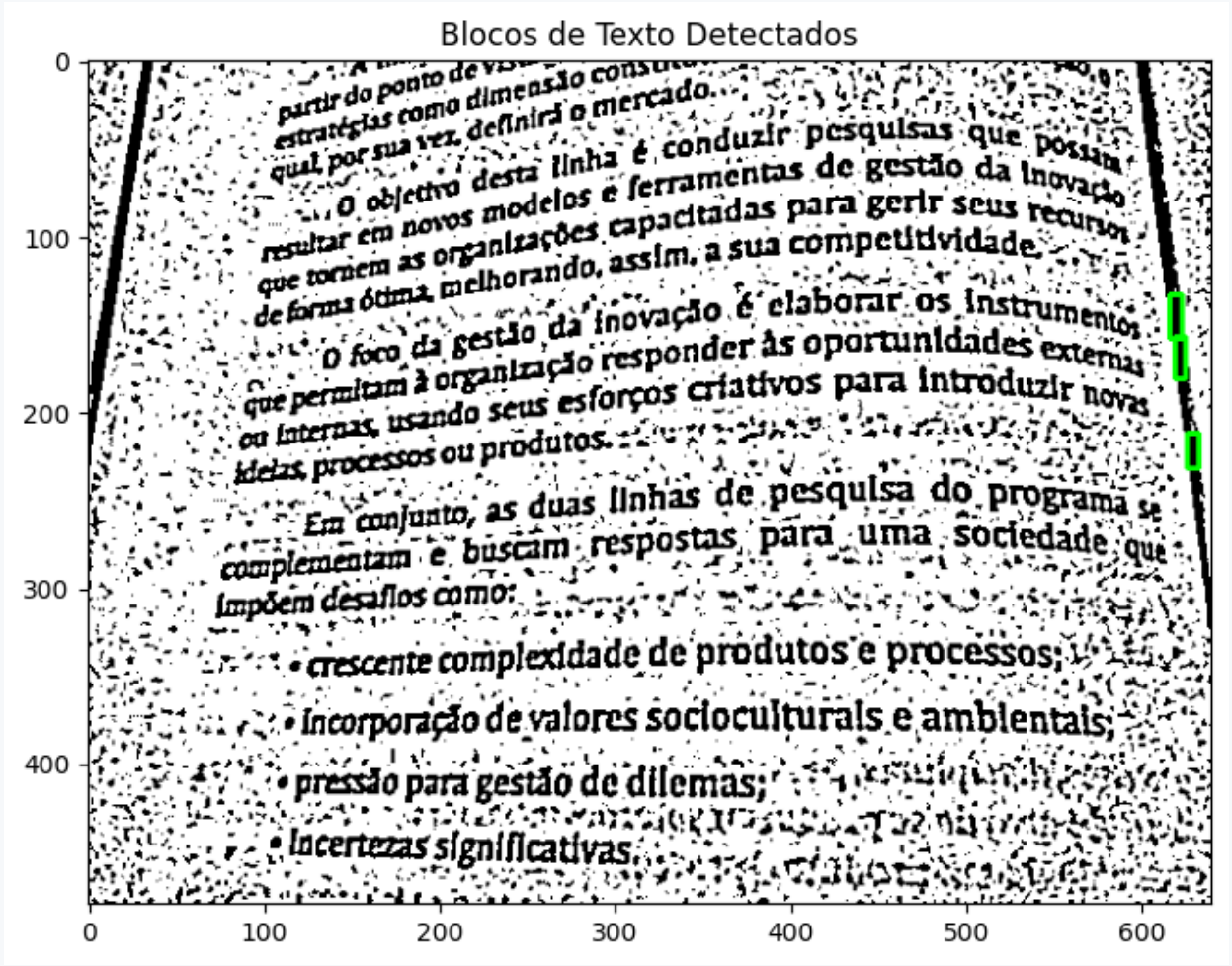


Imagem Processada para OCR
partir do ponto de
estratégias como dimensão
qual, por sua vez, definirá o mercado.
O objetivo desta linha é conduzir pesquisas que possam
resultar em novos modelos e ferramentas de gestão da inovação
que tornem as organizações capacitadas para gerir seus recursos
de forma ótima, melhorando, assim, a sua competitividade.
O foco da gestão da inovação é elaborar os instrumentos
que permitam à organização responder às oportunidades externas
ou internas, usando seus esforços criativos para introduzir novas
idéias, processos ou produtos.
Em conjunto, as duas linhas de pesquisa do programa se
complementam e buscam respostas para uma sociedade que
impõem desafios como:
• crescente complexidade de produtos e processos;
• incorporação de valores socioculturais e ambientais;
• pressão para gestão de dilemas;
• incertezas significativas.
0
100
200
300
400
500
600



TEXTO RECONHECIDO:

```
 EQ A O DO TE AÇÃO CONRO SS E ROS, ED A
PS S TO, SO OO DO E ASA ES O
A . EO O  MECÃ:,; VALE AC SDR RA
O RA C PSL AN NE
E A  J  O
S (O S E E, S P B SS F AE
FEIST E Ç, , U P:
E , OS CS
D: S RO ÇO É CIB  I? D
PIS A Á GB  A S :
E E G ÃO F I E
F EN SOS. É HE O A
NÉ OSS CU PRECE GO ELOS
OS, P ,  S Q ÃO
É B O S S  P UMA SO :
E G  D FESPOFGS, E SO .
E J COS GU, E SS
(ES C  CO J O  S;:,.
RD TIE DA PA
I H S I  ;
```

```
(S P RO
ERRO GBA DO MT LAR REG CTA NE
EA  S A TA OD
E TA ORE A TA DE RAD A PNI DDS EA


RECONHECIMENTO POR BLOCOS:
------------------------------------------------
Bloco 1: |
Bloco 2: l
Bloco 3: E
------------------------------------------------

VERIFICAÇÃO:
Total de blocos detectados: 3
Tamanho da imagem processada: (480, 640)
Detected java 11.0. LanguageTool requires Java >= 17 for version latest.
```

Questionário de Avaliação do Experimento

Responda as seguintes perguntas com base em sua experiência:

| Questão | Resposta (Nota de 0 a 5)

1. Você conseguiu executar o sistema sem dificuldades?
2. O texto foi extraído corretamente da imagem?
3. Você entendeu o funcionamento do programa?
4. O experimento foi útil para aprender sobre digitalização de textos?
5. Você recomendaria esse experimento a outra pessoa?

Enquete Subjetiva de Opinião (ESO)

(a) Perguntas abertas

1. O que mais gostou no sistema?
2. O que poderia ser melhorado?
3. Você teria sugestões de novas aplicações?

(b) Perguntas com escala (1 a 5)

| Pergunta | Nota (1 - Discordo totalmente / 5 - Concordo totalmente) |

1. O sistema é fácil de usar.
2. As instruções foram claras.
3. O resultado foi útil para meu aprendizado.
4. Me senti confiante operando o sistema.
5. Gostaria de ver mais experimentos semelhantes.

Análise dos Resultados dos Testes de Campo

Para o teste de campo, juntamos 11 pessoas, sendo pelo menos uma de cada grupo presente durante a aula, para testar o funcionamento do programa, rodando o código por sua integridade, tirando uma foto de sua escolha (disponibilizamos duas opções: uma folha impressa e um livro) e verificando a transcrição desse texto. Ao realizar os testes de campo, obtivemos as seguintes avaliações:

Avaliação do experimento	Pessoa 1	Pessoa 2	Pessoa 3	Pessoa 4	Pessoa 5	Pessoa 6	Pessoa 7	Pessoa 8	Pessoa 9	Pessoa 10	Pessoa 11
1. Você conseguiu executar o sistema sem dificuldades?	5	5	5	5	5	5	5	5	5	5	5
2. O texto foi extraído corretamente da imagem?	4	5	5	5	5	4	4	5	4	4	4
3. Você entendeu o funcionamento do programa?	5	5	5	5	5	5	5	5	5	5	5
4. Oexperimento foi útil para aprender sobre digitalização de textos?	5	5	5	5	5	5	5	5	5	5	5
5. Você recomendaria esse experimento a outra pessoa?	5	5	5	5	5	5	5	5	5	5	5

Podemos observar que todos usuários foram capazes de utilizar e entender o programa, observando possibilidades de uso deste em aplicações da vida real, no qual, somente na assertividade da extração do texto que houve falhas, porém ainda recebendo uma nota média de 4,45 de 5

Conclusões

Com a realização do trabalho, fomos capazes de realizar e atingir os objetivos do curso de processamento de imagens, realizando operações morfológicas para ampliar a qualidade/uso da imagem, de forma a obtermos uma imagem "limpa" em preto e branco que seja de fácil leitura para um programa de reconhecimento de textos e transcrição.

Entretanto, ao utilizar uma biblioteca para realizar este reconhecimento, não obtivemos grande sucesso, sendo incapaz de identificar letras/textos que estavam claras, ainda mais com o

processamento realizado. Portanto, entendemos que seria necessário realizar um programa mais complexo, além da biblioteca utilizada do openCV, para obtenção de uma maior taxa de sucesso na leitura dos textos.

Referências Bibliográficas

OPENCV - Python Tutorials. [S. l.], 8 jan. 2025. Disponível em: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html. Acesso em: 4 maio 2025.

Anexos

Projeto

Roteiro do Laboratório Experimental

Pesquisa de Usuário