2024-1-Projeto PIM

César Eduardo de Souza, Lucas Ziemann Ferreira

[1]Departamento de Ciência da Computação - Universidade do Estado de Santa Catarina (UDESC)
Caixa Postal 631 – 89.219-710 – Santa Catarina – SC – Brazil

cesar.souza@edu.udesc.br, lucas.ziemann@edu.udesc.br

Abstract. *This project demonstrates the application of Fourier transform techniques, including masking and structural similarity measurements, using Python and various image processing libraries.*

Resumo. *Este projeto demonstra a aplicação de técnicas de transformada de Fourier, incluindo mascaramento e medições de similaridade estrutural, utilizando linguagem Python e suas bibliotecas de processamento de imagens.*

1. Sobre o trabalho

1.1. Introdução

Este trabalho envolve o estudo da aplicação da transformada de Fourier em imagens, técnicas de mascaramento e a avaliação da similaridade estrutural (SSIM), conforme estudado em sala. A implementação é feita na linguagem Python, utilizando diversas bibliotecas de processamento de imagem como OpenCV, PIL, e skimage, mencionadas na seção de requisitos.

1.2. Requisitos

1. Python 3
2. Numpy
3. OpenCV
4. Matplotlib
5. PIL
6. scikit-image

2. Código principal

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from PIL import Image, ImageEnhance
from skimage.io import imread, imshow
from skimage.color import rgb2hsv, rgb2gray, rgb2yuv
from skimage import color, exposure, transform
from skimage.exposure import equalize_hist,
 ↪  equalize_adapthist
from skimage.metrics import structural_similarity

```

```
def fourier_masker_ver(image, i):
    font_size = 8
    dark_image_grey_fourier =
    ↪  np.fft.fftshift(np.fft.fft2(image))
    w, h = image.size

    # Linha Vertical
    dark_image_grey_fourier[:(int(h/2)-10),
    ↪  int(w/2)-2:int(w/2)+2] = i
    dark_image_grey_fourier[-(int(h/2)-10):,
    ↪  int(w/2)-2:int(w/2)+2] = i

    # Linha horizontal
    dark_image_grey_fourier[int(h/2)-2:int(h/2)+2,
    ↪  :(int(w/2)-10)] = i
    dark_image_grey_fourier[int(h/2)-2:int(h/2)+2,
    ↪  -(int(w/2)-10):] = i

    reversed_fourier =
    ↪  abs(np.fft.ifft2(dark_image_grey_fourier))
    ssimOG = structural_similarity(reversed_fourier,
    ↪  np.array(image), data_range=255)

    fig, ax = plt.subplots(3,3,figsize=(12,8))
    ax[0,1].imshow(np.log(abs(dark_image_grey_fourier)),
    ↪  cmap='gray')
    ax[0,1].set_title('Masked Fourier', fontsize =
    ↪  font_size)
    ax[0,1].axis('off')

    ax[0,0].imshow(image, cmap = 'gray')
    ax[0,0].set_title('Greyscale Image', fontsize =
    ↪  font_size)
    ax[0,0].axis('off')

    transformed_eq_hist =
    ↪  increase_brightness_equalizing_histogram(reversed_fourier)
    ssim_eq_hist =
    ↪  structural_similarity(transformed_eq_hist,
    ↪  np.array(image), data_range=255)
    ax[2,0].imshow(transformed_eq_hist, cmap='gray')
    ax[2,0].set_title('Hist Eq: '+ str(ssim_eq_hist),
    ↪  fontsize = font_size)
    ax[2,0].axis('off')

    transformed_eq_hist_adapted =
    ↪  increase_brightness_equalizing_histogram_adapted(reversed_fouri
```

```
43      ssim_eq_hist_adapted =
        ↪  structural_similarity(transformed_eq_hist_adapted,
        ↪  np.array(image), data_range=255)
44      ax[1,0].imshow(transformed_eq_hist_adapted,
        ↪  cmap='gray')
45      ax[1,0].set_title('Adapt Hist Eq: '+
        ↪  str(ssim_eq_hist_adapted), fontsize = font_size)
46      ax[1,0].axis('off')
47
48      (transformed_pil, ssim_pil) =
        ↪  increase_brightness_pil_enhancer(reversed_fourier)
49      ax[1,1].imshow(transformed_pil, cmap='gray')
50      ax[1,1].set_title('Pil Enhance: '+ str(ssim_pil),
        ↪  fontsize = font_size)
51      ax[1,1].axis('off')
52
53      (transformed_cv, ssim_cv) =
        ↪  increase_brightness_cv2(reversed_fourier)
54      ax[1,2].imshow(transformed_cv, cmap='gray')
55      ax[1,2].set_title('CV2 Enhance: '+ str(ssim_cv),
        ↪  fontsize = font_size)
56      ax[1,2].axis('off')
57
58      ax[0,2].imshow(reversed_fourier, cmap='gray')
59      ax[0,2].set_title('Fourier: '+ str(ssimOG), fontsize =
        ↪  font_size)
60      ax[0,2].axis('off')
61
62      print("SSIMs:\n")
63      print("{:<30} {:<10.6f}".format("Original:", ssimOG))
64      print("{:<30} {:<10.6f}".format("Equalized Histogram:",
        ↪  ssim_eq_hist))
65      print("{:<30} {:<10.6f}".format("Equalized Adapted
        ↪  Histogram:", ssim_eq_hist_adapted))
66      print("{:<30} {:<10.6f}".format("PIL Enhancement:",
        ↪  ssim_pil))
67      print("{:<30} {:<10.6f}".format("CV2 Enhancement:",
        ↪  ssim_cv))
68
69
70  def increase_brightness_equalizing_histogram(dark_image):
71      lighter_image = (dark_image - np.min(dark_image)) /
        ↪  (np.max(dark_image) - np.min(dark_image))
72      lighter_image = lighter_image * 255
73      lighter_image =
        ↪  equalize_hist(lighter_image.astype(np.uint8))
```

```
74      return lighter_image
75
76  def
    ↪  increase_brightness_equalizing_histogram_adapted(dark_image):
77      lighter_image = (dark_image - np.min(dark_image)) /
        ↪  (np.max(dark_image) - np.min(dark_image))
78      lighter_image = lighter_image * 255
79      lighter_image =
        ↪  equalize_adapthist(lighter_image.astype(np.uint8))
80      return lighter_image
81
82  def increase_brightness_pil_enhancer(dark_image):
83      if isinstance(dark_image, np.ndarray):
84          dark_image =
            ↪  Image.fromarray(dark_image.astype('uint8'))
85      i = 0.1
86      prev_ssim = -1
87      img_r_np = np.array(img_r)
88      while i <= 1.0:
89          enhancer = ImageEnhance.Brightness(dark_image)
90          enhanced_image = enhancer.enhance(i)
91          enhanced_image_np = np.array(enhanced_image)
92          current_ssim =
            ↪  structural_similarity(enhanced_image_np,
            ↪  img_r_np, multichannel=True)
93          if current_ssim < prev_ssim:
94              break
95          prev_ssim = current_ssim
96          i += 0.1
97      return enhanced_image, current_ssim
98
99  def increase_brightness_cv2(dark_image):
100     path_r2 = "assets/folhas1_Reticulada.jpg"
101     img_r2 = cv2.imread(path_r2, cv2.IMREAD_GRAYSCALE)
102     i = 0.1
103     prev_ssim = -1
104     while i <= 1.0:
105         lighter_image = cv2.convertScaleAbs(dark_image,
            ↪  alpha=i)
106         lighter_image = cv2.equalizeHist(lighter_image)
107         current_ssim = structural_similarity(lighter_image,
            ↪  img_r2)
108         if current_ssim < prev_ssim:
109             break
110         prev_ssim = current_ssim
111         i += 0.1
```

```
    return lighter_image, current_ssim

path_r = "assets/folhas1_Reticulada.jpg"
img_r = Image.open(path_r).convert('L')
path = "assets/folhas1.jpg"
img = Image.open(path).convert('L')
img_np = np.array(img)
fourier_masker_ver(img, 0.000000000001)
```

3. Resultados

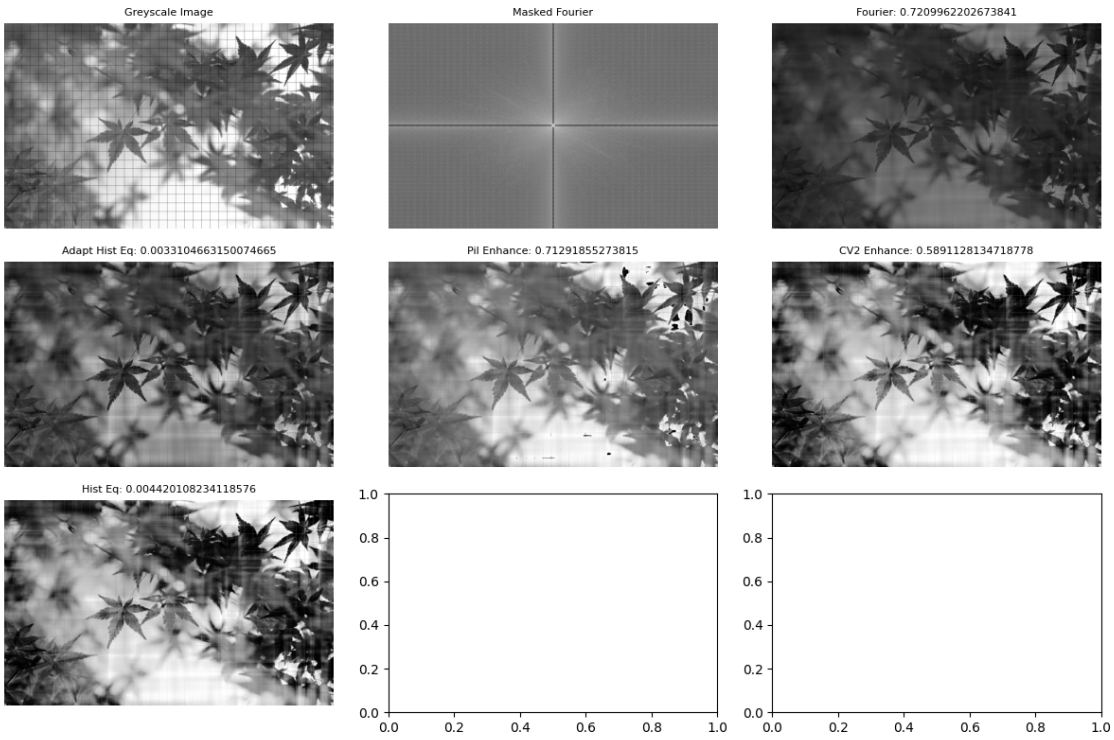

Figura 1. Imagem original: Figure_1.png

4. Conclusão

O código fornecido realiza uma série de transformações em uma imagem usando a transformada de Fourier e técnicas de mascaramento, e avalia a similaridade estrutural das imagens transformadas. A visualização das etapas permite um entendimento claro das alterações feitas na imagem e sua comparação com a imagem original através da métrica SSIM.