فوریه

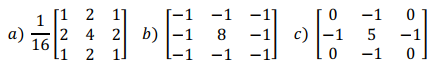
سحر محمدی

|  |  |  |
| --- | --- | --- |
| اطلاعات گزارش |  | چکیده |
| **تاریخ: دی 1401** |  | در این گزارش، نحوه انجام تمرینات مربوط به فصل تبدیل فوریه توضیح داده شده است. همچنین در ادامه، به مشاهده و بررسی نتایج پرداخته شده است. |
| **واژگان كليدي:**  فوریه  تبدیل  مقدار  فاز  فیلتر |  |

1-توصیف تکنیکال

مسئله مورد بررسی اول(4.1.1)، شامل 3 فیلتر است که باید تبدیل فوریه این فیلترها بدست آید. فیلترها روی تصویر اعمال شود و در نهایت تشخیص دهیم هر کدام از بین(نرم کردن، تشخیص لبه و یا تقویت لبه) مربوط به کدام مورد می‌باشد.

فیلترها بصورت زیر است:

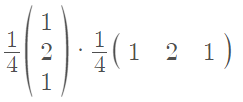


همچنین تصویر مورد بررسی، تصویر Lena بصورت سیاه‌سفید است که آن را در img ریخته‌ایم.



لازم است تبدیل فوریه تصویر بدست آید. همچنین فوریه فیلتر. مگنیتودها در هم ضرب داخلی شوند و داخل متغیر final\_mag ریخته می‌شود. و در اخر بر روی خروجی) همان final\_mag) و فاز تصویر اولیه Lena، معکوس تبدیل فوریه انجام شود.

فیلتر b و c تفکیک‌پذیر نیستند، پس تبدیل فوریه بصورت مستقیم انجام میشود. اما فیلتر a تفکیک‌پذیر است و میتوان آن را بصورت زیر نوشت:



مسئله 4.1.2، در قسمت می‌خواهیم تبدیل فوریه دو بعدی گسسته تصاویر سیاه‌سفید Lena, Barbara, F16, and Baboon را بدست آوریم. در نهایت magnitude ها را با و بدون شیفت دادن و همچنین با و بدون لگاریتم گرفتن نمایش دهیم.

مسئله 4.2.1،

مسئله 4.2.2،

2-نتایج

مسئله 4.1.1، خروجی فیلتر a بصورت زیر است:



که تصویر تار یا smooth شده.

خروجی فیلتر b:



واضح است که این فیلتر مربوط به تشحیص لبه(edge detection) می‌باشد.

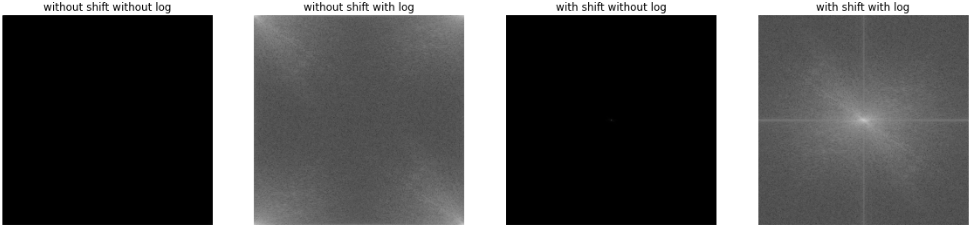
همچنین خروجی فیلتر c:

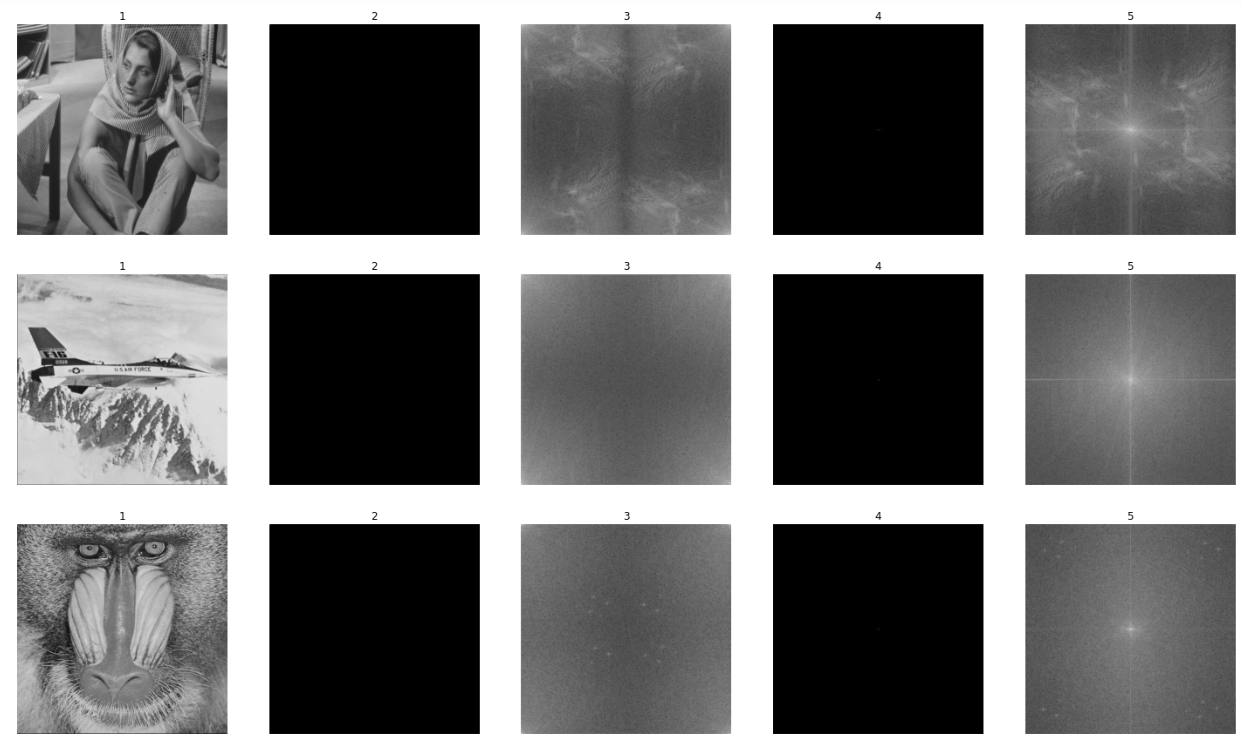


که نشان می‌دهد فیلتر مربوط به تقویت لبه می‌باشد.

4.1.2:







3- ضمایم

کدها:

import cv2

import numpy as np

import matplotlib.pyplot as plt

from math import floor, ceil, sqrt

import warnings

warnings.filterwarnings("ignore")

# In[2]:

img = cv2.imread("Images/4/Lena.bmp", cv2.IMREAD\_GRAYSCALE)

# In[3]:

def imshow(\*args, figsize=25, to\_rgb=True, title=None, fontsize=12):

if isinstance(figsize, int):

figsize = (figsize, figsize)

images = args[0] if type(args[0]) is list else list(args)

if to\_rgb:

images = list(map(lambda x: cv2.cvtColor(x, cv2.COLOR\_BGR2RGB), images))

if title is not None:

assert len(title) == len(images), "Please provide a title for each image."

plt.figure(figsize=figsize)

for i in range(1, len(images)+1):

plt.subplot(1, len(images), i)

if title is not None:

plt.title(title[i-1], fontsize=fontsize)

plt.imshow(images[i-1])

plt.axis('off')

# In[4]:

def fourier\_transform(image):

fft = np.fft.fftshift(np.fft.fft2(image))

magnitude, phase = np.abs(fft), np.angle(fft)

return magnitude, phase

# In[5]:

def inverse\_fourier\_transform(magnitude, phase):

real = np.cos(phase) \* magnitude

imag = np.sin(phase) \* magnitude

fft = real + (1j \* imag)

image = np.fft.ifft2(np.fft.ifftshift(fft))

image = np.real(image)

image = image.clip(min=0, max=255).astype('uint8')

return image

# In[6]:

def normalize(image, rmin=0, rmax=255, to\_uint=True):

norm = ((image - image.min()) / (image.max() - image.min())) \* (rmax - rmin) + rmin

if to\_uint:

norm = norm.astype('uint8')

return norm

# In[7]:

def visualize\_magnitude(magnitude):

return normalize(20 \* np.log(magnitude))

# In[8]:

imshow(img)

# # 4.1.1

# In[9]:

mag, phase = fourier\_transform(img)

mag\_visual = visualize\_magnitude(mag)

# In[10]:

imshow(mag\_visual)

# In[11]:

#create filtera

a = [[1/16,2/16,1/16],[2/16,4/16,2/16],[1/16,2/16,1/16]]

filtera = np.array(a)

h, w = filtera.shape

th, tw = img.shape # target width & height

y\_start = ceil(th//2 - h/2)

y\_end = floor(th//2 + h/2)

x\_start = ceil(tw//2 - w/2)

x\_end = floor(tw//2 + w/2)

#resize the filter

filtera\_resized = np.zeros(img.shape, dtype='float64')

filtera\_resized[y\_start:y\_end+1 , x\_start:x\_end+1] = filtera

#find the filter magnitude

filtera\_mag, \_ = fourier\_transform(filtera\_resized)

# Apply filter (calculate final magnitude)

final\_mag = mag \* filtera\_mag

# Reconstruct the image

reconstructed = inverse\_fourier\_transform(final\_mag, phase)

# Plot

img\_mag\_vis = visualize\_magnitude(mag)

final\_mag\_vis = visualize\_magnitude(final\_mag)

# titles = ['Original Image', 'Image Magnitude', 'Final Magnitude', 'Result']

# imshow(img, img\_mag\_vis, final\_mag\_vis, reconstructed, figsize=30, title=titles)

imshow(img, reconstructed, title=['1','2'])

# In[12]:

#create filterb

b = [[-1,-1,-1],[-1,8,-1],[-1,-1,-1]]

filterb = np.array(b)

h, w = filterb.shape

th, tw = img.shape # target width & height

y\_start = ceil(th//2 - h/2)

y\_end = floor(th//2 + h/2)

x\_start = ceil(tw//2 - w/2)

x\_end = floor(tw//2 + w/2)

#resize the filter

filterb\_resized = np.zeros(img.shape, dtype='float64')

filterb\_resized[y\_start:y\_end+1 , x\_start:x\_end+1] = filterb

#find the filter magnitude

filterb\_mag, \_ = fourier\_transform(filterb\_resized)

# Apply filter (calculate final magnitude)

final\_mag = mag \* filterb\_mag

# Reconstruct the image

reconstructed = inverse\_fourier\_transform(final\_mag, phase)

# Plot

img\_mag\_vis = visualize\_magnitude(mag)

final\_mag\_vis = visualize\_magnitude(final\_mag)

imshow(img, reconstructed, title=['1','2'])

# In[13]:

#create filterc

c = [[0,-1,0],[-1,5,-1],[0,-1,0]]

filterc = np.array(c)

h, w = filterc.shape

th, tw = img.shape # target width & height

y\_start = ceil(th//2 - h/2)

y\_end = floor(th//2 + h/2)

x\_start = ceil(tw//2 - w/2)

x\_end = floor(tw//2 + w/2)

#resize the filter

filterc\_resized = np.zeros(img.shape, dtype='float64')

filterc\_resized[y\_start:y\_end+1 , x\_start:x\_end+1] = filterc

#find the filter magnitude

filterc\_mag, \_ = fourier\_transform(filterc\_resized)

# Apply filter (calculate final magnitude)

final\_mag = mag \* filterc\_mag

# Reconstruct the image

reconstructed = inverse\_fourier\_transform(final\_mag, phase)

# Plot

img\_mag\_vis = visualize\_magnitude(mag)

final\_mag\_vis = visualize\_magnitude(final\_mag)

imshow(img, reconstructed, title=['1','2'])

# ## 4.1.2

# In[14]:

def fourier\_transform\_woutsh(image):

fft = np.fft.fft2(image)

magnitude, phase = np.abs(fft), np.angle(fft)

return magnitude, phase

# In[15]:

lena = cv2.imread("Images/4/Lena.bmp", cv2.IMREAD\_GRAYSCALE)

bar = cv2.imread("Images/4/Barbara.bmp", cv2.IMREAD\_GRAYSCALE)

f = cv2.imread("Images/4/F16.bmp", cv2.IMREAD\_GRAYSCALE)

bab = cv2.imread("Images/4/Baboon.bmp", cv2.IMREAD\_GRAYSCALE)

# In[16]:

mag1, phase1 = fourier\_transform(lena)

mag2, phase2 = fourier\_transform(bar)

mag3, phase3 = fourier\_transform(f)

mag4, phase4 = fourier\_transform(bab)

mag12, phase12 = fourier\_transform\_woutsh(lena)

mag\_normal12 = normalize(mag12)

mag\_visual12 = visualize\_magnitude(mag12)

mag22, phase22 = fourier\_transform\_woutsh(bar)

mag\_normal22 = normalize(mag22)

mag\_visual22 = visualize\_magnitude(mag22)

mag32, phase32 = fourier\_transform\_woutsh(f)

mag\_normal32 = normalize(mag32)

mag\_visual32 = visualize\_magnitude(mag32)

mag42, phase42 = fourier\_transform\_woutsh(bab)

mag\_normal42 = normalize(mag42)

mag\_visual42 = visualize\_magnitude(mag42)

mag\_normal1 = normalize(mag1)

mag\_visual1 = visualize\_magnitude(mag1)

mag\_normal2 = normalize(mag2)

mag\_visual2 = visualize\_magnitude(mag2)

mag\_normal3 = normalize(mag3)

mag\_visual3 = visualize\_magnitude(mag3)

mag\_normal4 = normalize(mag4)

mag\_visual4 = visualize\_magnitude(mag4)

imshow(lena, mag\_normal12, mag\_visual12, mag\_normal1, mag\_visual1, title=['org', 'without shift without log', 'without shift with log', 'with shift without log', 'with shift with log'])

imshow(bar, mag\_normal22, mag\_visual22, mag\_normal2, mag\_visual2, title=['1', '2', '3', '4', '5'])

imshow(f, mag\_normal32, mag\_visual32, mag\_normal3, mag\_visual3, title=['1', '2', '3', '4', '5'])

imshow(bab, mag\_normal42, mag\_visual42, mag\_normal4, mag\_visual4, title=['1', '2', '3', '4', '5'])

# ## 4.2.2

# In[17]:

bar = cv2.imread("Images/4/Barbara.bmp", cv2.IMREAD\_GRAYSCALE)

# In[18]:

def prog():

mag, phase = fourier\_transform(bar)

reconstructed = inverse\_fourier\_transform(final\_mag, phase)

return reconstructed

مراجع

[1] [FUM-ComputerVision/CV\_S03.ipynb at master · alifarrokh/FUM-ComputerVision (github.com)](https://github.com/alifarrokh/FUM-ComputerVision/blob/master/CV_S03.ipynb)