Министерство образования и науки Российской Федерации

Федеральное государственное автономное

Образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский   
государственный университет им. Н.И. Лобачевского»

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**Обработка спектра Фурье**

**Выполнил:**

Утюгов Денис  
Студент группы 381906-2

**Преподаватель:**

Александра Александровна Гетманская

Нижний Новгород

2021

**Содержание:**

1. Введение
2. Постановка учебно-практической задачи
3. Метод решения задачи
4. Неудачные методы
5. Список литературы
6. Приложение

**Введение**

Дискретное преобразование Фурье нередко используется для операции свертки. Спектральный анализ же можно использовать для выявления характерных частот в целях подавления шума. Зная уже это, можно предположить, что с помощью спектрального анализа мы можем найти периодические шумы, которые в следствии можно удалить.

**Постановка учебно-практической задачи**

В рамках лабораторной работы ставится задача удаления шума, путем использования преобразования Фурье. Необходимо поставить эксперименты и определить, какие способы ближе к поставленной задаче.

**Метод решения задачи**

Изучив задачу, я пришел к выводу, что наибольшее внимание стоит уделить спектральному анализу.

Так как в полученном спектре могут быть большие числа, для иллюстрации спектра логично использовать логарифм, т.е. представлять спектр в виде np.log(1+magnitude), где magnitude – наш спектр. Отрицательные числа покажут значения, близкие к 0, положительные будут на порядки меньше, что нам и нужно.

Для работы со спекторм, я сконцентрировал частоты к центру с помощью функции fftshift(). Таким образом мы можем попробовать убрать кусок от центра, избавившись от наибольшего скопления (Рис. 1)

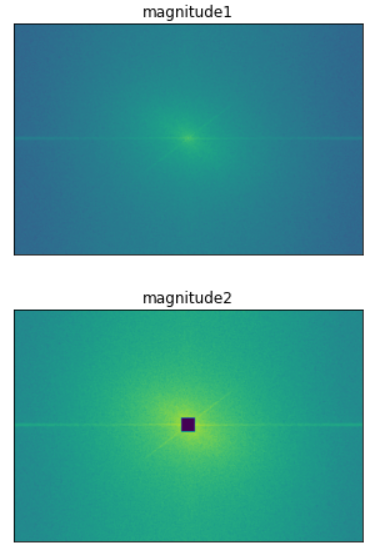


Рисунок 1. Убираем скопление частот.

Посмотрев спектры разных изображений, я решил работать с этим изображением (Рис. 2.)

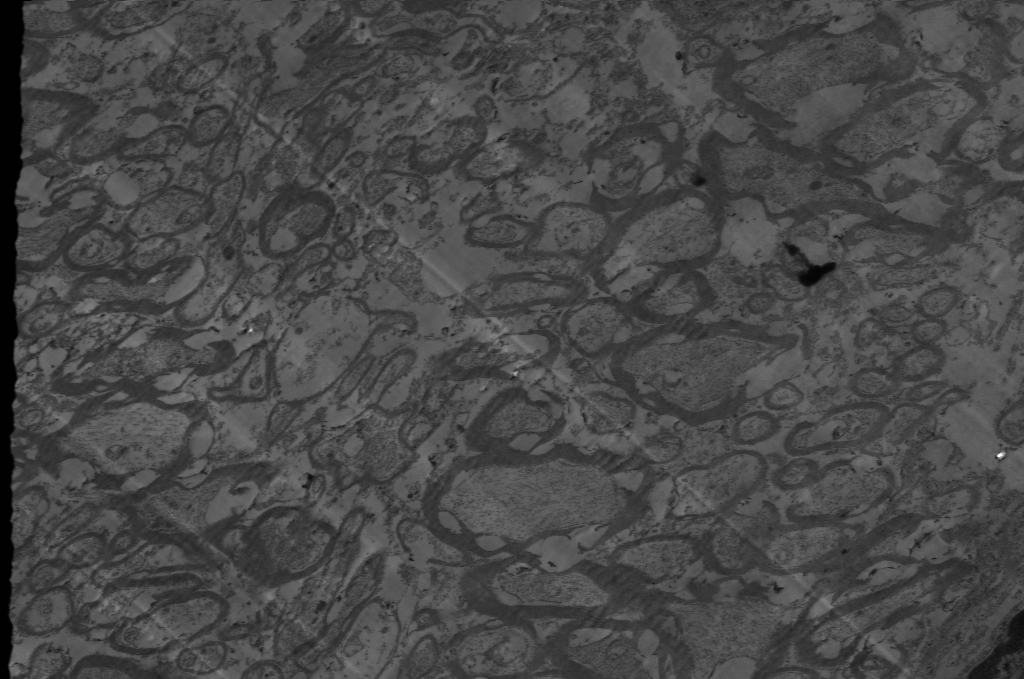


Рисунок 2. Выбранное изображение (Bbain\_03\_55.png)

Такой выбор я сделал исходя из того, что вижу некоторые характерные диагональные линии в спектре. Выдвигаю предположение, что стоит попробовать вырезать их.

Но сначала посмотрим, что случится с изображением после того, как мы вырежем центр нашего спектра. Подбирая размер квадрата, я пришел к наилучшему, на мой взгляд, варианту (Рис. 3).

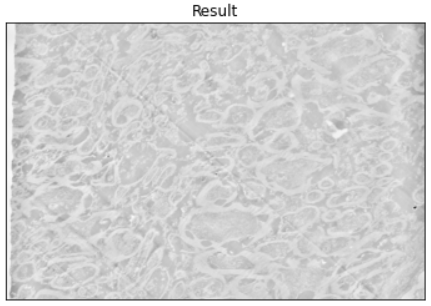
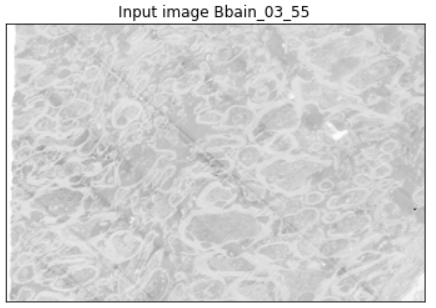


Рисунок 3. Результат после удаления частот из центра.

Мы видим следующие изменения: пропали тени на линиях, но сами лини никуда не делись. Поэтому мы возвращаемся к выдвинутому предположению: пробуем убрать горизонтальную линию из спектра (Рис. 4).

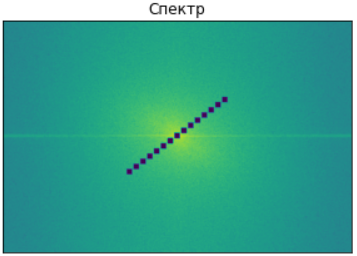


Рисунок 4. Спектр с удалением горизонтальной полосы.

Осталось применить обратное преображение Фурье и посмотреть на результат (Рис. 5).

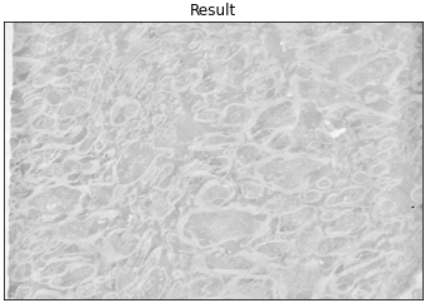


Рисунок 5. Результат лабораторной.

Таким образом мы полностью избавились от полос на этом изображении – от них не осталось и следа. Значит нам удалось достичь поставленной задачи.

**Неудачные попытки**

Кроме этого метода я пытался найти частоты 98 перцентиля (т.е. 2% высоких, или, напротив, 98 низких, а также наоборот), но особого результата это не давало. Алгоритм был следующий:

С помощью magnitude<np.percentile(magnitude,98) мы получаем массив с True и False. По размеру он равен фотографии. Далее я применял обратное преобразование Фурье на полученный список и умножал его на исходный спектр.

Таким образом определенная группа (высоких или низких) частот должна была обратиться в ноль.

Все попытки приводили к порче изображения – оно размывалось, удалялись не те часты или вообще изменялось до неузнаваемости.

Оглядываясь назад, теперь я могу предположить, что это могло сработать, но мы бы заодно обращали в ноль горизонтальную линию, что, скорее всего, привело бы к потере каких-то важных фрагментов. Следует проверить данную гипотезу (Рис. 6).

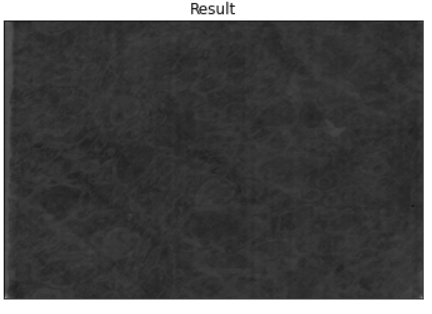
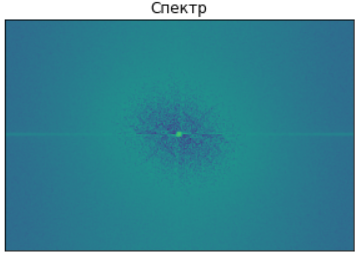


Рисунок 6. Полученный спектр и результат.

Полученный результат не очень радует. Диагональные линии менее заметны, но и все изображение стало темнее. Удаления диагональных линий хуже, чем в предыдущем методе.

**Заключение**

В ходе выполнения лабораторной работы поставленная задача была выполнена – лишние линии были удалены с изображения. Всего было использовано два метода, один из которых полностью решил проблему.

**Литература**

1. Приложенная к заданию методичка «Преобразование Фурье».
2. Сайт «Polybook» статья [«Спектральный анализ»](http://www.polybook.ru/comma/1.6.pdf).
3. Wikipedia статья [«Дискретное преобразование Фурье»](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A4%D1%83%D1%80%D1%8C%D0%B5).
4. Книга «Элегантный SciPy» Хуан Нуньес-Иглесиас, Штефан ван дер Уолт, Харриет Дэшноу.

**Приложение**

import cv2

import sys

import numpy as np

import matplotlib.pyplot as plt

import os

def CalcOfDamageAndNonDamage (image\_name):

'''Использование методов watershed(), выделение марекров.'''

image = image\_name

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (7,7))

image\_erode = cv2.erode(image, kernel)

hsv\_img = cv2.cvtColor(image\_erode, cv2.COLOR\_BGR2HSV)

#gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

#mask = cv2.compare(gray, 5, cv2.CMP\_LT)

#img[mask > 0] = 255

#Создаем массив из нулей - маркеры, после выделяем зоны (фон - 1 и листик - 255).

markers = np.zeros((image.shape[0], image.shape[1]), dtype = "int32")

markers[40:170, 90:140] = 255 #Решили, что лист всегда будет в центре

markers[236:255, 0:20] = 1

markers[0:20, 0:20] = 1

markers[0:20, 236:255] = 1

markers[236:255, 235:255] = 1

#markers[mask > 0] = 2 #Попытка отделить тень в отдельный класс

#Применяем watershed()

leafs\_area\_BGR = cv2.watershed(image\_erode, markers)

healthy\_part = cv2.inRange(hsv\_img, (36, 25, 25), (86, 255, 255)) #Параметры hsv вроде бы выставлены корректно и менять их смыла нет

ill\_part = leafs\_area\_BGR - healthy\_part

shadow = cv2.inRange(hsv\_img, (0, 0, 0, 0), (180, 255, 15, 0))

#Создаем маску

mask = np.zeros\_like(image, np.uint8 )

mask[leafs\_area\_BGR > 1] = (255,0,255)

mask[ill\_part>1] = (0,0,255)

mask[shadow > 1] = (0,0,0)

return mask

img = cv2.imread('8.jpg')

def NlMeans(img):

'''Шумоподавляющий фильтр'''

b,g,r = cv2.split(img)

rgb\_img = cv2.merge([r,g,b])

dst = cv2.fastNlMeansDenoisingColored(img, None, 7, 10, 7, 21) #img, dst, h, tWS, sWS. Есть смысл рассматривать h

b,g,r = cv2.split(dst)

rgb\_dst = cv2.merge([r,g,b])

return rgb\_dst

rgb\_dst = NlMeans(img)

plt.subplot(211), plt.imshow(img)

plt.subplot(212), plt.imshow(rgb\_dst)

plt.show()

a = CalcOfDamageAndNonDamage(rgb\_dst)

b = CalcOfDamageAndNonDamage(img)

plt.subplot(211), plt.imshow(a)

plt.subplot(212), plt.imshow(b)

plt.show()

# Read the image .

img = cv2.imread('8.jpg')

# diameter = 15, sigmaColor = sigmaSpace = 75.

bilateral = cv2.bilateralFilter(img, 3, 75, 75)

plt.imshow(bilateral)

plt.subplot(211), plt.imshow(img)

plt.subplot(212), plt.imshow(bilateral)

plt.show()

a = CalcOfDamageAndNonDamage(rgb\_dst)

b = CalcOfDamageAndNonDamage(bilateral)

plt.subplot(211), plt.imshow(a)

plt.subplot(212), plt.imshow(b)

plt.show()

medBl = cv2.medianBlur(img, 3)

plt.imshow(medBl)

a = CalcOfDamageAndNonDamage(img)

b = CalcOfDamageAndNonDamage(medBl)

plt.subplot(211), plt.imshow(a)

plt.subplot(212), plt.imshow(b)

plt.show()