

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806: «Вычислительная математика и программирование»

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №5

По курсу: «Цифровая обработка изображений»

Тема: «Цифровая обработка изображений в частотной области.

Часть 1»

Студент: Чернышев Д.В.

Группа: М8О-107М-22

Вариант: 8

Преподаватель: Гаврилов К.Ю.

Москва

2023

СОДЕРЖАНИЕ

1	Задание к лабораторной работе №5.....	3
2	Выполнение лабораторной работы №5	5
3	Выводы	27

1 Задание к лабораторной работе №5

Часть 1

Исходное изображение представлено в виде двумерной матрицы, записанной в формате Matlab в файле *A5_08_1.mat*. Постройте заданное изображение в пространственной области, вычислите и отобразите на экране его амплитудный и фазовый спектр. Используя свойство преобразования Фурье о сдвиге, выполните сдвиг изображения на 120 пикселей вправо и 10 пикселей вниз путем преобразования спектра в частотной области. Вычислите амплитудный и фазовый спектр сдвинутого изображения. Отобразите на экране сдвинутое изображение и его спектр. Проведите анализ и сравнение спектров исходного и сдвинутого изображений. Объясните полученные результаты. Рекомендация. При построении амплитудных спектров изображений целесообразно использовать логарифмическое преобразование яркости. Сохраните все построенные изображения в формате jpg для представления в отчете с комментариями.

Часть 2

Используя в качестве исходного изображения матрицу файла *A5_08_1.mat* из п. 1, постройте центрированный спектр изображения, в котором нулевая частота располагается в центре частотной области. Выполните построение центрированного спектра первым способом – путем преобразования исходного изображения в пространственной области. Сравните два спектра – центрированный и нецентрированный. Выполните построение центрированного спектра вторым способом – путем преобразований полученного в частотной области. Сравните изображения центрированных амплитудных спектров, полученных двумя способами. С помощью обратного преобразования Фурье постройте изображение в пространственной области, соответствующее центрированному спектру изображения в частотной области. Сравните его с исходным изображением.

Сохраните все построенные изображения в формате jpg для представления в отчете с комментариями.

Часть 3

Дано текстовое изображение, представленное файлом *A5_08_3.jpg*. Выполните с этим изображением следующие операции: 1. Вычислите спектр изображения и на экране отобразите его абсолютное значение в центрированном виде. 2. Выполните расфокусировку изображения с помощью фильтра нижних частот (ФНЧ), в качестве которого используйте фильтр Баттервортса второго порядка с частотой среза на уровне 50 пикселей. 3. Для расфокусированного изображения с целью повышения резкости выполните фильтрацию с помощью Лапласиана в частотной области. Частотную характеристику фильтра Лапласиана постройте путем двумерного ДПФ от маски Лапласиана 3x3 в пространственной области. Рекомендация. С целью улучшения эффективности повышения резкости изображения при фильтрации используйте двух- или трехкратное последовательное применение фильтра Лапласиана. Конечный результат фильтрации для улучшения контраста изображения подверните нелинейному преобразованию с помощью гамма-коррекции или логарифмического преобразования яркости в комбинации с эквалайзацией гистограммы. 4. С целью выделения контуров объектов на изображении выполните фильтрацию исходного изображения с помощью фильтра верхних частот (ФВЧ). В качестве ФВЧ используйте гауссов фильтр. Рекомендация. Частоту среза ФВЧ выберите около 200 пикселей.

Сохраните все построенные изображения в формате jpg для представления в отчете с комментариями.

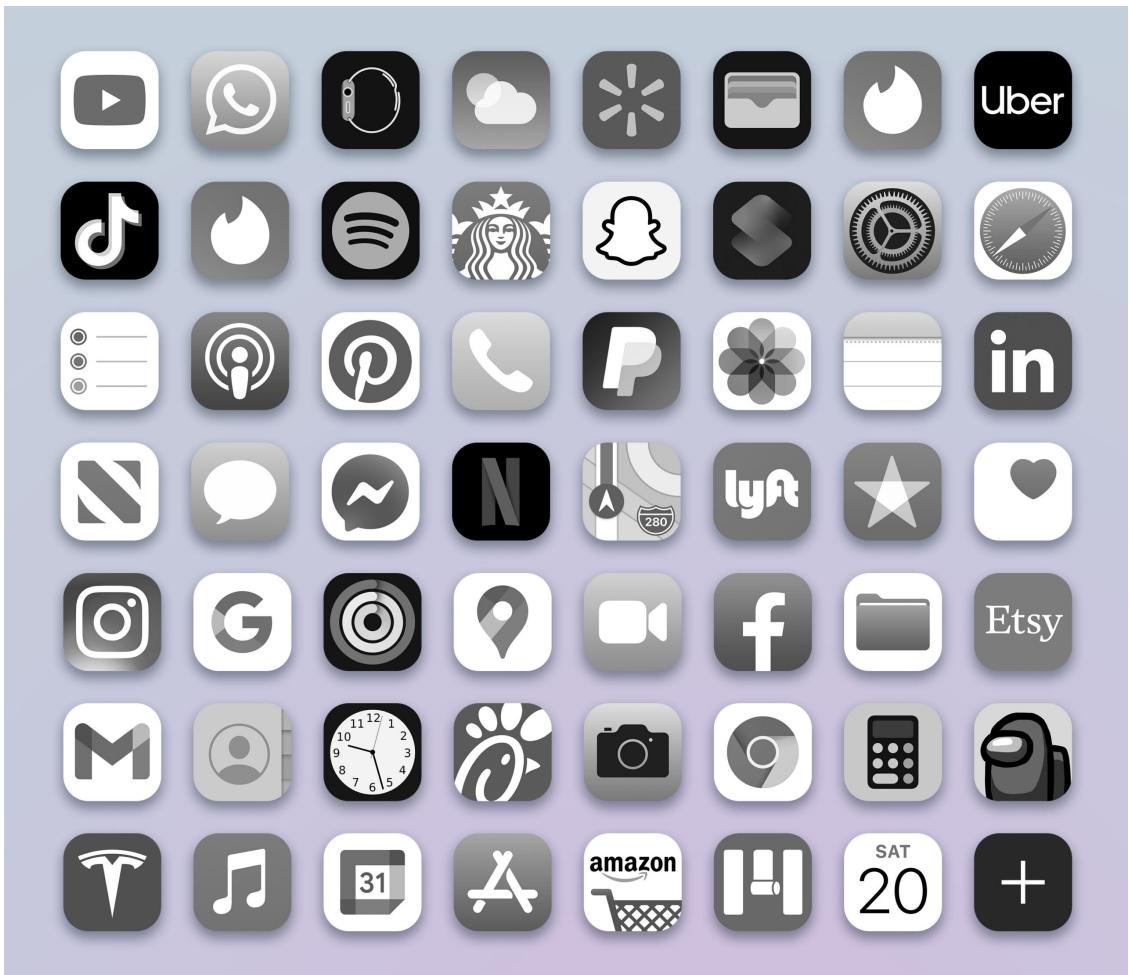


Рис. 1: A5_08_3.jpg

2 Выполнение лабораторной работы №5

Импортируем необходимые библиотеки

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import os
4
5 import cv2
6 from scipy.fft import fft, fftshift, fft2
7 from scipy.fft import fftfreq, rfft
8 from scipy.fft import rfftfreq, irfft,
9     ifft, ifftshift, ifft2, dct
10
11 from scipy import signal
```

```
12 from scipy import ndimage  
13  
14  
15 from PIL import Image, \  
16     ImageDraw, ImageStat ,ImageOps  
17 import cv2  
18 import bisect  
19 from numba import jit  
20 import os
```

Часть 1

Исходное изображение:

```
1 MAT = MAT_RAW_FIX.reshape(*MAT_RAW_FIX.shape).astype(np.uint8)  
2  
3 MAT = np.stack((MAT,) * 3, axis=-1)  
4 image_raw = Image.fromarray(MAT)  
5 image_raw.save(os.path.join(OUTPUT,"raw_image.png"))  
6 %matplotlib inline  
7 plt.figure(figsize=(6,6))  
8 plt.axis("off")  
9 plt.imshow(image_raw)
```

Для начала подгрузим необходимые модули для работы

```
1 # загрузка изображения в оттенках серого  
2 img = cv2.imread(os.path.join(OUTPUT,"raw_image.png"), 0)  
3  
4 # применение двумерного БПФ (Быстрое преобразование Фурье)  
5 f = fft2(img)  
6  
7 # сдвиг низких частот к центру спектра  
8 fshift = fftshift(f)  
9
```



Рис. 2: A5_08_3.jpg

```
10 # вычисление амплитудного спектра
11 magnitude_spectrum = 20*np.log(np.abs(fshift))
12
13 # вычисление фазового спектра
14 phase_spectrum = np.angle(fshift)
15 plt.figure(figsize=(18,9))
16
17 plt.subplot(141),plt.imshow(img, cmap='gray')
18 plt.title('Input Image'), plt.xticks([]), plt.yticks([])
19
20 plt.subplot(142),plt.imshow(20*np.log(np.abs(f)), cmap='gray')
21 plt.title('Magnitude Spectrum (Not shifted)')
22 plt.xticks([])
23 plt.yticks([])
24
25 plt.subplot(143),plt.imshow(magnitude_spectrum, cmap='gray')
```

```

26 plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
27
28 plt.subplot(144),plt.imshow(phase_spectrum, cmap='gray')
29 plt.title('Phase Spectrum'), plt.xticks([]), plt.yticks([])
30
31 plt.savefig(os.path.join(OUTPUT,"InputPlots.png"))

```

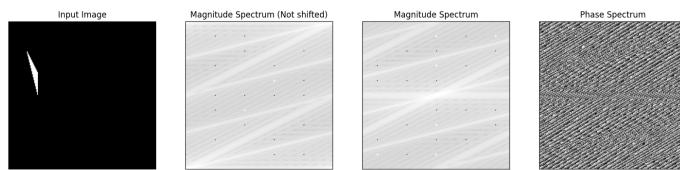


Рис. 3: Графики спектра и фазы исходного изображения

Расчет быстрого преобразования Фурье

```

1 f = fft2(img)
2
3 # Shift zero frequency to center
4 fshift = fftshift(f)
5
6 # Define the distance to shift along x and y axis
7 dx = 120
8 dy = -10
9
10 # Compute shift along x and y axis
11 M, N = img.shape
12 shiftx, shifty = np.meshgrid(np.arange(N), np.arange(M))
13 shiftx = np.exp(-2j * np.pi * dx * shiftx / N)
14 shifty = np.exp(-2j * np.pi * dy * shifty / M)
15

```

```

16 # Shift spectrum to perform shift in spatial domain
17 fshift = fshift * shiftx * shifty
18
19 # Inverse shift for visualizing the result image
20 f_ishift = ifftshift(fshift)
21
22 # Compute inverse fft of shifted spectrum
23 img_back = ifft2(f_ishift)
24
25 img_back = np.abs(img_back)
26
27
28
29 # Отображение измененного изображения
30 plt.figure(figsize=(16,8))
31 plt.subplot(121)
32 plt.imshow(img, cmap = 'gray')
33 plt.title('Исходное изображение')
34 plt.xticks([])
35 plt.yticks([])
36 plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
37 plt.title('Сдвинутое изображение')
38 plt.xticks([])
39 plt.yticks([])
40 plt.show()
41
42 plt.savefig(os.path.join(OUTPUT,"CompareInputShifted.png"))

```

Применение двумерного БПФ (Быстрое преобразование Фурье)

```

1 # применение двумерного БПФ (Быстрое преобразование Фурье)
2 f = fft2(img_back)
3
4 # сдвиг низких частот к центру спектра

```

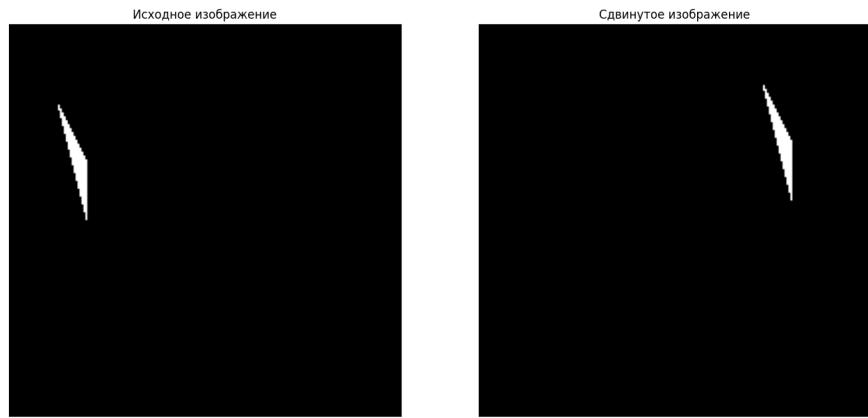


Рис. 4: Сравнение исходного и сдвинутого изображения

```

5   fshift = fftshift(f)

6

7 # вычисление амплитудного спектра
8 magnitude_spectrum = 20*np.log(np.abs(fshift))

9

10 # вычисление фазового спектра
11 phase_spectrum = np.angle(fshift)

12

13 plt.figure(figsize=(18,9))

14

15 plt.subplot(141)
16 plt.imshow(img_back, cmap='gray')
17 plt.title('Input Image')
18 plt.xticks([])
19 plt.yticks([])

20

21 plt.subplot(142)
22 plt.imshow(20*np.log(np.abs(f)), cmap='gray')
23 plt.title('Magnitude Spectrum (Not shifted)')
24 plt.xticks([])
25 plt.yticks([])

26

27 plt.subplot(143)

```

```
28 plt.imshow(magnitude_spectrum, cmap='gray')
29 plt.title('Magnitude Spectrum')
30 plt.xticks([])
31 plt.yticks([])

32
33 plt.subplot(144),plt.imshow(phase_spectrum, cmap='gray')
34 plt.title('Phase Spectrum')
35 plt.xticks([])
36 plt.yticks([])

37
38 plt.savefig(os.path.join(OUTPUT, "ShiftedPlots.png"))
```

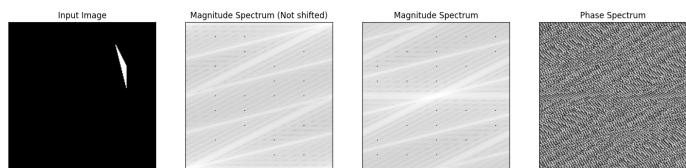


Рис. 5: Графики спектра и фазы сдвинутого изображения

Часть 2

```
1 def F_chess_2D(height, width):
2     chessboard = np.indices((height, width)).sum(axis=0) % 2
3     return chessboard
4
5 # Чтение изображения
6 M = np.loadtxt(file)
7 sz = np.shape(M)
8 sy = sz[0] # высота
9 sx = sz[1] # ширина
10 CH = F_chess_2D(sy, sx)
11
12 # Спектр, центрированный в пространственной области
13 RF = fft2(M * CH)
14
15 # Исходное изображение
16 plt.figure('Original image', figsize=(12,12))
17 plt.imshow(M, cmap='gray')
18 plt.axis('off')
19 plt.title('Original image')
20 plt.axis("off")
21 plt.savefig(os.path.join(OUTPUT, "InputIm"))
22
```

Спектр, центрированный в частотной области

```
1 RF1 = fft2(M)
2 ksdvm = 100
3 ksdvn = 100
4 FIMsdv = RF1.copy()
5
6 for im in range(sy):
7     for in_ in range(sx):
```

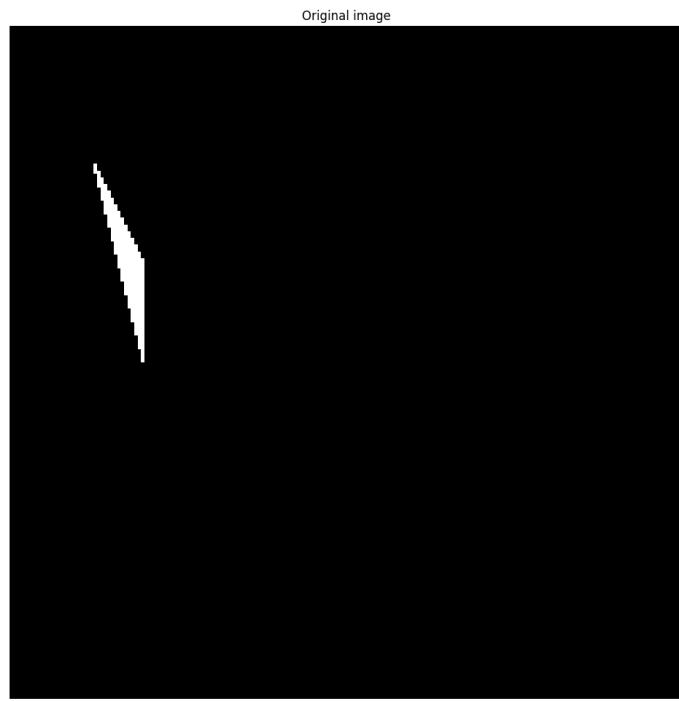


Рис. 6: Исходное изображение, его амплитудный (центрированный) и фазовый спектр, построенный путем преобразования исходного изображения в пространственной области

```

8      if im - ksdvm >= 0 and in_ - ksdvn >= 0:
9          FIMsdv[im, in_] = RF1[
10             im - ksdvm,
11             in_ - ksdvn]
12      elif im - ksdvm >= 0:
13          FIMsdv[im, in_] = RF1[
14              im - ksdvm,
15              sx - ksdvn + in_]
16      elif in_ - ksdvn >= 0:
17          FIMsdv[im, in_] = RF1[
18              sy - ksdvm + im,
19              in_ - ksdvn]
20      else:
21          FIMsdv[im, in_] = RF1[

```

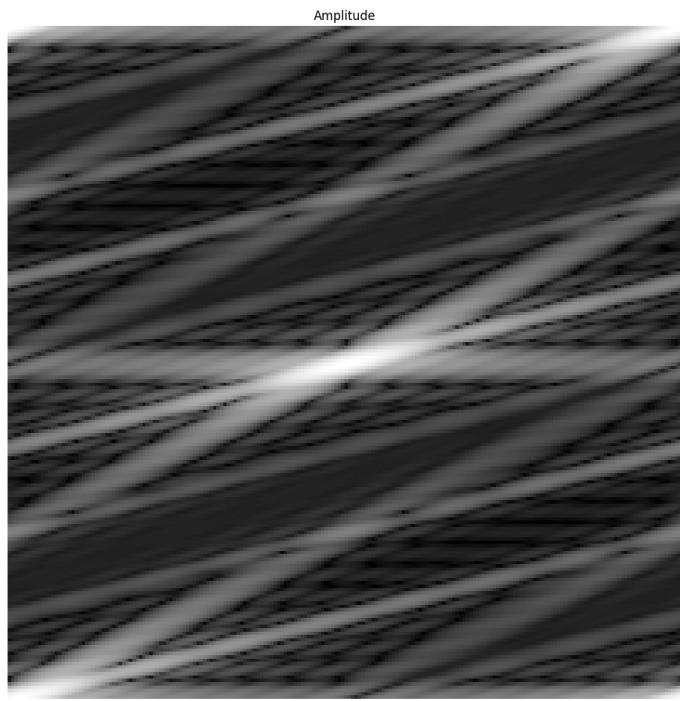


Рис. 7: Амплитуда исходного изображения

```

22                     sy - ksdvm + im,
23                     sx - ksdvn + in_]

24
25 SIM = ifft2(FIMsdv)
26 plt.figure(figsize=(12,12))
27 plt.imshow(np.abs(SIM), cmap='gray')
28 plt.axis('off')
29 plt.title('Filtered Image')
30 plt.axis("off")
31 plt.savefig(os.path.join(OUTPUT, "FilteredInput"))
32

```

Часть 3

```

1 img_source =     Image.open(SOURCE_PIC).convert('L')
2

```

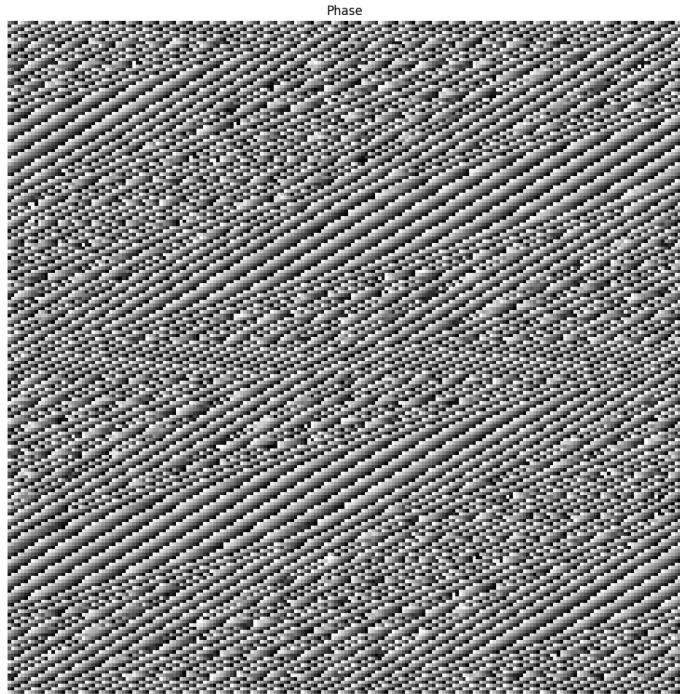


Рис. 8: Фаза исходного изображения

```
3 plt.figure(figsize=(11,11))
4 plt.imshow(img_source,cmap='gray')
5
6 plt.axis('off')
7 plt.savefig(os.path.join(OUTPUT_DIR_3,"Input"))
```

Применим преобразование Фурье

```
1 # Aply fft2 to image
2 f = fft2(img_source)
3 # Shift
4 f_shift = fftshift(np.abs(f))

5

6

7 make_subplots(
8     [
9         img_source,20 * np.log(f_shift)
```



Рис. 9: Восстановленное изображение, его амплитудный (центрированный) и фазовый спектр, построенный путем преобразования изображения в частотной области

```
10      ],
11
12      [
13          "A5_08_3.jpg" , "Spectrum Centered"
14      ]
15
16
17 # Convert image mode to grayscale
18 image_gray = np.abs(20 * np.log(f_shift)).astype(np.uint8)
19
20 # Create PIL Image object
21 image_pil = Image.fromarray(image_gray, mode='L')
22
```

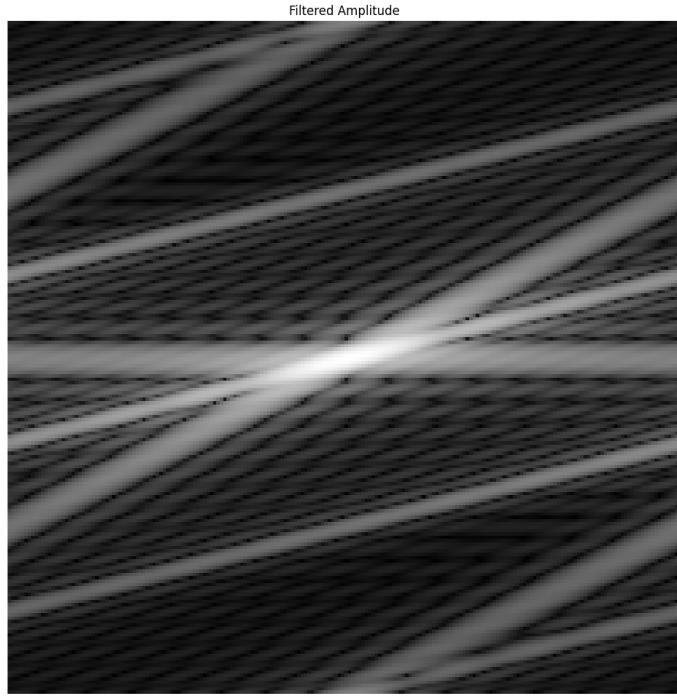


Рис. 10: Амплитуда исходного изображения

```
23 # Save the image as JPEG
24 image_pil.save(os.path.join(OUTPUT_DIR_3, "Input_Spectrum.jpg"))
```

Функции фильтра Баттервортса и ФВЧ

```
1 def butterworth_filter(image, cutoff_frequency, order):
2     # Размеры изображения
3     sy, sx = image.size
4     sx2, sy2 = sx // 2, sy // 2
5
6     # Создание фильтра Баттервортса
7     HB = np.zeros((sy, sx))
8     for im in range(sy):
9         for in_ in range(sx):
10            dt = np.sqrt((in_ - sx2) ** 2 + (im - sy2) ** 2)
11            HB[im, in_] = 1 / (1 + (dt / cutoff_frequency) \
12                               ** (2 * order))
```

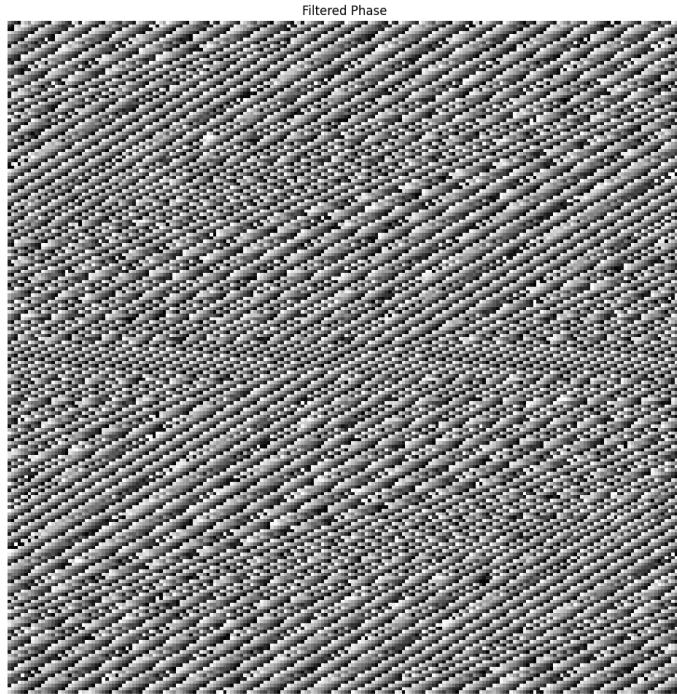


Рис. 11: Фаза исходного изображения

```
13
14 # Применение фильтра к спектру изображения
15 FB = fft2(image)
16 G0 = np.abs(
17     ifft2(
18         signal.fftconvolve(FB,fft2(HB),
19         mode="same"
20     )
21     )
22 )
23
24 return G0
25
26 def high_pass_filter(image, cutoff_frequency):
27     # Применение двумерного дискретного
28     # преобразования Фурье (ДПФ) к изображению
```



Рис. 12: A5_08_3.jpg

```
29     image_frequency = fft2(image)

30

31 # Центрирование нулевой частоты
32 image_frequency_shifted = fftshift(image_frequency)

33

34 # Создание фильтра верхних
35 # частот (гауссов фильтр)
36 rows, cols = image.shape
37 center_row, center_col = rows // 2, cols // 2
38 x = np.arange(cols) - center_col
39 y = np.arange(rows) - center_row
40 xx, yy = np.meshgrid(x, y)
41 filter_mask = np.exp(-((xx ** 2 + yy ** 2) \
42                         / (2 * cutoff_frequency ** 2)))
43
44 # Применение фильтра верхних
```

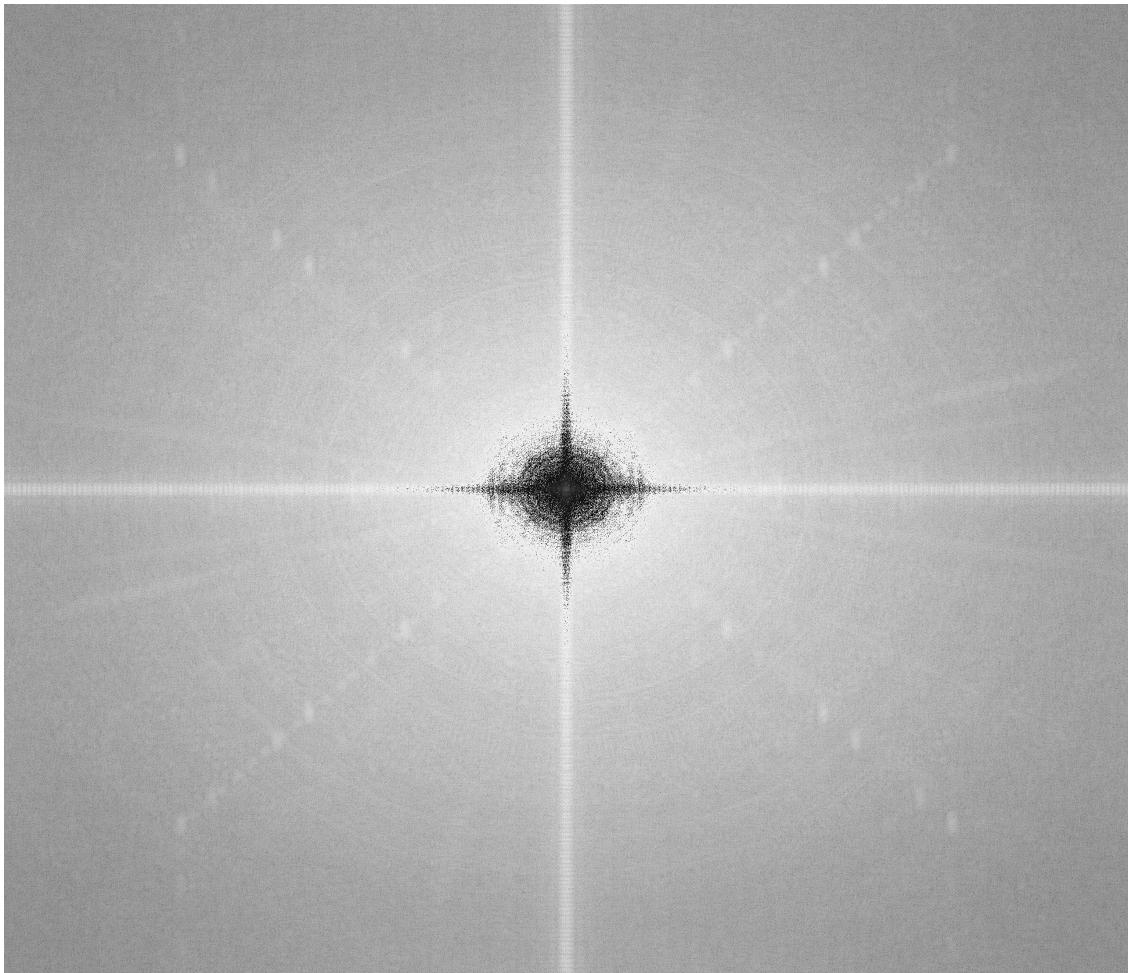


Рис. 13: Спектр изображения

```
45 # частот к спектру изображения
46 filtered_frequency_shifted = image_frequency_shifted * \
47     (1 - filter_mask)

48
49 # Обратное преобразование Фурье
50 # для получения отфильтрованного изображения
51 filtered_frequency = ifftshift(filtered_frequency_shifted)
52 filtered_image = ifft2(filtered_frequency)
53 filtered_image = np.abs(filtered_image)

54
55 # Масштабирование значений
56 # пикселей в диапазон от 0 до 255
57 filtered_image = (filtered_image - np.min(filtered_image)) \
58     / (np.max(filtered_image) - np.min(filtered_image))
```

```

59     filtered_image = (filtered_image * 255).astype(np.uint8)
60
61     return filtered_image

```

Функции фильтра Баттервортса и ФВЧ

```

1 W = 50 # Частота среза фильтра в пикселях
2 N = 2 # порядок
3 filt_image = butterworth_filter(
4     img_source,
5     cutoff_frequency=W,
6     order=N)
7 make_subplots(
8     [
9         # img_source,
10        filt_image,
11    ],
12
13    [
14        # "input",
15        f"Filtered A5_08_3_(freq={W},Order={N})",
16    ]
17 )
18 # Convert image mode to grayscale
19 image_gray = np.abs(20 * np.log(filt_image)).astype(np.uint8)
20
21 # Create PIL Image object
22 image_pil = Image.fromarray(image_gray, mode='L')
23
24 # Save the image as JPEG
25 image_pil.save(
26     os.path.join(OUTPUT_DIR_3,
27     f"Filtered A5_08_3_(freq={W},Order={N}).jpg"))
28

```

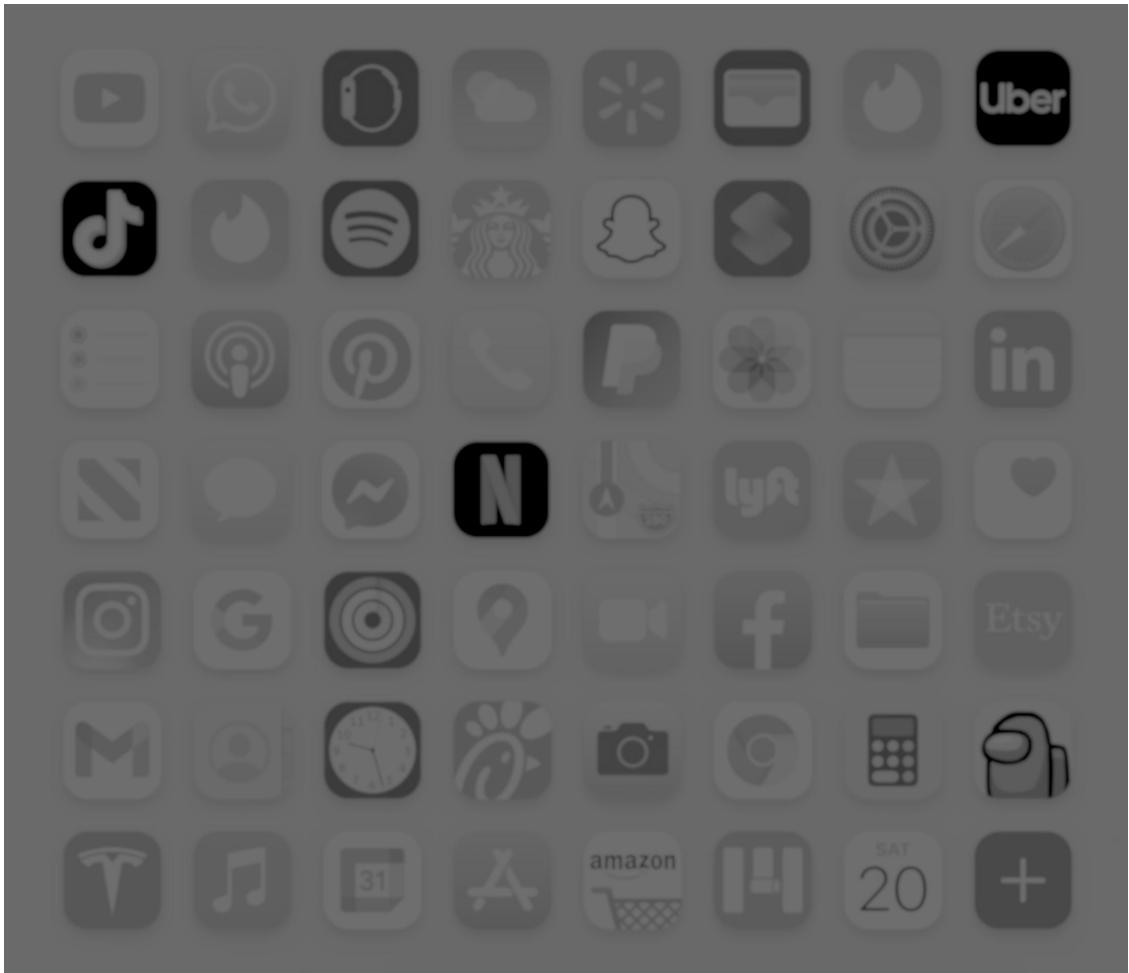


Рис. 14: Зашумленное изображение

Функции фильтра Лапласиана в частотной области и маски Лапласиана

```
1 def apply_laplacian_filter(
2     image,
3     iterations,
4     dimension:int=3,
5     center_value:int=4):
6     filtered_image = image.copy()
7     for i in range(iterations):
8         # Применение преобразования Фурье
9         f = np.fft.fft2(filtered_image)
10        fshift = np.fft.fftshift(f)
11
12        # Создание маски Лапласиана в частотной области
13        mask = laplacian_mask(dimension,center_value)
```

```

14     # Применение маски Лапласиана
15     fshift_filtered = signal.fftconvolve(
16         fshift,
17         mask,
18         "same")
19
20     # Обратное преобразование Фурье
21     f_ishift = np.fft.ifftshift(fshift_filtered)
22     filtered_image = np.fft.ifft2(f_ishift)
23     filtered_image = np.abs(filtered_image)
24
25     return filtered_image
26
27 def laplacian_mask(size, center_value):
28     center = (size - 1) // 2
29     mask = np.ones((size, size)) * (-np.sign(center_value))
30     mask[center, center] = center_value
31
32     return mask

```

Функции фильтра Лапласиана в частотной области и маски Лапласиана

```

1 iterations = 3
2 # Применение фильтра Лапласиана
3
4 # Параметры
5 lapl_img = filt_image
6 ksize = 3
7 val = 4
8 # Применение
9 lapl_img = apply_laplacian_filter(
10     lapl_img,
11     iterations=iterations,
12     dimension=ksize,

```

```

13     center_value=32)

14

15

16 # Повышение контраста изображения
17 # с помощью гамма-коррекции или эквализации гистограммы
18 c = 255 / np.log(1 + np.max(lapl_img))
19 log_image = c * (np.log(lapl_img + 1))
20 log_image = np.array(log_image)

21

22 # Преобразование типа данных в uint8
23 log_image = log_image.astype(np.uint8)

24

25 # Эквализация гистограммы
26 enhanced_image = cv2.equalizeHist(log_image)

27

28 make_subplots(
29     [filt_image,
30         enhanced_image
31     ],
32     [f"Filtered(f={W},Or={N})",
33         f"Lapl_(it_{iterations}__ks_{ksize}__val_{val})_Eq_Log"
34     ]
35 )
36 Image.fromarray(enhanced_image).save(
37     os.path.join(
38         OUTPUT_DIR_3,
39         f"Lapl_(it_{iterations}__ks_{ksize}__val_{val})_Eq_Log.png"
40     )
41 )

```

Примените фильтр Гаусса к исходному изображению:

```

1 freq_cut = 200
2 # Примените фильтр Гаусса к исходному изображению:

```



Рис. 15: После фильтрации Лапласианом и эквализации

```
3 image_contours = high_pass_filter(  
4     enhanced_image,  
5     cutoff_frequency=freq_cut  
6 )  
7  
8 make_subplots(  
9     [  
10        enhanced_image,  
11        image_contours  
12    ],  
13    [  
14        f'Lapl(it={iterations})_Eq_Log",  
15        f'Countours_cut_freq_{freq_cut}"  
16    ]
```

```
17 )
18 # Convert image mode to grayscale
19 image_gray = np.abs(20 * np.log(
20     image_contours)).astype(np.uint8)
21
22 # Create PIL Image object
23 image_pil = Image.fromarray(image_gray, mode='L')
24
25 # Save the image as JPEG
26 image_pil.save(
27     os.path.join(
28         OUTPUT_DIR_3,
29         f"Countours_cut_freq_{freq_cut}.jpg"
30     )
31 )
32
```



Рис. 16: Итоговое обработанное изображение

3 Выводы

В ходе выполнения данной лабораторной работы были обработаны различные методы цифровой обработки изображений в частотной области были хорошо обработанные искусственно зашумленные фотографии