|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Калужский филиал федерального государственного автономного образовательного учреждения высшего образования**  **«Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»**  **(КФ МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИУК «Информатика и управление

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологи»

**ДОМАШНЯЯ РАБОТА**

**«Преобразование изображений, наложение шума**

**на изображение и фильтрация изображений»**

**по дисциплине: «Цифровая обработка сигналов»**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил: студент группы ИУК4-72Б | |  |  | И. А. Петроченков | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Проверил(-а): | |  |  | О. И. Чурилин | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | | | |
|  | - Балльная оценка:  - Оценка: | | | |

Калуга, 2025

**Целью** выполнения домашней работы является формирование практических навыков доступа к объектам графики с помощью программных средств, выполнения сравнительного анализа различных преобразований использования двумерного преобразования Фурье при исследовании диапазона яркости изображения, наложения на изображение шума и фильтрации изображения.

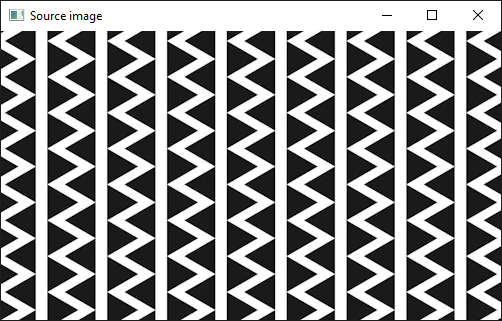
**Основными задачами** выполнения лабораторной работы являются:

1. Вывести изображение в графическое окно;
2. Осуществить прямое и обратное косинусное преобразование над изображением. Вывести в графическое окно результаты преобразований;
3. Осуществить прямое и обратное преобразование Фурье над изображением. Вывести в графическое окно результаты преобразований;
4. Определить коэффициент корреляции между исходным изображением и изображениями, полученными в результате обратных преобразований.
5. Выявить какое из полученных изображений менее всего отличается от оригинала;
6. Построить график зависимости спектра яркости от частоты;
7. На исходное изображение наложить различного рода шум, согласно варианту;
8. Произвести фильтрацию исходного изображения, согласно варианту;
9. Производить фильтрацию для трех различных масок фильтров;
10. С помощью коэффициента корреляции оценить действие каждого из фильтров с учетом размера маски. Построить графики зависимости коэффициента корреляции от размера маски.

**Вариант задания:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ вар.** | **Вид шума, тип фильтра** | **Вид шума, тип фильтра** | **Вид шума, тип фильтра** | |
| 19 | Гауссовый шум, медианная фильтрация | Шум в виде включенных пикселей, ранговая фильтрация | | Мультипликативный шум, фильтрация Винера |

**Порядок выполнения работы:**

****

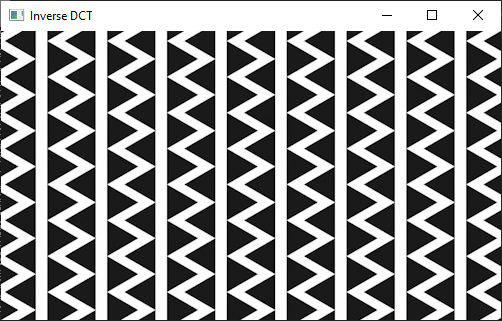
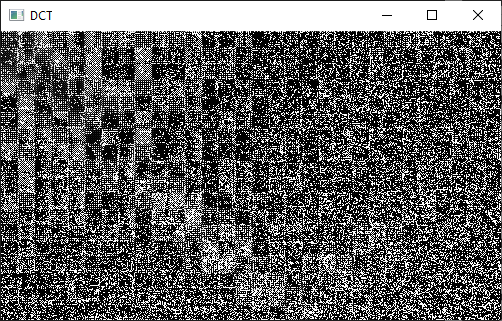
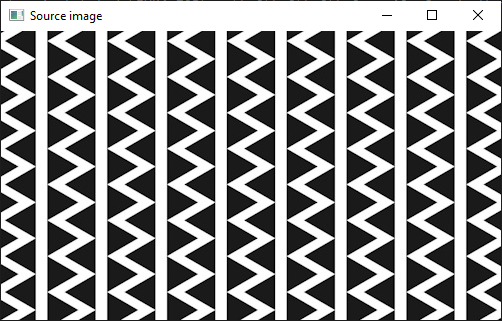
****

Рис. 1 Сравнение исходного изображения, результата прямого и обратного косинусных преобразований над исходным изображением



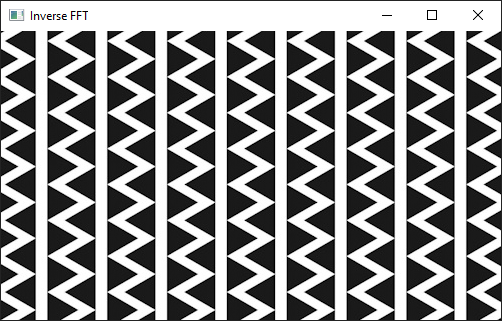
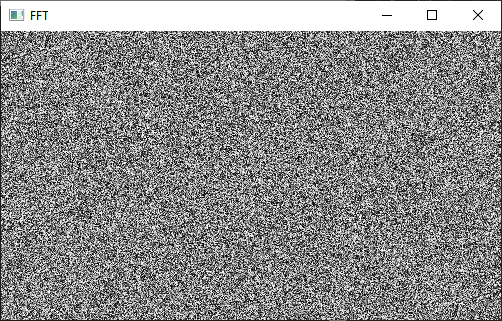


Рис. 2 Сравнение исходного изображения, результата прямого и обратного дискретных преобразований Фурье над исходным изображением

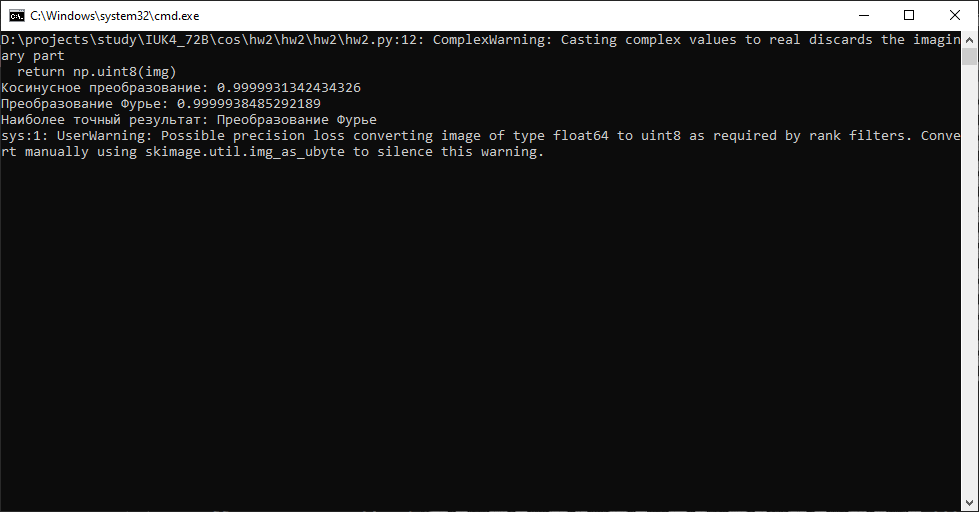
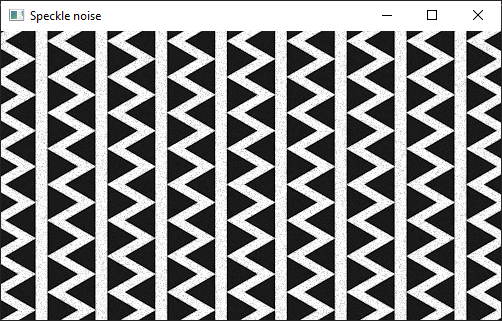
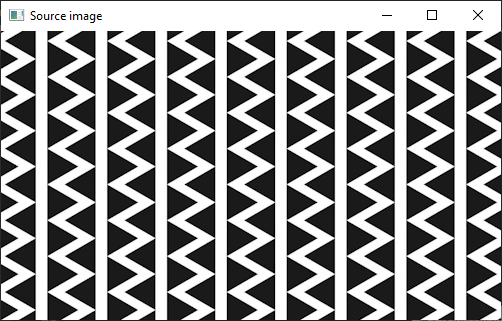


Рис. 3 Сравнение коэффициентов корреляции между исходным изображением, результатом обратных преобразований Фурье и косинусного преобразования



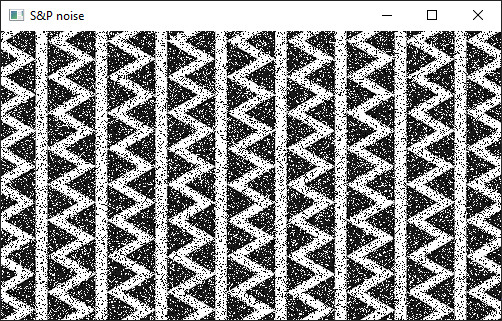
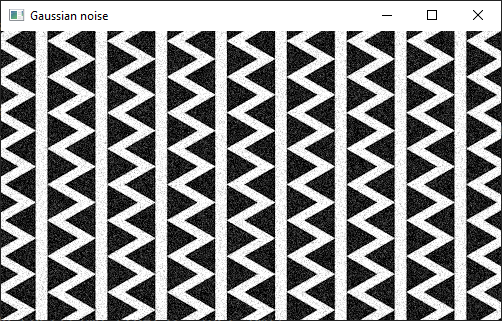
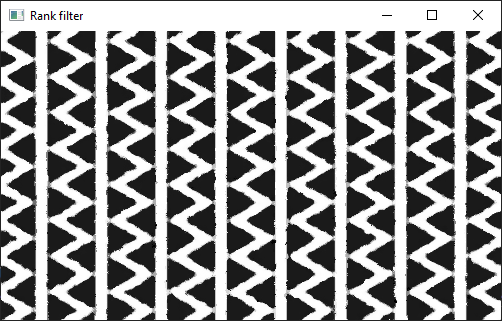
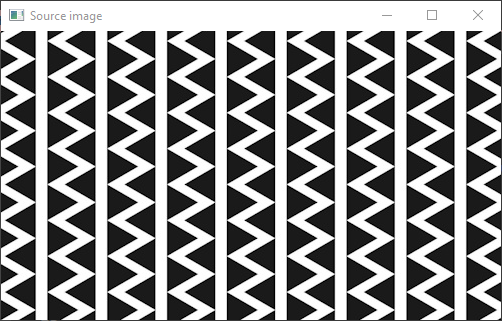


Рис. 4 Результат применения к исходному изображению шумов разного типа: шум в виде включенных пикселей, мультипликативный шум, Гауссовый шум



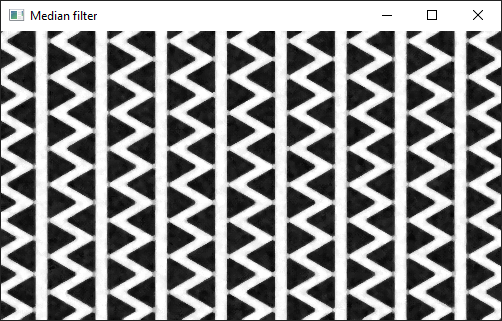
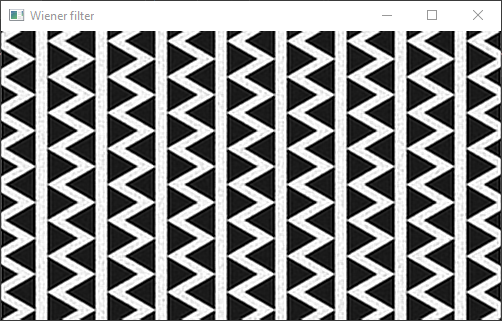
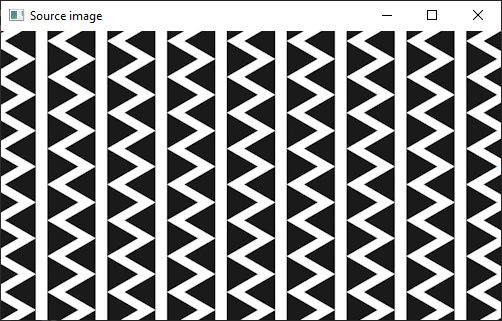


Рис. 5 Результат применения фильтрации к зашумленным изображениям (размер маски фильтра – 3 пикс.)



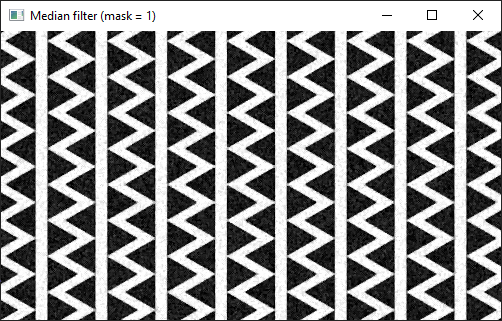
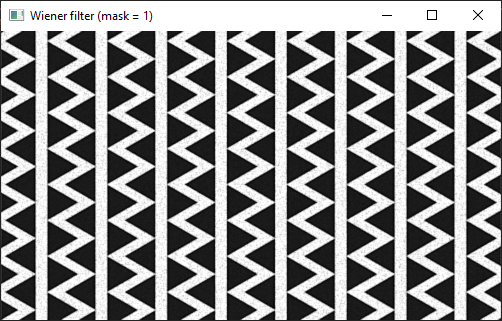
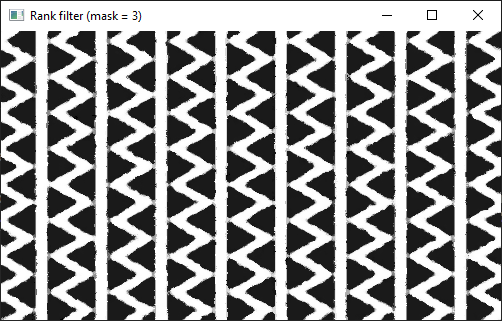
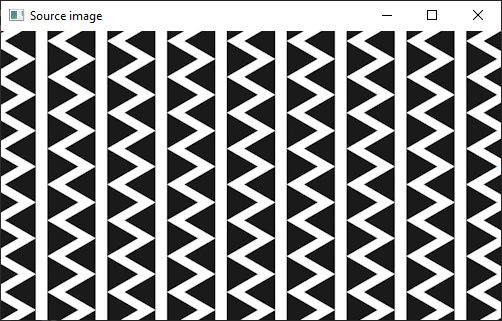


Рис. 6 Сравнение результатов фильтрации в зависимости от размера маски фильтра (1 пикс.)



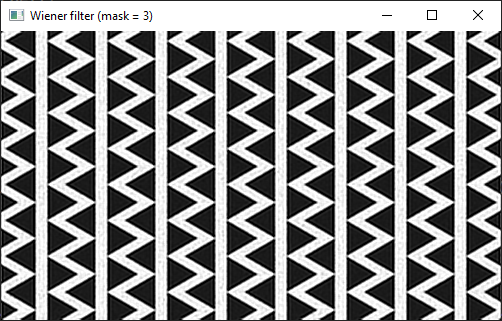
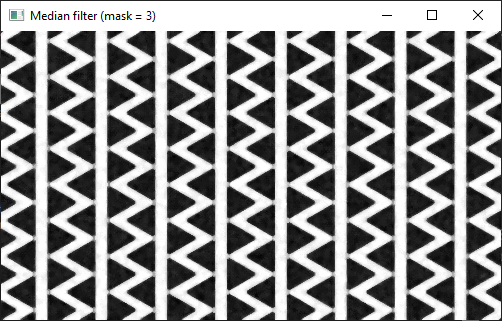
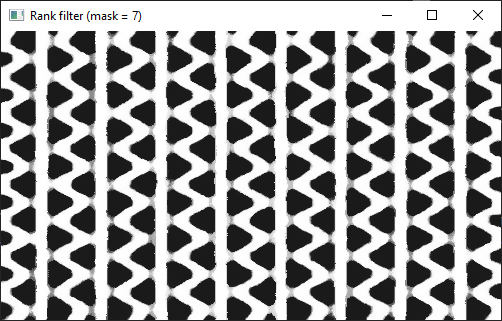
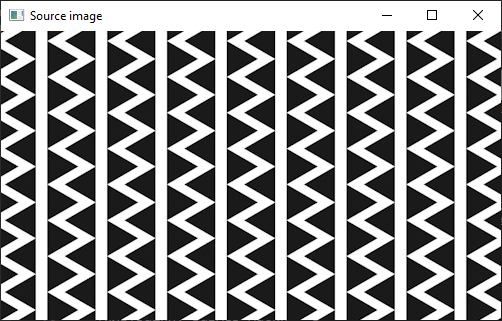


Рис. 7 Сравнение результатов фильтрации в зависимости от размера маски фильтра (3 пикс.)



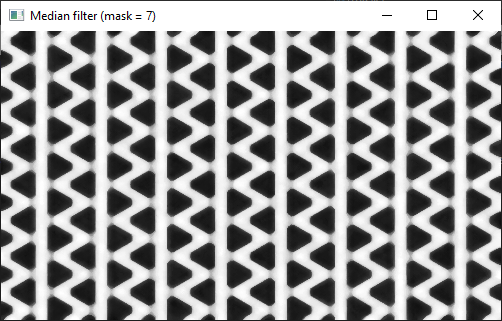


Рис. 8 Сравнение результатов фильтрации в зависимости от размера маски фильтра (7 пикс.)

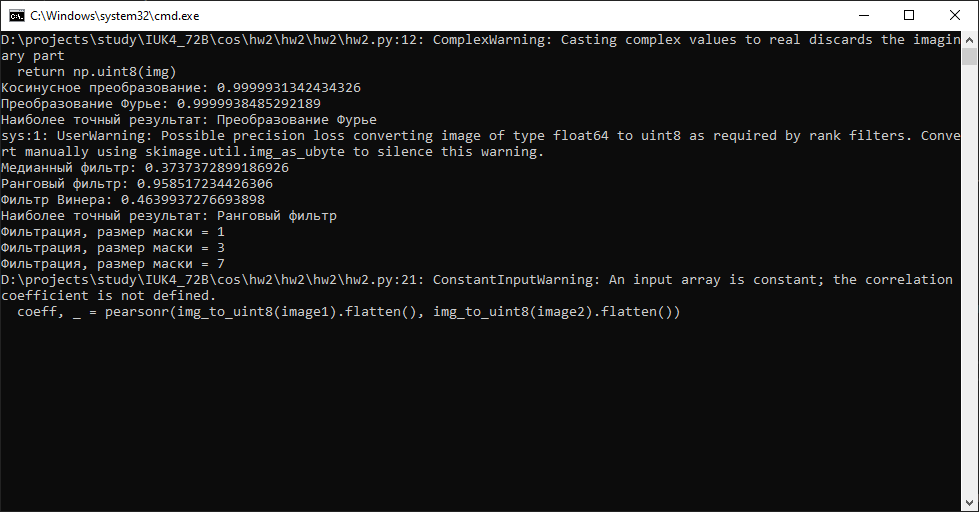


Рис. 9 Результат сравнение коэффициентов корреляции отфильтрованных от шума с помощью различных подходов зашумленных изображений

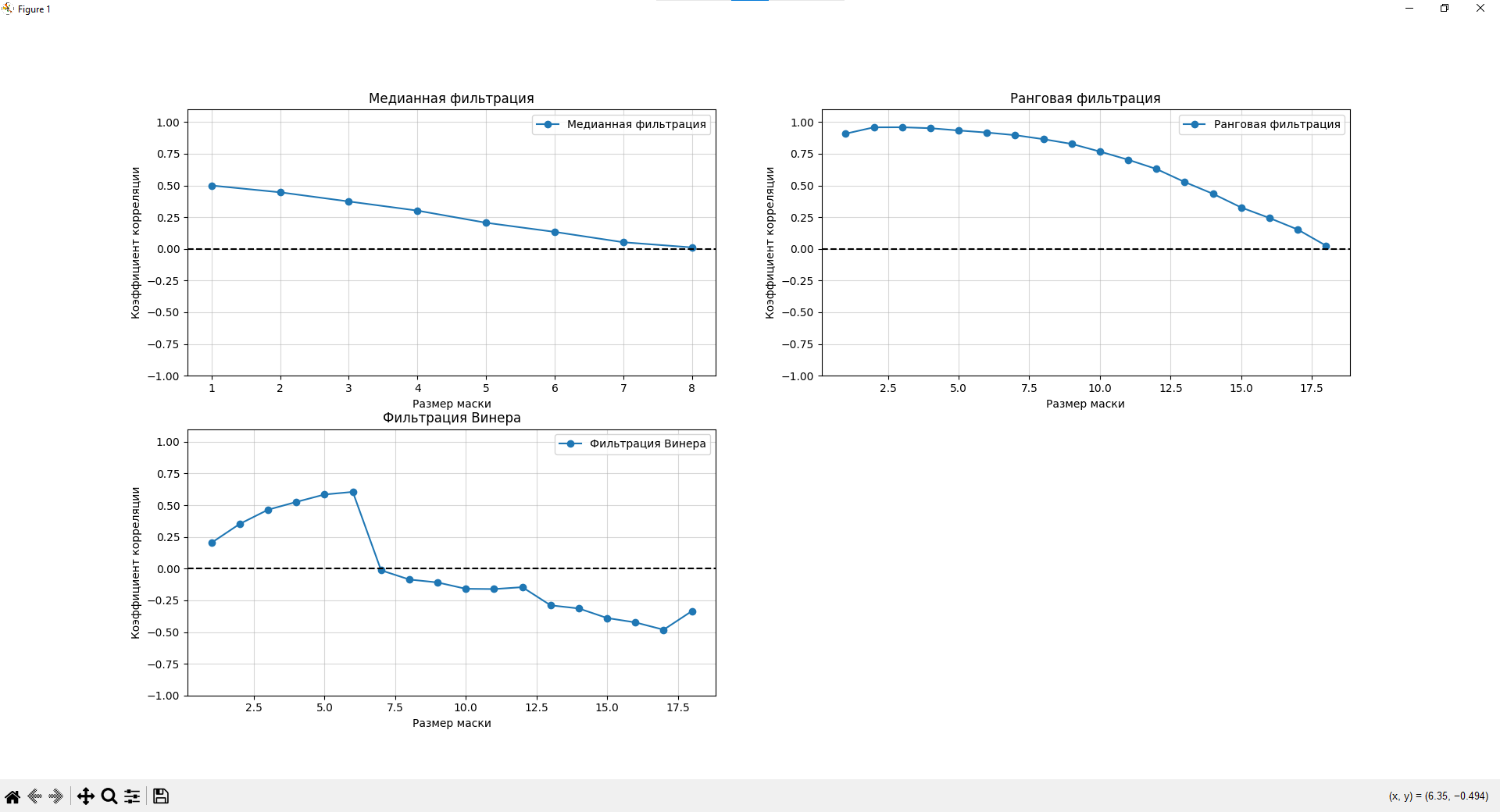


Рис. 10 График зависимости значений коэффициента корреляции исходного и отфильтрованного изображения в зависимости от размера маски фильтра

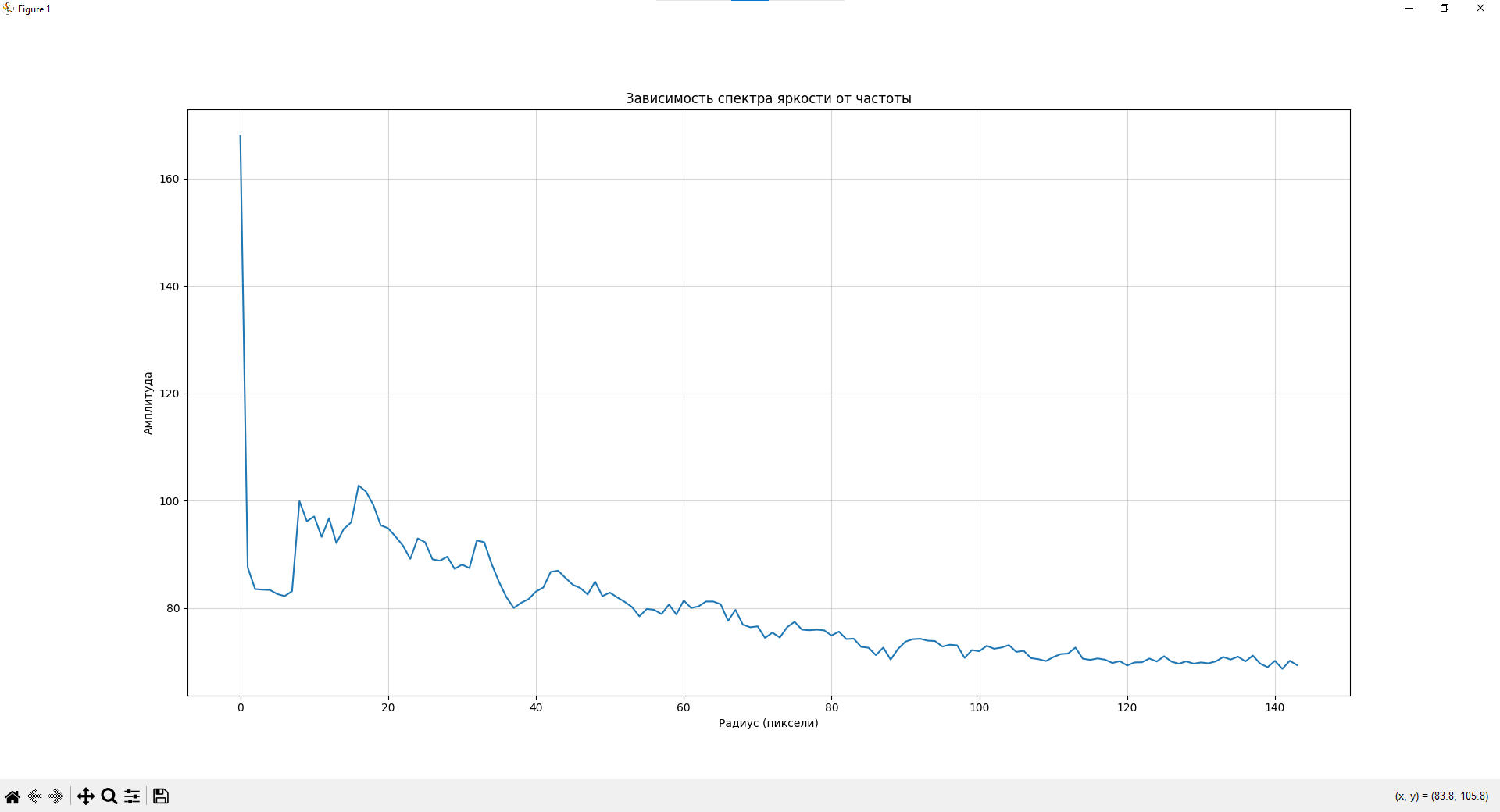


Рис. 11 График зависимости спектра яркости от частоты

**Листинг кода:**

**hw2.py**

import cv2

import numpy as np

from scipy.stats import pearsonr

from skimage import util as util

from skimage import filters as filt

from skimage.morphology import disk, square

from skimage import restoration

from scipy import ndimage

import matplotlib.pyplot as plt

def img\_to\_uint8(img):

return np.uint8(img)

def show\_images(images\_infos):

for name in images\_infos:

cv2.imshow(name, images\_infos[name])

cv2.waitKey(0)

cv2.destroyAllWindows()

def get\_correlation\_coeff(image1, image2):

coeff, \_ = pearsonr(img\_to\_uint8(image1).flatten(), img\_to\_uint8(image2).flatten())

return coeff

img = cv2.imread('./image2.jpg', 0)

img\_float32 = np.float32(img)

dct\_img = cv2.dct(img\_float32)

idct\_img = cv2.idct(dct\_img)

dct\_corr\_coeff = get\_correlation\_coeff(img\_float32, idct\_img)

show\_images({'Source image' : img, 'DCT': img\_to\_uint8(dct\_img), 'Inverse DCT': img\_to\_uint8(idct\_img)})

f\_img = np.fft.fft2(img\_float32)

if\_img = np.fft.ifft2(f\_img)

f\_corr\_coeff = get\_correlation\_coeff(img\_float32, if\_img)

show\_images({'Source image' : img, 'FFT': img\_to\_uint8(f\_img), 'Inverse FFT': img\_to\_uint8(if\_img)})

def show\_corr\_coeffs(corr\_coeffs):

for coeff in corr\_coeffs:

print(f'{coeff}: {corr\_coeffs[coeff]}')

mx\_coeff = max(corr\_coeffs, key=corr\_coeffs.get)

print(f'Наиболее точный результат: {mx\_coeff}')

corr\_coeffs = {'Косинусное преобразование': dct\_corr\_coeff, 'Преобразование Фурье': f\_corr\_coeff}

show\_corr\_coeffs(corr\_coeffs)

# print(f'Коэфф. корреляции косинусного преобразования: {dct\_corr\_coeff}\nКоэфф. корреляции преобразования Фурье: {f\_corr\_coeff}')

# if dct\_corr\_coeff > f\_corr\_coeff:

# print ('Результат косинусного преобразования более точен!')

# else:

# print ('Результат преобразования Фурье более точен!')

gaussian\_img = util.random\_noise(img, mode='gaussian', var=0.01)

salt\_and\_pepper\_img = util.random\_noise(img, mode='s&p', amount=0.3)

speckle\_img = util.random\_noise(img, mode='speckle',var=0.01)

mask\_size = 3

def get\_psf(ds):

psf = ndimage.gaussian\_filter(np.ones((int(ds), int(ds))), sigma=1.1)

psf = psf / psf.sum()

return psf

balance = 0.05

median\_img = filt.median(gaussian\_img, footprint=disk(mask\_size))

rank\_img = filt.rank.median(salt\_and\_pepper\_img, footprint=disk(mask\_size))

wiener\_img = restoration.wiener(speckle\_img, get\_psf(mask\_size), balance=balance)

show\_images({'Source image' : img, 'Gaussian noise': gaussian\_img, 'S&P noise': salt\_and\_pepper\_img, 'Speckle noise': speckle\_img})

show\_images({'Source image' : img, 'Median filter': median\_img, 'Rank filter': rank\_img, 'Wiener filter': wiener\_img})

median\_corr\_coeff = get\_correlation\_coeff(img\_float32, median\_img)

rank\_corr\_coeff = get\_correlation\_coeff(img\_float32, rank\_img)

wiener\_corr\_coeff = get\_correlation\_coeff(img\_float32, wiener\_img)

corr\_coeffs = {'Медианный фильтр': median\_corr\_coeff, 'Ранговый фильтр': rank\_corr\_coeff, 'Фильтр Винера': wiener\_corr\_coeff}

show\_corr\_coeffs(corr\_coeffs)

masks = [1, 3, 7]

for mask in masks:

print(f"Фильтрация, размер маски = {mask}")

median\_img = filt.median(gaussian\_img, footprint=disk(mask))

rank\_img = filt.rank.median(salt\_and\_pepper\_img, footprint=disk(mask))

wiener\_img = restoration.wiener(speckle\_img, get\_psf(mask), balance=balance)

show\_images({'Source image' : img, f'Median filter (mask = {mask})': median\_img, f'Rank filter (mask = {mask})': rank\_img, f'Wiener filter (mask = {mask})': wiener\_img})

mask\_sizes = np.arange(1, 19, 1)

median\_coeffs = [get\_correlation\_coeff(img\_float32, filt.median(gaussian\_img, footprint=disk(i))) for i in mask\_sizes]

rank\_coeffs = [get\_correlation\_coeff(img\_float32, filt.rank.median(salt\_and\_pepper\_img, footprint=disk(i))) for i in mask\_sizes]

wiener\_coeffs = [get\_correlation\_coeff(img\_float32, restoration.wiener(speckle\_img, get\_psf(i), balance=balance)) for i in mask\_sizes]

plt.figure()

plt.subplot(2,2,1)

plt.plot(mask\_sizes, median\_coeffs, 'o-', label='Медианная фильтрация')

plt.xlabel('Размер маски')

plt.ylabel('Коэффициент корреляции')

plt.grid(True, alpha=0.5)

plt.ylim(-1.0, 1.1)

plt.axhline(y=0.0, color='black', linestyle='--')

plt.legend()

plt.title('Медианная фильтрация')

plt.subplot(2,2,2)

plt.plot(mask\_sizes, rank\_coeffs, 'o-', label='Ранговая фильтрация')

plt.xlabel('Размер маски')

plt.ylabel('Коэффициент корреляции')

plt.grid(True, alpha=0.5)

plt.ylim(-1.0, 1.1)

plt.axhline(y=0.0, color='black', linestyle='--')

plt.legend()

plt.title('Ранговая фильтрация')

plt.subplot(2,2,3)

plt.plot(mask\_sizes, wiener\_coeffs, 'o-', label='Фильтрация Винера')

plt.xlabel('Размер маски')

plt.ylabel('Коэффициент корреляции')

plt.grid(True, alpha=0.5)

plt.ylim(-1.0, 1.1)

plt.axhline(y=0.0, color='black', linestyle='--')

plt.legend()

plt.title('Фильтрация Винера')

plt.show()

f\_img = 10 \* np.log(np.abs(np.fft.fftshift(f\_img)) + 1)

rows, cols = f\_img.shape

center\_x, center\_y = cols // 2, rows // 2

y, x = np.indices((rows, cols))

r = np.sqrt((x - center\_x)\*\*2 + (y - center\_y)\*\*2)

r = r.astype(int)

tbin = np.bincount(r.ravel(), f\_img.ravel())

nr = np.bincount(r.ravel())

radial\_profile = tbin / nr

plt.plot(radial\_profile[:min(rows, cols)//2])

plt.title('Зависимость спектра яркости от частоты')

plt.xlabel('Радиус (пиксели)')

plt.ylabel('Амплитуда')

plt.grid(True, alpha=0.5)

plt.show()

**Вывод:** в ходе выполнения домашней работы были сформированы практические навыки разложения сигналов с использованием дискретного преобразования Фурье (ДПФ).