Лабораторная работа №2 Дискретное преобразование Фурье в **приложениях**

Выполнил Казачинский Глеб, 3 курс 6 группа

Вариант 1. Сжатие изображений

1) постановка задачи (скриншот);

Общая постановка задачи

Написать программу, которая осуществляет сжатие изображений с использованием дискретного преобразования Фурье. Для вычисления ДПФ можно ипользовать как встроенные функции выбранного вами языка программирования, так и самостоятельно реализованный алгоритм БПФ из ЛР №1.

Пусть X — матрица тестового черно-белого изображения (выбирается самостоятельно), $Y = F_n X F_n$ — ДПФ матрицы X, Y^ϵ — матрица, полученная из Y после обнуления всех элементов, по модулю не превосходящих ϵ, X^ϵ — матрица, полученная после применения обратного ДПФ к матрице Y^ϵ. Процент нулевых элементов в матрице Y^ϵ обозначим $Z(\epsilon)$.

1) Экспериментально подобрать значения параметра ϵ_{80}, ϵ_{90}, ϵ_{99}, при которых $Z(\epsilon_k) \approx k$. Для указанных значений ϵ необходимо

- построить «сжатое» изображение, т. е. изображение матрицы X^ϵ;
- вычислить значение PSNR (см. ниже).

Коэффициент PSNR — стандартная количественная оценка искажений для изображений. PSNR расшифровывается как peak signal-to-noise ratio (пиковое отношение сигнал/шум) и вычисляется по формуле

$$\mathrm{PSNR} = 20 \log_{10} \left(\frac{M_{gray}}{\mathrm{rms}} \right), \quad \text{где}$$

- M_{gray} — количество градаций серого;
- $\mathrm{rms} = \sqrt{\frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (X_{ij} - X_{ij}^\epsilon)^2}$ — среднеквадратичное отклонение между точками X_{ij} исходного и точками X_{ij}^ϵ «сжатого» изображения;
- m и n — соответственно число строк и столбцов в матрице изображения.

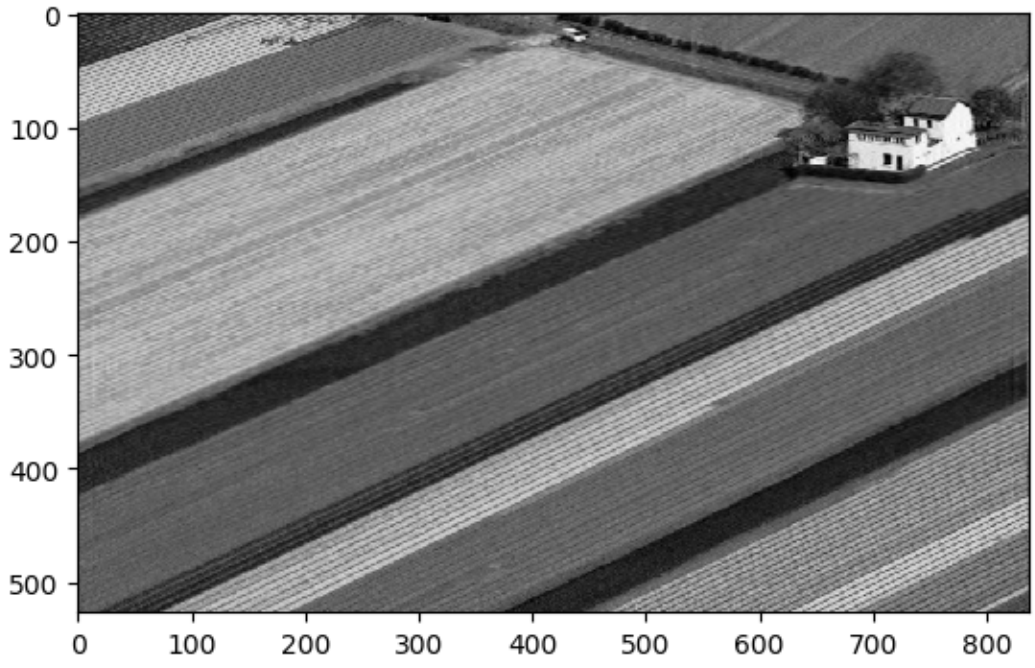
2) Изменяя значение ε так, чтобы $Z(\epsilon)$ изменялось от 0 до 100, построить диаграмму «$Z(\epsilon)$ — PSNR(ϵ)», содержащую как минимум 10 значений. Сделать выводы.

2) тестовое изображение;

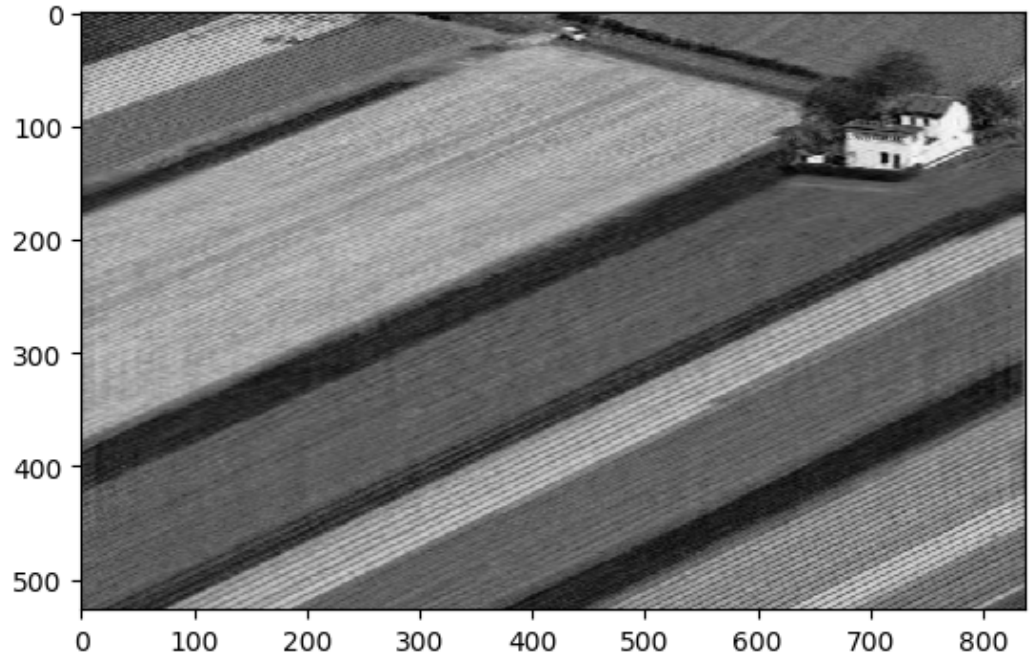
3) значения ϵ_{80}, ϵ_{90}, ϵ_{99}, соответствующие им значения $Z(\epsilon)$, и все изображения из пункта 1;

```
eps_80:  (579.012929496637+81.1937826678975j) Z(eps_80): 80.36347497656388
eps_90:  (1034.1541865016754+1247.729981487967j) Z(eps_90): 90.10859867851984
eps_99:  (5985.335140626678+3386.3955800468716j) Z(eps_99): 98.95046940171095
```

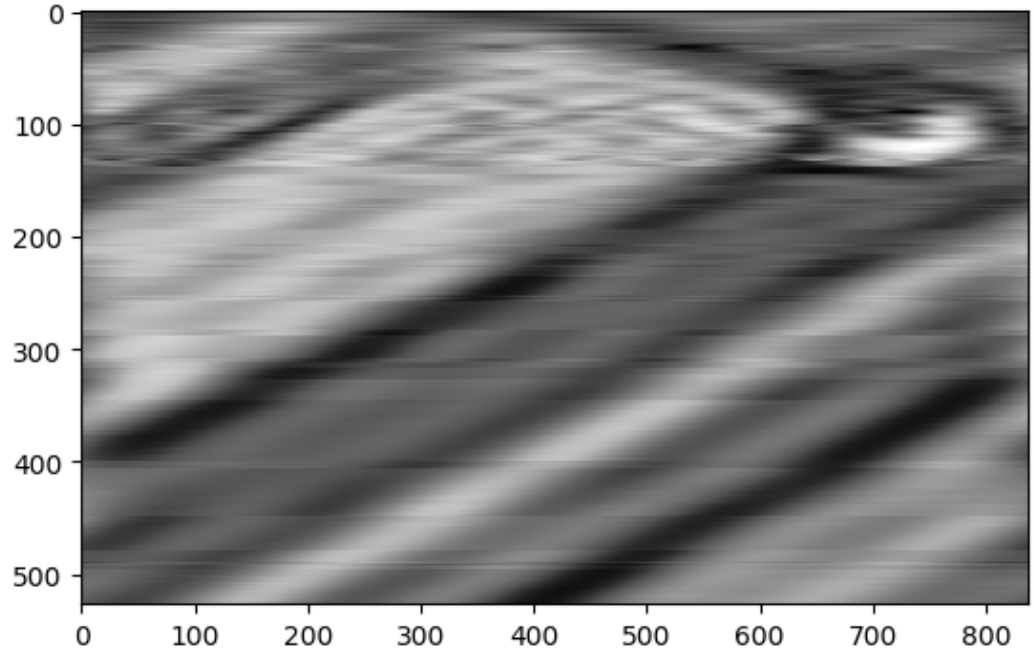
ε_{80}:



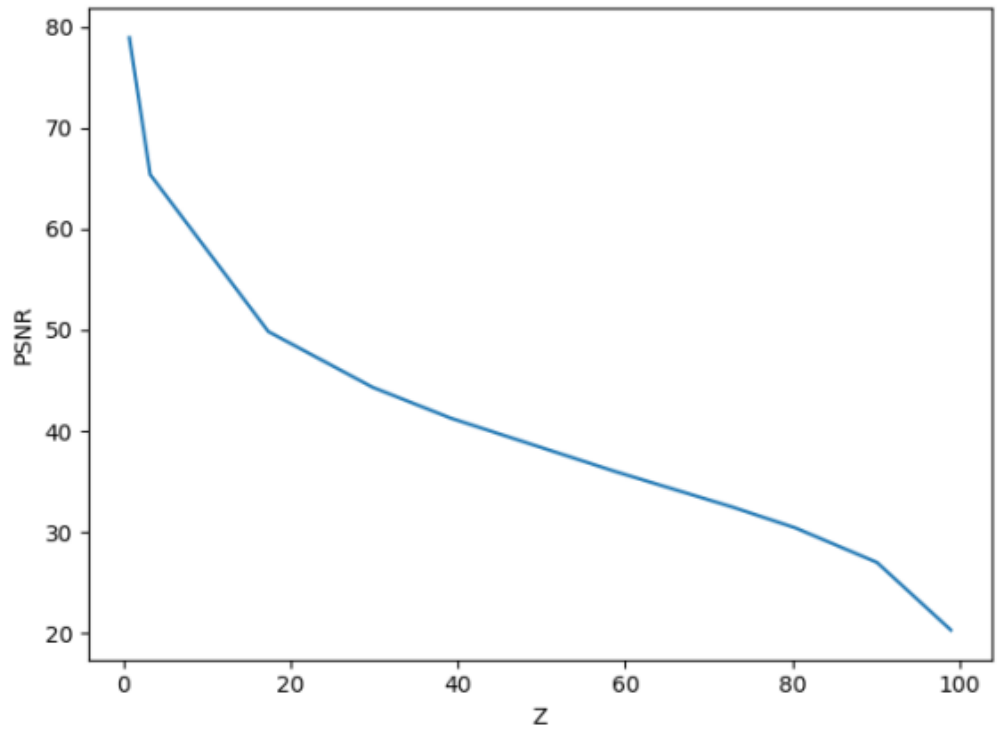
ε_{90}:



ε_{99}:



4) диаграмма из пункта 2;



5) выводы и комментарии;

Как видим на диаграмме “Z(ε) - PSNR” оценка PSNR обратна пропорциональна Z(ε), что соответсвует определению PSNR(”PSNR является инженерным термином, означающим соотношение между максимумом возможного значения сигнала и мощностью шума, искажающего значения сигнала.” - чем меньше искажения, тем больше значение оценки)

Также видим, что с увеличением Z(ε), уменьшается размер изображения - осуществляется сжатие изображения

eps_0.png	Сегодня, 21:39	243 КБ	PNG
eps_3.png	Сегодня, 21:39	243 КБ	PNG
eps_17.png	Сегодня, 21:39	243 КБ	PNG
eps_29.png	Сегодня, 21:40	243 КБ	PNG
eps_39.png	Сегодня, 21:40	241 КБ	PNG
eps_58.png	Сегодня, 21:40	237 КБ	PNG
eps_72.png	Сегодня, 21:40	227 КБ	PNG
eps_80.png	Сегодня, 21:40	221 КБ	PNG
eps_90.png	Сегодня, 21:40	201 КБ	PNG
eps_98.png	Сегодня, 21:40	81 КБ	PNG

6) исходные тексты всех программ.

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import rgb2gray


def fft(x):
    return np.fft.fft(x)


def ifft(x):
    return np.fft.ifft(x)


def Z(m):
    count_zeros = 0
    count_matrix_numbers = len(m[:, 0]) * len(m[0, :])
    for i in range(len(m[:, 0])):
        for j in range(len(m[0, :])):
            if m[i][j] == 0:
                count_zeros += 1
    return count_zeros / count_matrix_numbers * 100


def get_Y_eps(Y, eps):
    n = len(Y[:, 0])
    m = len(Y[0, :])
    Y_eps = Y.copy()
    for i in range(n):
        for j in range(m):
            if abs(Y_eps[i][j]) <= eps:
                Y_eps[i][j] = 0
    return Y_eps


```

```
35. def PSNR(X, X_eps, n, m):
36.     s = 0
37.     for i in range(n):
38.         for j in range(m):
39.             s += (X[i][j] - X_eps[i][j]) ** 2
40.     rms = np.sqrt(s / (m * n))
41.     M_gray = 255
42.     return 20 * np.log10(M_gray / rms)
43.
44.
45. X = np.int32(
46.     rgb2gray(plt.imread(
47.         r'/Users/fpm.kazachin/PycharmProjects/wavelet_analysis/l2/
tulip_fields.jpg')) * 255)  # матрица из интенсивностей серого цвета
48.
49. imgplot_1 = plt.imshow(X, cmap='Greys_r')
50. plt.savefig('compressed_img/source_black.png', bbox_inches='tight')
51.
52. n = len(X[:, 0])
53. m = len(X[0, :])
54.
55. Y = np.zeros((n, m), dtype=complex)
56. for i in range(m):  # применили fft к столбцам X
57.     Y[:, i] = fft(X[:, i])
58.
59. for i in range(n):  # применили fft к строкам X
60.     Y[i, :] = fft(X[i, :])
61.
62. eps_0 = Y[52][60]
63. eps_3 = Y[201][700]
64. eps_17 = Y[21][60]
65. eps_29 = Y[25][100]
66. eps_39 = Y[31][60]
67. eps_58 = Y[21][100]
68. eps_72 = Y[26][60]
69. eps_80 = Y[25][55]
```

```
eps_90 = Y[24][22]
eps_99 = Y[32][10]
eps_arr = [eps_0, eps_3, eps_17, eps_29, eps_39, eps_58, eps_72, eps_80, eps_90, eps_99]

Z_arr = []
PSNR_arr = []
for k in range(len(eps_arr)):
    Y_eps = get_Y_eps(Y, eps_arr[k])

    X_eps = np.zeros((n, m), dtype=complex)
    for i in range(m):  # применили ifft к столбцам Y
        X_eps[:, i] = ifft(Y_eps[:, i])

    for i in range(n):  # применили ifft к строкам Y
        X_eps[i, :] = ifft(Y_eps[i, :])

    imgplot_2 = plt.imshow(np.abs(X_eps), cmap='Greys_r')
    plt.savefig('compressed_img/eps_{0}.png'.format(round(Z(Y_eps))), bbox_inches='tight')

    PSNR_arr.append(np.abs(PSNR(X, X_eps, n, m)))
    Z_arr.append(Z(Y_eps))

    if k == 7:
        print('eps_80: ', eps_arr[k], 'Z(eps_80):', Z_arr[k])
    if k == 8:
        print('eps_90: ', eps_arr[k], 'Z(eps_90):', Z_arr[k])
    if k == 9:
        print('eps_99: ', eps_arr[k], 'Z(eps_99):', Z_arr[k])

plt.figure()
plt.plot(Z_arr, PSNR_arr)
plt.xlabel('Z')
plt.ylabel('PSNR')
plt.show()
```