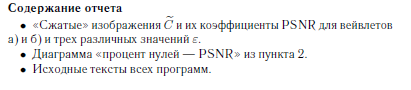
**Лабораторная работа №3**

**«Двумерное дискретное преобразование Хаара»**

Мусорского Павла, студента 3 курса 6б группы, 1 вариант



1. Ход выполнения работы

Задание 0:

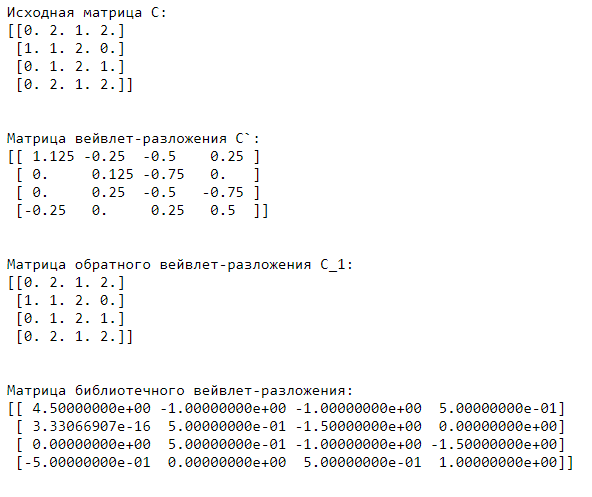
Проверку написанных функций и сравнение с библиотечной функцией буду проводить на матрице-примере из УСР 4:

*C=*

Согласно УСР 4, конечный результат должен быть таков:

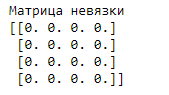
=

Написав программу на языке Python, где реализовал алгоритмы, описанные в УСР 4, в итоге получил следующие результаты:



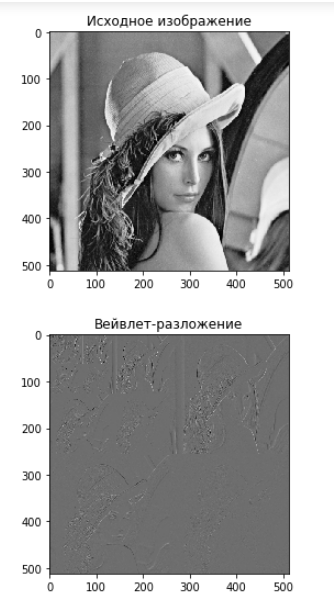
Результаты программы сходятся с ожидаемыми, но отличаются от полученных библиотечной функцией. Можно сделать вывод, что в Питоне вейвлет Хаара реализован несколько иначе, чем было предложено в УСР 4, из-за чего и получилась разница в результатах.

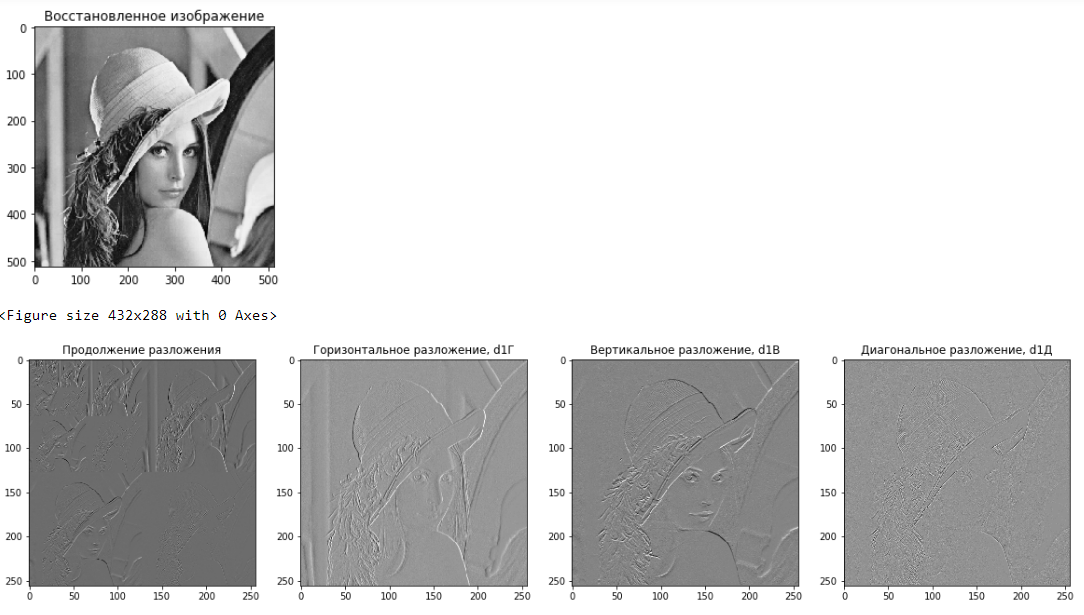
В качестве проверки определю невязку, сравнив матрицы С и С\_1:



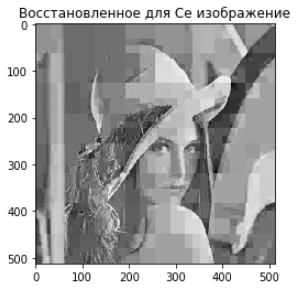
Считаю, можно заключить, что разложение удалось реализовать приемлемо точно, во всяком случае касательно небольших числовых матриц.

Далее проверю, как программа работает с изображениями. Применив ее на заготовленной картинке, получил следующий результат:



Задание 1:

Выбрал В итоге восстановления получил следующее изображение:



Можно заметить, что в отдельных местах картинка стала более смазанной, однако общие детали сохранились, видимо из-за малого значения

***Задание 2:***

Согласно Википедии, для подобных изображений Использовав его, получил следующий результат:

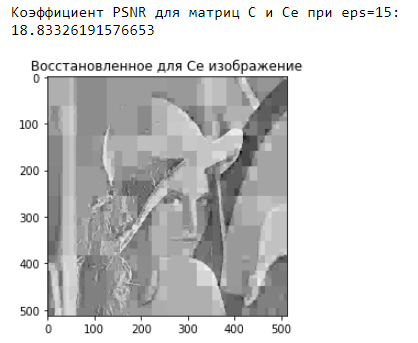
Коэффициент PSNR для матриц С и Сe при eps=10:

22.35717474204823

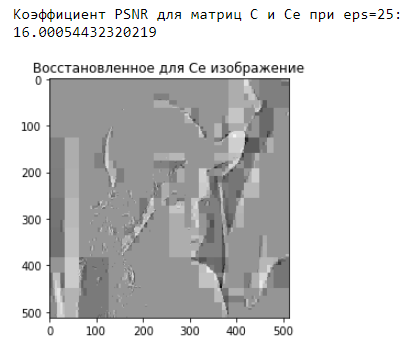
Теперь возьму **,** то есть уменьшу количество «обнуляемых» элементов. Коэффициент PSNR увеличился.



При коэффициент PSNR снизился***.***



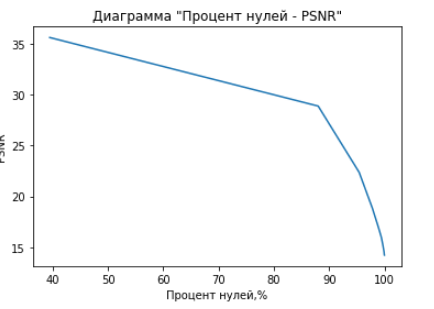
При наблюдается очередное снижение значения коэффициента PSNR.



На основании данных трех измерений, предполагаю, можно выдвинуть гипотезу о наличии некой обратной зависимости между «коэффициентом обнуления» коэффициентом PSNR.

***Задание 3:***

Продолжив эксперимент, который я провел на наборе из 15-и последовательно увеличивающихся значений «коэффициента обнуления» , получил следующий график.



Благодаря графику наличие обратной зависимости между и PSNR стало еще более заметным. Можно считать, что гипотеза, выдвинутая в прошлом пункте, подтвердилась.

2. Листинг программ

import pywt

import numpy as np

from matplotlib import pyplot as plt

import math as m

from matplotlib.pyplot import imread

from skimage.color import rgb2gray

eps=10; # задает "коэффициент обнуления"

def clear(er,r): # обнуляет элементы матрицы для задания 1

dim=len(er);

w=er.copy();

for i in range(dim):

for j in range(dim):

if (np.fabs(w[i][j])<r):

w[i][j]=0;

return w;

def PSNR(x,y):

dim=len(x);

rms=0;

for i in range (dim):

for j in range (dim):

rms=rms+m.pow(x[i][j]-y[i][j],2);

rms=m.sqrt(rms/(dim\*dim));

return 20\*m.log10(255/rms);

def zero\_proc(x): # подсчитывает процент нулей для диаграммы

dim=len(x);

count=0;

for i in range (dim):

for j in range (dim):

if (x[i][j]==0):

count=count+1;

return count/(dim\*dim)\*100;

# реализация алгоритмов вейвлет-разложения

def haar1d(x):

k=int(0.5\*len(x));

f=[];

g=[];

ans=[];

for i in range(k):

f.append(0.5\*(x[2\*i]+x[2\*i+1]));

g.append(0.5\*(x[2\*i]-x[2\*i+1]));

for i in range(k):

ans.append(f[i]);

for i in range(k):

ans.append(g[i]);

return ans;

def haar2d(mat):

dim=int(len(mat));

while (dim>=2):

m=np.zeros((dim,dim),dtype=float);

for i in range(dim):

for j in range(dim):

m[i][j]=mat[i][j];

m1=np.zeros((dim,dim),dtype=float);

for i in range(dim):

w=haar1d(m[i]);

m1[i]=w;

m2=np.zeros((dim,dim),dtype=float);

for i in range(dim):

w=haar1d(m1[:, i]);

m2[:, i]=w;

for i in range(dim):

for j in range (dim):

mat[i][j]=m2[i][j];

dim=int(0.5\*dim);

return mat;

def inv\_haar1d(x):

k=int(0.5\*len(x));

ans=np.zeros(len(x), dtype=float);

for i in range (k):

ans[2\*i]=x[i]+x[i+k];

ans[2\*i+1]=x[i]-x[i+k];

return ans;

def inv\_haar2d(mat):

dim=int(len(mat));

k=2;

while (k<=dim):

m=np.zeros((k,k),dtype=float);

for i in range(k):

for j in range(k):

m[i][j]=mat[i][j];

m1=np.zeros((k,k),dtype=float);

for i in range(k):

w=inv\_haar1d(m[:, i]);

m1[:, i]=w;

m2=np.zeros((k,k),dtype=float);

for i in range(k):

w=inv\_haar1d(m1[i]);

m2[i]=w;

for i in range(k):

for j in range (k):

mat[i][j]=m2[i][j];

k=k\*2;

return mat;

# обработка числовой матрицы

mat = np.array([[0,2,1,2],[1,1,2,0],[0,1,2,1],[0,2,1,2]],dtype=float)

img=mat.copy();

print("Исходная матрица С:")

print(mat);

print();

print();

print("Матрица вейвлет-разложения С`:")

mx=haar2d(mat);

print(mx);

print();

print();

print("Матрица обратного вейвлет-разложения С\_1:")

my=inv\_haar2d(mx);

print(my);

print();

print();

print("Матрица библиотечного вейвлет-разложения:");

dwt = pywt.wavedec2(img, 'haar')

co\_matrix, \_ = pywt.coeffs\_to\_array(dwt);

print(co\_matrix);

print();

print();

dim=len(img);

nev=np.zeros((dim,dim),dtype=float);

for i in range(dim):

for j in range (dim):

nev[i][j]=np.fabs(img[i][j]-my[i][j]);

print("Матрица невязки:");

print(nev);

print();

print();

# обработка изображений

origin=np.int32(rgb2gray(imread(r'test.png')) \* 255);

plt.imshow(origin, cmap='Greys\_r')

plt.title('Исходное изображение:')

orig=origin.copy();

plt.figure()

transform=haar2d(origin)

tr=transform.copy();

plt.imshow(transform, cmap='Greys\_r')

plt.title('Изображение вейвлет-разложения:')

plt.figure()

reverse=inv\_haar2d(transform)

plt.imshow(reverse, cmap='Greys\_r')

plt.title('Восстановленное изображение:')

plt.figure(); # подробнее изобразит результаты первого этапа разложения

dim=len(tr);

w=int(0.5\*dim);

hor=np.zeros((w,w));

ver=np.zeros((w,w));

diag=np.zeros((w,w));

wave=np.zeros((w,w));

for i in range(w):

for j in range(w):

wave[i][j]=tr[i][j];

diag[i][j]=tr[i+w][j+w];

hor[i][j]=tr[i][j+w];

ver[i][j]=tr[i+w][j];

fig, axes = plt.subplots(1, 4)

fig.set\_figwidth(20)

fig.set\_figheight(20)

axes[0].set\_title('Продолжение разложения:')

axes[1].set\_title('Горизонтальное разложение, d1Г:')

axes[2].set\_title('Вертикальное разложение, d1В:')

axes[3].set\_title('Диагональное разложение, d1Д:')

axes[0].imshow(wave, cmap='Greys\_r')

axes[1].imshow(hor, cmap='Greys\_r')

axes[2].imshow(ver, cmap='Greys\_r')

axes[3].imshow(diag, cmap='Greys\_r')

plt.figure();

e=clear(tr,eps);

Ce=inv\_haar2d(e);

plt.imshow(Ce, cmap='Greys\_r')

plt.title('Восстановленное для Сe изображение:')

print("Коэффициент PSNR для матриц С и Сe при eps="+str(eps)+":")

print(PSNR(orig,Ce));

arr\_PSNR=np.zeros(15); # хранит значения коэффициентов PSNR

arr\_proc=np.zeros(15); # хранит значения процентов нулей

for i in range (15):

e=clear(tr,5\*i);

arr\_proc[i]=zero\_proc(e);

Ce=inv\_haar2d(e);

arr\_PSNR[i]=PSNR(orig,Ce);

plt.figure() # нарисует диаграмму

plt.plot(arr\_proc, arr\_PSNR)

plt.xlabel('Процент нулей,%')

plt.ylabel('PSNR')

plt.title('Диаграмма "Процент нулей - PSNR"')

plt.show()