Лабораторная работа №4

Сокрытие данных в частотной области неподвижных изображений на основе кодирования разности абсолютных значений коэффициентов дискретно-косинусного преобразования

по курсу: "Стеганография"
студенты группы КБ-41
Кривич Максим, Науменко Данил
Харьков - 2017г.

In [1]:

```
%matplotlib inline
import re
import copy
import codecs
import numpy as np
import matplotlib.pyplot as plt

from PIL import Image
from math import cos, pi, sqrt
from scipy import fftpack
from functools import lru_cache
from pprint import pprint
```

In [2]:

```
np.set_printoptions(formatter={'float_kind':lambda x: "%.2f" % x})
```

In [3]:

```
IMG_TEMP = 'images/img{}.bmp'
STEG_IMG_TEMP = 'images/stego{}.bmp'
TEXT_FILE = 'text.txt'
MARKER = '0111111111111110'
EOL = '$$'
```

In [4]:

```
def dct(img):
    return fftpack.dct(fftpack.dct(img.T, norm='ortho').T, norm='ortho')

def idct(coefficients):
    return fftpack.idct(fftpack.idct(coefficients.T, norm='ortho').T, norm='ort
```

In [5]:

```
def np_2_image(array):
    try:
        return Image.fromarray(array)
    except:
        return None
```

In [6]:

```
def image_2_np(image):
    try:
        return np.array(image)
    except:
        return None
```

In [7]:

```
def open_image(filename):
    return Image.open(filename)
```

In [8]:

```
def read_text(filename):
    with codecs.open(filename, encoding='utf-8', mode='r') as f:
        return f.read().strip()
```

In [9]:

```
def str_2_bin(*args):
    return ''.join(bin(ord(x))[2:].zfill(8) for x in ''.join(args))
```

In [10]:

```
def bin_2_str(binary, length=8):
    bin_l = [binary[i:i+length] for i in range(0, len(binary), length)]
    return ''.join([chr(int(c, 2)) for c in bin_l])
```

In [11]:

```
def chunks(l, n, step=4):
    for i in range(0, len(l) - n + 1, step):
        yield l[i:i + n]
```

In [12]:

```
def get_reconstructed_image(raw):
    img = raw.clip(0, 255)
    img = img.astype('uint8')
    img = Image.fromarray(img)
    return img
```

In [13]:

```
def get_blocks(img_arr):
    b = list()
    for r in range(0, img_arr.shape[0], 8):
        for c in range(0, img_arr.shape[1], 8):
            b.append(dct(img_arr[r:r + 8, c:c + 8]))
    return b
```

Задание №1 - Реализация алгоритмов прямого и обратного дискретно-косинусного преобразования. Исследование эффекта частотной чувствительности зрительной системы человека

In [14]:

```
# @lru_cache(maxsize=128)
def C(x):
    if x == 0:
        return 1 / sqrt(2)
    elif x > 0:
        return 1
```

In [15]:

```
"""

[23.11.2017] Этот код понимают только два человека Бог и Я
[25.11.2017] Только Бог

"""
N = 8
P = 1

def dct2d(matrix):
    coeff = matrix.copy()
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            coeff[i, j] = (C(i)*C(j)/sqrt(2*N))*sum([sum([matrix[x,y]*cos((2*i+
                                                                           *cos((2*j+
                                                                         for y in ra
    return coeff
dct2d.__code__ = dct.__code__
def idct2d(coeff):
    matrix = coeff.copy()
    for i in range(len(coeff)):
        for j in range(len(coeff[0])):
            matrix[i, j] = 1.0 / sqrt(2.0*N)*sum([sum([ C(x)*C(y)*cos((2*i+1)*p
                                                                        *cos((2*j+1)*p
                                                                      for y in range(N
    return matrix
idct2d.__code = idct.__code__
```

In [16]:

```
image = open_image(IMG_TEMP.format(1))
red, green, blue = image.split()
red
```

Out[16]:

In [17]:

```
red_np = image_2_np(red)
```

In [18]:

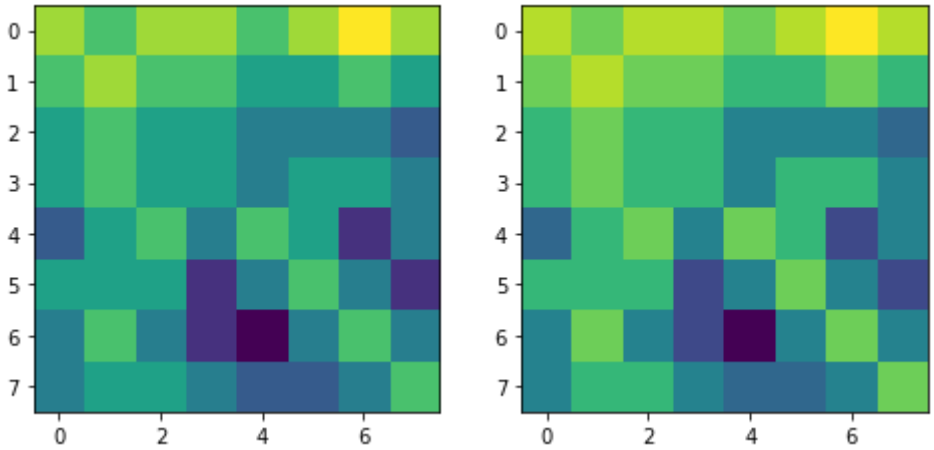
```
print('='*20 + 'DCT(1,1)'+'='*19)
print(re.sub('[ ]+', ' ', re.sub(' *[\\[\\]] *', '', np.array_str(dct2d(red_np[
print('='*45)
```

```
====================DCT(1,1)==================
318.63 2.18 1.18 -0.54 -3.13 -0.39 -0.08 -0.31
6.28 0.32 -1.53 0.38 1.28 1.30 -0.64 -0.20
2.99 -1.58 2.55 -1.30 0.35 1.73 -0.67 -0.11
1.66 -0.88 -0.68 1.84 -1.06 -0.30 1.39 0.87
2.12 -1.34 -1.13 -1.49 1.87 -0.54 0.87 0.12
0.15 -0.67 1.95 0.73 -1.43 1.75 -0.86 -0.63
-0.29 0.55 -0.92 0.97 0.53 0.06 0.95 0.44
-1.00 -0.17 -0.66 -1.05 0.36 -0.83 0.13 0.09
=============================================
```

In [19]:

```
a = dct(red_np[:N,:N])
b = a + 0.3*a # ~ 30%
print('='*20 + 'DCT(a)'+'='*19)
print(re.sub('[ ]+', ' ', re.sub(' *[\\[\\]] *', '', np.array_str(a))))
print('='*20 + 'DCT(b)'+'='*19)
print(re.sub('[ ]+', ' ', re.sub(' *[\\[\\]] *', '', np.array_str(b))))
print('='*45)

f,ax = plt.subplots(1,2, figsize=(8, 8))
ax[0].imshow(get_reconstructed_image(idct(a)))
ax[1].imshow(get_reconstructed_image(idct(b)))
plt.show()
```

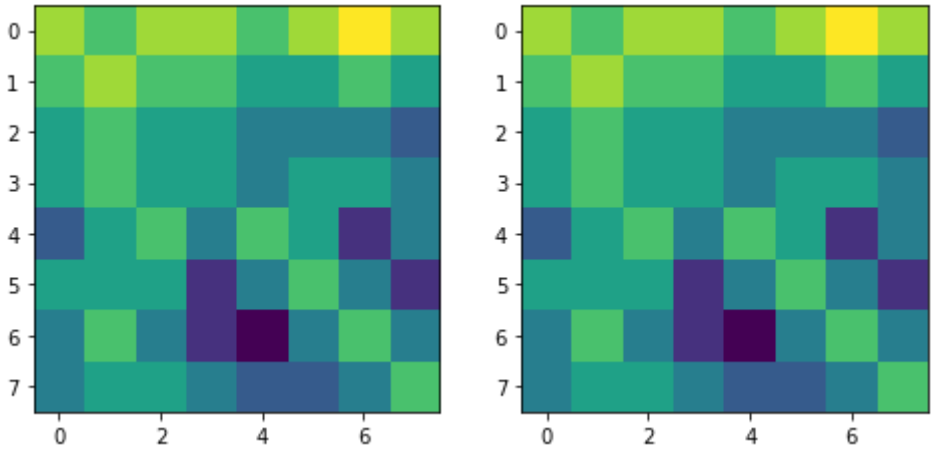
```
====================DCT(a)====================
318.63 2.18 1.18 -0.54 -3.13 -0.39 -0.08 -0.31
6.28 0.32 -1.53 0.38 1.28 1.30 -0.64 -0.20
2.99 -1.58 2.55 -1.30 0.35 1.73 -0.67 -0.11
1.66 -0.88 -0.68 1.84 -1.06 -0.30 1.39 0.87
2.12 -1.34 -1.13 -1.49 1.87 -0.54 0.87 0.12
0.15 -0.67 1.95 0.73 -1.43 1.75 -0.86 -0.63
-0.29 0.55 -0.92 0.97 0.53 0.06 0.95 0.44
-1.00 -0.17 -0.66 -1.05 0.36 -0.83 0.13 0.09
====================DCT(b)====================
414.21 2.84 1.54 -0.71 -4.06 -0.51 -0.11 -0.40
8.16 0.42 -2.00 0.50 1.66 1.69 -0.83 -0.26
3.89 -2.05 3.31 -1.70 0.46 2.25 -0.87 -0.14
2.15 -1.15 -0.88 2.40 -1.38 -0.40 1.80 1.13
2.76 -1.74 -1.46 -1.93 2.44 -0.71 1.13 0.15
0.20 -0.86 2.54 0.94 -1.86 2.27 -1.11 -0.82
-0.38 0.72 -1.20 1.26 0.69 0.08 1.24 0.57
-1.30 -0.22 -0.85 -1.36 0.47 -1.07 0.17 0.12
=============================================
```



In [20]:

```
a = dct(red_np[:N,:N])
b = a + a # ~ 100%
print('='*20 + 'DCT(a)'+'='*19)
print(re.sub('[ ]+', ' ', re.sub(' *[\\[\\]] *', '', np.array_str(a))))
print('='*20 + 'DCT(b)'+'='*19)
print(re.sub('[ ]+', ' ', re.sub(' *[\\[\\]] *', '', np.array_str(b))))
print('='*45)

f,ax = plt.subplots(1,2, figsize=(8, 8))
ax[0].imshow(get_reconstructed_image(idct(a)))
ax[1].imshow(get_reconstructed_image(idct(b)))
plt.show()
```

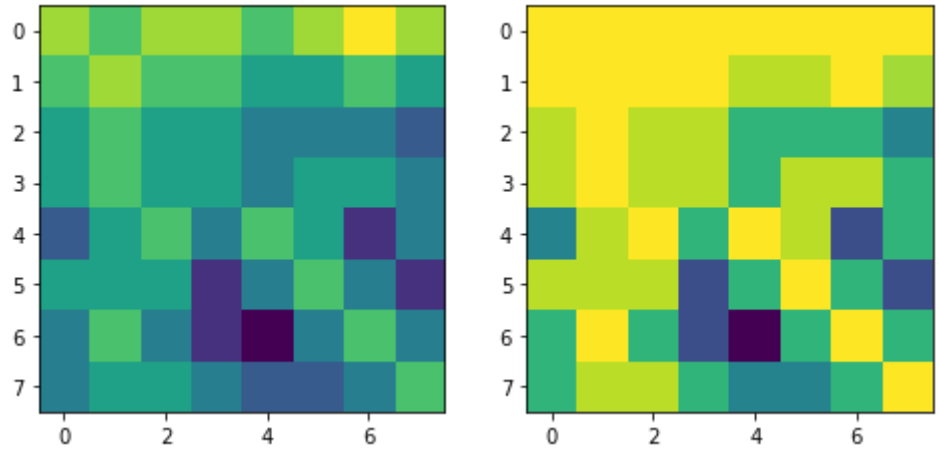
```
====================DCT(a)==================
318.63 2.18 1.18 -0.54 -3.13 -0.39 -0.08 -0.31
6.28 0.32 -1.53 0.38 1.28 1.30 -0.64 -0.20
2.99 -1.58 2.55 -1.30 0.35 1.73 -0.67 -0.11
1.66 -0.88 -0.68 1.84 -1.06 -0.30 1.39 0.87
2.12 -1.34 -1.13 -1.49 1.87 -0.54 0.87 0.12
0.15 -0.67 1.95 0.73 -1.43 1.75 -0.86 -0.63
-0.29 0.55 -0.92 0.97 0.53 0.06 0.95 0.44
-1.00 -0.17 -0.66 -1.05 0.36 -0.83 0.13 0.09
====================DCT(b)==================
637.25 4.37 2.37 -1.09 -6.25 -0.78 -0.17 -0.62
12.56 0.64 -3.07 0.76 2.56 2.60 -1.27 -0.40
5.98 -3.15 5.09 -2.61 0.71 3.46 -1.34 -0.21
3.31 -1.76 -1.36 3.69 -2.12 -0.61 2.77 1.74
4.25 -2.68 -2.25 -2.98 3.75 -1.09 1.75 0.23
0.31 -1.33 3.90 1.45 -2.87 3.50 -1.71 -1.26
-0.58 1.10 -1.84 1.94 1.06 0.12 1.91 0.88
-2.00 -0.34 -1.31 -2.10 0.72 -1.65 0.27 0.18
=============================================
```



In [21]:

```
a = dct(red_np[:N,:N])
b = a + 5.3*a # ~530%
print('='*20 + 'DCT(a)'+'='*19)
print(re.sub('[ ]+', ' ', re.sub(' *[\\[\\]] *', '', np.array_str(a))))
print('='*20 + 'DCT(b)'+'='*19)
print(re.sub('[ ]+', ' ', re.sub(' *[\\[\\]] *', '', np.array_str(b))))
print('='*45)

f,ax = plt.subplots(1,2, figsize=(8, 8))
ax[0].imshow(get_reconstructed_image(idct(a)))
ax[1].imshow(get_reconstructed_image(idct(b)))
plt.show()
```

```
====================DCT(a)==================
318.63 2.18 1.18 -0.54 -3.13 -0.39 -0.08 -0.31
6.28 0.32 -1.53 0.38 1.28 1.30 -0.64 -0.20
2.99 -1.58 2.55 -1.30 0.35 1.73 -0.67 -0.11
1.66 -0.88 -0.68 1.84 -1.06 -0.30 1.39 0.87
2.12 -1.34 -1.13 -1.49 1.87 -0.54 0.87 0.12
0.15 -0.67 1.95 0.73 -1.43 1.75 -0.86 -0.63
-0.29 0.55 -0.92 0.97 0.53 0.06 0.95 0.44
-1.00 -0.17 -0.66 -1.05 0.36 -0.83 0.13 0.09
====================DCT(b)==================
2007.34 13.76 7.45 -3.42 -19.69 -2.45 -0.53 -1.94
39.55 2.02 -9.67 2.40 8.05 8.19 -4.00 -1.25
18.84 -9.92 16.04 -8.22 2.23 10.90 -4.22 -0.67
10.44 -5.56 -4.29 11.61 -6.67 -1.92 8.73 5.50
13.39 -8.43 -7.10 -9.38 11.81 -3.42 5.50 0.73
0.96 -4.19 12.29 4.58 -9.03 11.01 -5.39 -3.98
-1.84 3.47 -5.80 6.11 3.34 0.37 6.01 2.76
-6.31 -1.06 -4.13 -6.61 2.27 -5.21 0.84 0.56
=============================================
```



Задание №2 - Реализация алгоритмов встраивания и извлечения сообщений в частотную область изображений (метод Коха-Жао)

Входные параметры

In [22]:

```
N = 8
Pr = 30
message = "Стеганография"
```

In [23]:

```
def H(h1, h2, Pr=Pr):
    if abs(h1) - abs(h2) > Pr:
        return '1'
    if abs(h1) - abs(h2) < -Pr:
        return '0'
    else:
        return -1
```

Процедура вставки сообщения

In [24]:

```
def encode(image, new_image, m, Pr=Pr):
    red, green, blue = image.split()
    red = np.array(red)
    new_red = np.array(new_image.split()[0])
    x_bin = ''
    for c in m:
        x_bin += "{0:016b}".format(ord(c))
    x_bin += MARKER
    s = 0
    for r in range(0, red.shape[0], 8):
        for c in range(0, red.shape[1], 8):
            block = dct(red[r:r + 8, c:c + 8])
            first = block[1][3]
            second = block[3][1]
            if x_bin[s] == '1' and (m != H(first, second, Pr) or H(first, secon
                if first > 0:
                    block[3][1] = abs(second) + Pr
                else:
                    block[3][1] = - abs(second) - Pr
            elif x_bin[s] == '0' and (m != H(first, second, Pr) or H(first, sec
                if second > 0:
                    block[1][3] = abs(first) + Pr
                else:
                    block[1][3] = - abs(first) - Pr
            s += 1
            new_red[r:r+8,c:c+8] = idct(block)
            if s >= len(x_bin):
                return new_red
    return new_red
```

Процедура извлечения сообщения

In [25]:

```
def decode(blocks):
    ch = ''
    res = ''
    for i in range(len(blocks)):
        if abs(blocks[i].item((3, 1))) > abs(blocks[i].item((1, 3))):
            ch += '1'
        elif abs(blocks[i].item((3, 1))) <= abs(blocks[i].item((1, 3))):
            ch += '0'
        if len(ch) == 16:
            if ch == MARKER:
                return res
            try:
                res += chr(int(ch, 2))
                ch = ''
            except UnicodeEncodeError:
                return res
    return res
```

Тестирование

In [26]:

```
with Image.open(IMG_TEMP.format(2)) as image:
    new_image = image.copy()
    r, g, b = image.split()
    new_data = encode(image, new_image, message)
    Image.merge("RGB", (Image.fromarray(new_data), g, b)).save(STEG_IMG_TEMP.fo
print("Исходное сообщение: {}".format(message))
```

```
Исходное сообщение: Стеганография
```

In [27]:

```
with Image.open(STEG_IMG_TEMP.format(2)) as new_image:
    data = np.array(new_image.split()[0])
blocks_dct = get_blocks(data)
new_message = decode(blocks_dct)
print("Полученное сообщение: {}".format(new_message))
```

```
Полученное сообщение: Стеганография
```

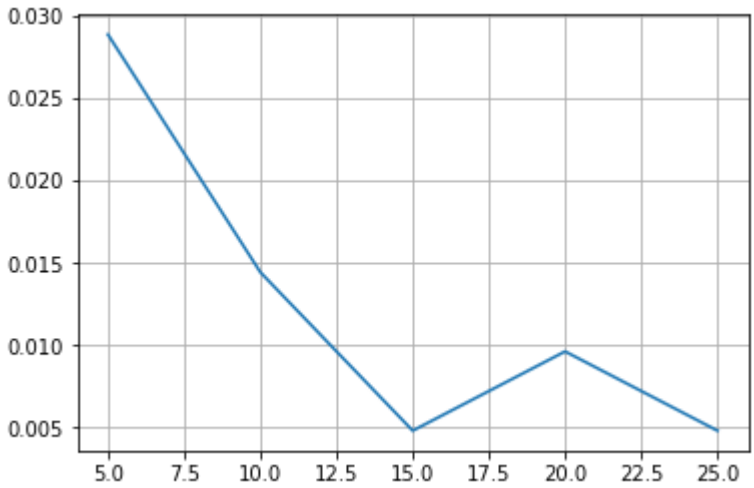
Оценка вероятности ложного изъятия информационных данных

In [28]:

```
X = []
Y = []
for i in range(5, 30, 5):
    with Image.open(IMG_TEMP.format(2)) as image:
        new_image = image.copy()
        r, g, b = image.split()
        new_data = encode(image, new_image, message, Pr=i)
        Image.merge("RGB", (Image.fromarray(new_data), g, b)).save(STEG_IMG_TEM

    with Image.open(STEG_IMG_TEMP.format(2)) as new_image:
        data = np.array(new_image.split()[0])
    blocks_dct = get_blocks(data)
    new_message = decode(blocks_dct)


    m_bin = ''
    for c in message:
        m_bin = m_bin + "{0:016b}".format(ord(c))
    m_new_bin = ''
    for c in new_message:
        m_new_bin = m_new_bin + "{0:016b}".format(ord(c))

    v = 0
    for j in range(len(m_bin)):
        if m_bin[j] != m_new_bin[j]:
            v += 1
    v /= len(m_bin)
    X.append(v)
    Y.append(i)
```

In [29]:

```
plt.plot(Y, X)
plt.grid(True)
plt.gca().xaxis.grid(True, which='minor')
plt.show()
```



Задание №3 - Реализация стеганоатакы на основе использования алгоритма сжатия JPEG и исследования его возможностей

Задача №4 - Реализация усовершенствованных алгоритмов встраивания и извлечения сообщений в частотную область изображений (метод Бенгама-Мемон-Эо-Юнга)