

Лабораторная работа №4

Соккрытие 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]:

```
1 %matplotlib inline
2 import re
3 import copy
4 import codecs
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 from PIL import Image
9 from math import cos, pi, sqrt
10 from scipy import fftpack
11 from functools import lru_cache
12 from pprint import pprint
```

In [2]:

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

In [3]:

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

In [4]:

```
1 def dct(img):
2     return fftpack.dct(fftpack.dct(img.T, norm='ortho').T, norm='ortho')
3
4 def idct(coefficients):
5     return fftpack.idct(fftpack.idct(coefficients.T, norm='ortho').T, norm='ort
```

In [5]:

```
1 def np_2_image(array):
2     try:
3         return Image.fromarray(array)
4     except:
5         return None
```

In [6]:

```
1 def image_2_np(image):
2     try:
3         return np.array(image)
4     except:
5         return None
```

In [7]:

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

In [8]:

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

In [9]:

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

In [10]:

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

In [11]:

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

In [12]:

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

In [13]:

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

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

In [14]:

```

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

```

In [15]:

```

1 """
2
3 [23.11.2017] Этот код понимают только два человека Бог и Я
4 [25.11.2017] Только Бог
5
6 """
7 N = 8
8 P = 1
9
10 def dct2d(matrix):
11     coeff = matrix.copy()
12     for i in range(len(matrix)):
13         for j in range(len(matrix[0])):
14             coeff[i, j] = (C(i)*C(j)/sqrt(2*N))*sum([sum([matrix[x,y]*cos((2*i+1)*p
15                                                         *cos((2*j+1)*p
16                                                         for y in range(N
17     return coeff
18 dct2d.__code__ = dct.__code__
19 def idct2d(coeff):
20     matrix = coeff.copy()
21     for i in range(len(coeff)):
22         for j in range(len(coeff[0])):
23             matrix[i, j] = 1.0 / sqrt(2.0*N)*sum([sum([ C(x)*C(y)*cos((2*i+1)*p
24                                                         *cos((2*j+1)*p
25                                                         for y in range(N
26     return matrix
27 idct2d.__code__ = idct.__code__

```

In [16]:

```

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

```

Out[16]:



In [17]:

```

1 red_np = image_2_np(red)

```

In [18]:

```

1 print('='*20 + 'DCT(1,1)'+ '='*19)
2 print(re.sub('[ ]+', ' ', re.sub(' *[\n\\] *', '', np.array_str(dct2d(red_np[
3 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]:

```

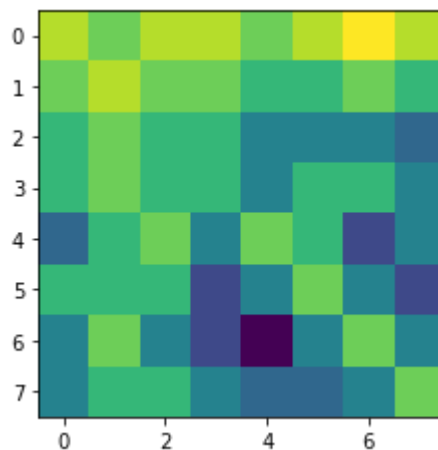
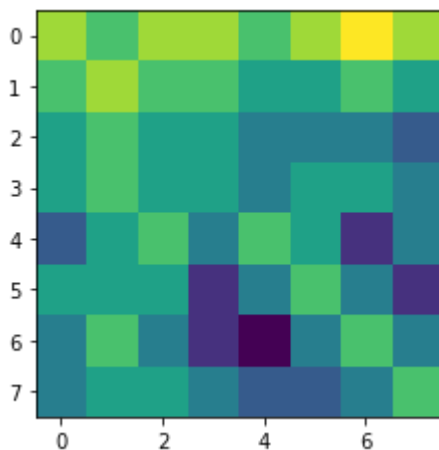
1 a = dct(red_np[:N,:N])
2 b = a + 0.3*a # ~ 30%
3 print('='*20 + 'DCT(a)'+ '='*19)
4 print(re.sub('[ ]+', ' ', re.sub(' *[\n[\]] *', '', np.array_str(a))))
5 print('='*20 + 'DCT(b)'+ '='*19)
6 print(re.sub('[ ]+', ' ', re.sub(' *[\n[\]] *', '', np.array_str(b))))
7 print('='*45)
8
9 f,ax = plt.subplots(1,2, figsize=(8, 8))
10 ax[0].imshow(get_reconstructed_image(idct(a)))
11 ax[1].imshow(get_reconstructed_image(idct(b)))
12 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]:

```

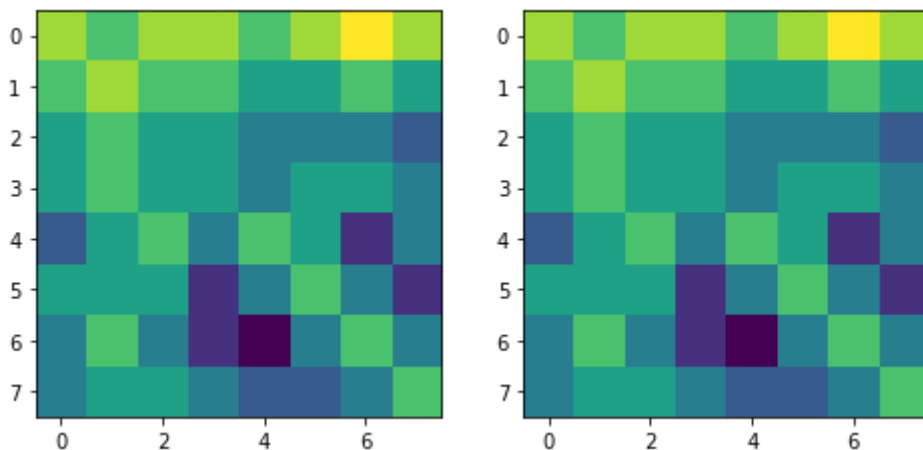
1 a = dct(red_np[:N,:N])
2 b = a + a # ~ 100%
3 print('='*20 + 'DCT(a)'+ '='*19)
4 print(re.sub('[ ]+', ' ', re.sub(' *[\n[\]] *', '', np.array_str(a))))
5 print('='*20 + 'DCT(b)'+ '='*19)
6 print(re.sub('[ ]+', ' ', re.sub(' *[\n[\]] *', '', np.array_str(b))))
7 print('='*45)
8
9 f,ax = plt.subplots(1,2, figsize=(8, 8))
10 ax[0].imshow(get_reconstructed_image(idct(a)))
11 ax[1].imshow(get_reconstructed_image(idct(b)))
12 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]:

```

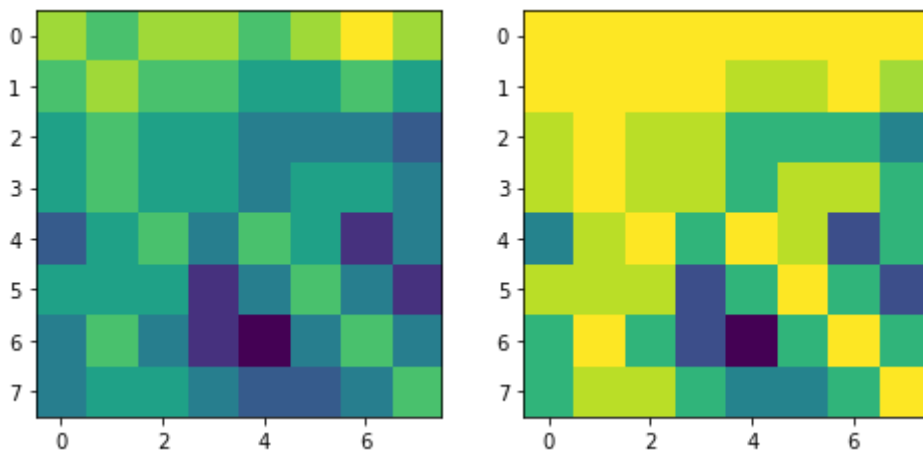
1 a = dct(red_np[:N,:N])
2 b = a + 5.3*a # ~530%
3 print('='*20 + 'DCT(a)'+ '='*19)
4 print(re.sub('[ ]+', ' ', re.sub(' *\\[\\[\\[\\[\\[ ] *', '', np.array_str(a))))
5 print('='*20 + 'DCT(b)'+ '='*19)
6 print(re.sub('[ ]+', ' ', re.sub(' *\\[\\[\\[\\[\\[ ] *', '', np.array_str(b))))
7 print('='*45)
8
9 f,ax = plt.subplots(1,2, figsize=(8, 8))
10 ax[0].imshow(get_reconstructed_image(idct(a)))
11 ax[1].imshow(get_reconstructed_image(idct(b)))
12 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]:

```
1 N = 8
2 Pr = 30
3 message = "Стеганография"
```

In [23]:

```
1 def H(h1, h2, Pr=Pr):
2     if abs(h1) - abs(h2) > Pr:
3         return '1'
4     if abs(h1) - abs(h2) < -Pr:
5         return '0'
6     else:
7         return -1
```

Процедура вставки сообщения

In [24]:

```
1 def encode(image, new_image, m, Pr=Pr):
2     red, green, blue = image.split()
3     red = np.array(red)
4     new_red = np.array(new_image.split()[0])
5     x_bin = ''
6     for c in m:
7         x_bin += "{0:016b}".format(ord(c))
8     x_bin += MARKER
9     s = 0
10    for r in range(0, red.shape[0], 8):
11        for c in range(0, red.shape[1], 8):
12            block = dct(red[r:r + 8, c:c + 8])
13            first = block[1][3]
14            second = block[3][1]
15            if x_bin[s] == '1' and (m != H(first, second, Pr) or H(first, second, Pr) == '1'):
16                if first > 0:
17                    block[3][1] = abs(second) + Pr
18                else:
19                    block[3][1] = - abs(second) - Pr
20            elif x_bin[s] == '0' and (m != H(first, second, Pr) or H(first, second, Pr) == '0'):
21                if second > 0:
22                    block[1][3] = abs(first) + Pr
23                else:
24                    block[1][3] = - abs(first) - Pr
25            s += 1
26            new_red[r:r+8,c:c+8] = idct(block)
27            if s >= len(x_bin):
28                return new_red
29    return new_red
```

Процедура извлечения сообщения

In [25]:

```
1 def decode(blocks):
2     ch = ''
3     res = ''
4     for i in range(len(blocks)):
5         if abs(blocks[i].item((3, 1))) > abs(blocks[i].item((1, 3))):
6             ch += '1'
7         elif abs(blocks[i].item((3, 1))) <= abs(blocks[i].item((1, 3))):
8             ch += '0'
9         if len(ch) == 16:
10             if ch == MARKER:
11                 return res
12             try:
13                 res += chr(int(ch, 2))
14                 ch = ''
15             except UnicodeEncodeError:
16                 return res
17     return res
```

Тестирование

In [26]:

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

Исходное сообщение: Стеганография

In [27]:

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

Полученное сообщение: Стеганография

Оценка вероятности ложного изъятия информационных данных

In [28]:

```

1 X = []
2 Y = []
3 for i in range(5, 30, 5):
4     with Image.open(IMG_TEMP.format(2)) as image:
5         new_image = image.copy()
6         r, g, b = image.split()
7         new_data = encode(image, new_image, message, Pr=i)
8         Image.merge("RGB", (Image.fromarray(new_data), g, b)).save(STEG_IMG_TEM
9
10    with Image.open(STEG_IMG_TEMP.format(2)) as new_image:
11        data = np.array(new_image.split()[0])
12        blocks_dct = get_blocks(data)
13        new_message = decode(blocks_dct)
14
15
16    m_bin = ''
17    for c in message:
18        m_bin = m_bin + "{0:016b}".format(ord(c))
19    m_new_bin = ''
20    for c in new_message:
21        m_new_bin = m_new_bin + "{0:016b}".format(ord(c))
22
23    v = 0
24    for j in range(len(m_bin)):
25        if m_bin[j] != m_new_bin[j]:
26            v += 1
27    v /= len(m_bin)
28    X.append(v)
29    Y.append(i)

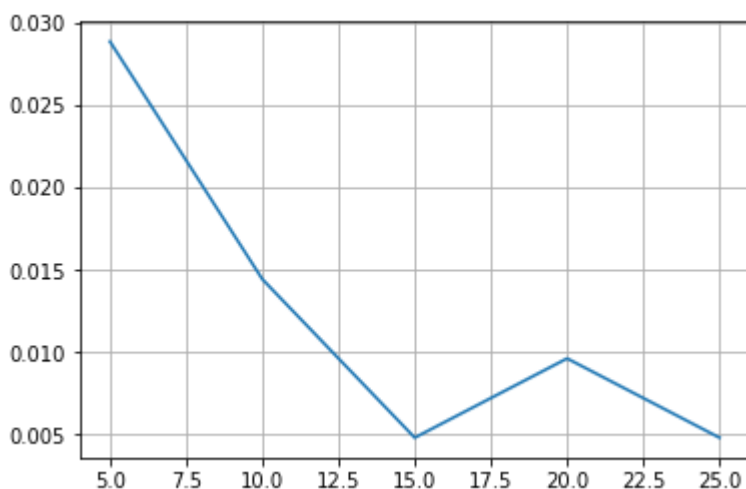
```

In [29]:

```

1 plt.plot(Y, X)
2 plt.grid(True)
3 plt.gca().xaxis.grid(True, which='minor')
4 plt.show()

```



Задание №3 - Реализация стеганоатаки 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)