**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Основы стеганографии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

“ Сокрытие информации в коэффициентах дискретного косинусного преобразования методом LSB”

Выполнил студент группы N3349

Ле Тиен Зунг



Проверил: ассистент ФБИТ,

Университет ИТМО,

Давыдов Вадим Валерьевич

Санкт-Петербург

2020

# Цель работы

Встраивание текстовой информации в картинки, используя метода LSB-DCT. Извлечение тектовой информации из стегоконтейнера.

Построение графика значений PSNR исходного изображения и изображения с информацией, встроенные по наростающей одно слово, пять, десять.

Оценка целесообразности метода с реальными примерами.

**Теоретическая часть:**

Стеганография – способ передачи или хранения информации с учётом сохранения в тайне самого факта такой передачи (хранения). Цифровая стеганография — направление классической стеганографии, основанное на сокрытии или внедрении дополнительной информации в цифровые объекты, вызывая при этом некоторые искажения этих объектов. В данной лабораторной работе был рассмотрен метод НЗБ с изображенным объектом. ([1] [https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%B3%D0%B0%D0%BD%D0%B](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%B3%D0%B0%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)

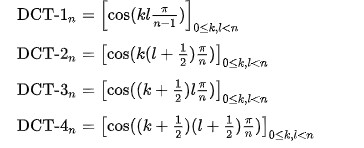
[E%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%B3%D0%B0%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) )

НЗБ (наименьший значимый бит) - суть этого метода заключается в замене последних значащих битов в контейнере (изображения) на биты скрываемого сообщения. Разница между пустым и заполненным контейнерами должна быть не ощутима для органов восприятия человека. ([2[] http://www.nestego.ru/2012/07/lsb.html)](http://www.nestego.ru/2012/07/lsb.html)

Дискретное косинусное преобразование (англ. Discrete Cosine Transform, DCT) — одно из ортогональных преобразований. Вариант косинусного преобразования для вектора действительных чисел. Применяется в алгоритмах сжатия информации с потерями, например, MPEG и JPEG. Это преобразование тесно связано с дискретным преобразованием Фурье и является гомоморфизмом его векторного пространства.

Математически преобразование можно осуществить умножением вектора на матрицу преобразования. При этом матрица обратного преобразования с точностью до множителя равна транспонированной матрице. В математике матрицы выбирают так, чтобы преобразование было ортонормированным, а постоянный множитель равен единице. В компьютерных приложениях это не всегда так.

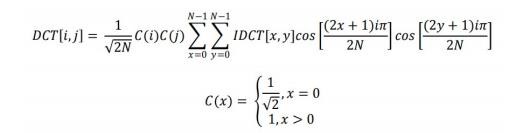
Различные периодические продолжения сигнала ведут к различным типам ДКП. Ниже приводятся матрицы для первых четырёх типов ДКП:



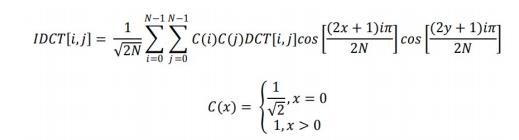
([3] <https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D1%81%D0%B8%D0%BD%D1%83%D1%81%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5> )

В этом работе, метод замены наименьший значащего бита выполнен с картинками в формате .bmp. BMP – это формат картинок, с помощью которого сохраняются только растровые изображения, а векторные нет. С английского языка слово переводится, как «Bitmap Picture» или BMP, что значит формат для хранения растровых изображений. ([5]<https://biz-iskun.ru/format-bmp-chto-eto-takoe.html>)

Формула дискретного косинусного преобразования:



Формула обратного дискретного косинусного преобразования :



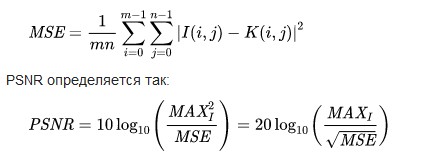
([4] [https://math.spbu.ru/SD\_AIS/documents/2013-12-341/2013-12-tw-14.pdf)](https://math.spbu.ru/SD_AIS/documents/2013-12-341/2013-12-tw-14.pdf)

Для того, чтобы встраивать исходное сообщение, оно преобразуется в двоичный код. Далее двоичная последовательность записывается указанным методом. Извлечение информации происходит следующим образом: исходя из метода НЗБ изображенный контейнер считывается и записывает полученную последовательность битов. Далее двоичный код преобразуется в зашифрованное сообщение. ([6] [https://www.chaynikam.info/dvoichnoe-kodirovanie.html)](https://www.chaynikam.info/dvoichnoe-kodirovanie.html)

PSNR -Пиковое отношение сигнала к шуму обозначается аббревиатурой PSNR и является инженерным термином, означающим соотношение между максимумом возможного значения сигнала и мощностью шума, искажающего значения сигнала. Поскольку многие сигналы имеют широкий динамический диапазон, PSNR обычно измеряется в логарифмической шкале в децибелах.

PSNR наиболее часто используется для измерения уровня искажений при сжатии изображений. Проще всего его определить через среднеквадратичную ошибку (СКО) или MSE.

Рассмотрим формулы как считается PSNR и MSE, а затем построим график.



* I,K: сравниваемые картинки
* i,j: расположение пикселей
* m,n: ширина и высота сравниваемых картинок.
* MAXi-максимальное значение, принимаемое пикселем изображения. Когда пиксели имеют разрядность 8 бит ,MAXi=255 .

([7]

[https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D0%BE%D0%B2%D0%BE %D0%B5\_%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D0%B0_%D0%BA_%D1%88%D1%83%D0%BC%D1%83)

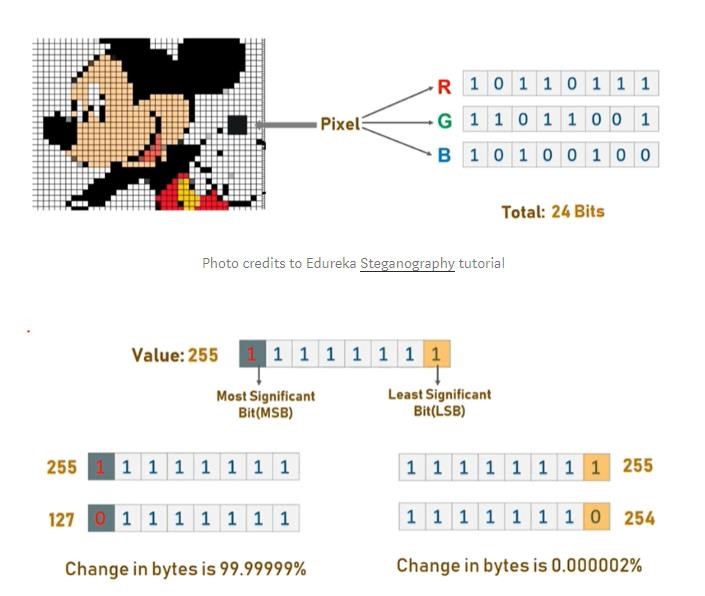
[D0%B5\_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D0%B0\_%D0%BA\_%](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D0%B0_%D0%BA_%D1%88%D1%83%D0%BC%D1%83)

[D1%88%D1%83%D0%BC%D1%83](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D0%B0_%D0%BA_%D1%88%D1%83%D0%BC%D1%83) )

**Практическая часть**

**Описание метода**

Суть метода заключается в следующем: Допустим, имеется 8-битное изображение в градациях серого. 00h (00000000b) обозначает чёрный цвет, FFh (11111111b) — белый. Всего имеется 256 градаций . Также предположим, что сообщение состоит из 1 байта — например, 01101011b. При использовании 2 младших бит в описаниях пикселей, нам потребуется 4 пикселя. Допустим, они чёрного цвета. Тогда пиксели, содержащие скрытое сообщение, будут выглядеть следующим образом: 00000001 00000010 00000010 00000011. Тогда цвет пикселей изменится: первого — на 1/255, второго и третьего — на 2/255 и четвёртого — на 3/255. Такие градации, мало того что незаметны для человека, могут вообще не отобразиться при использовании низкокачественных устройств вывода.

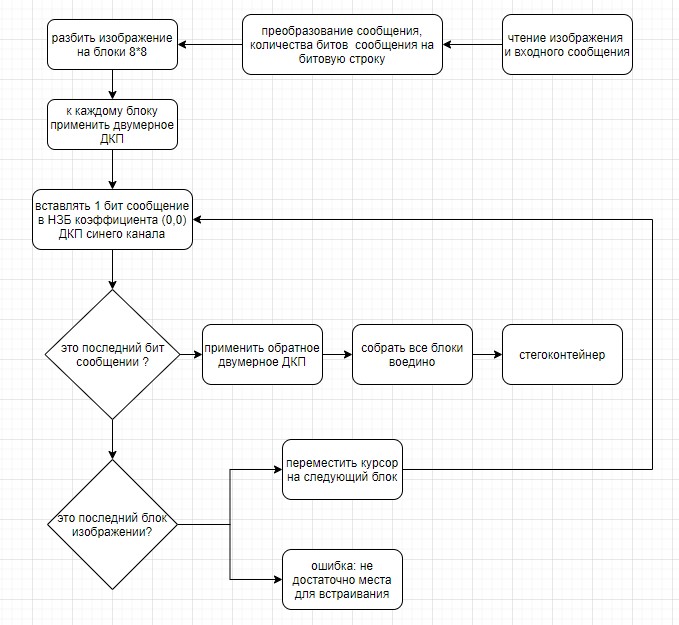


*Рус. 1: Пример использования LSB*

**Алгоритм встраивания секретного сообщения**

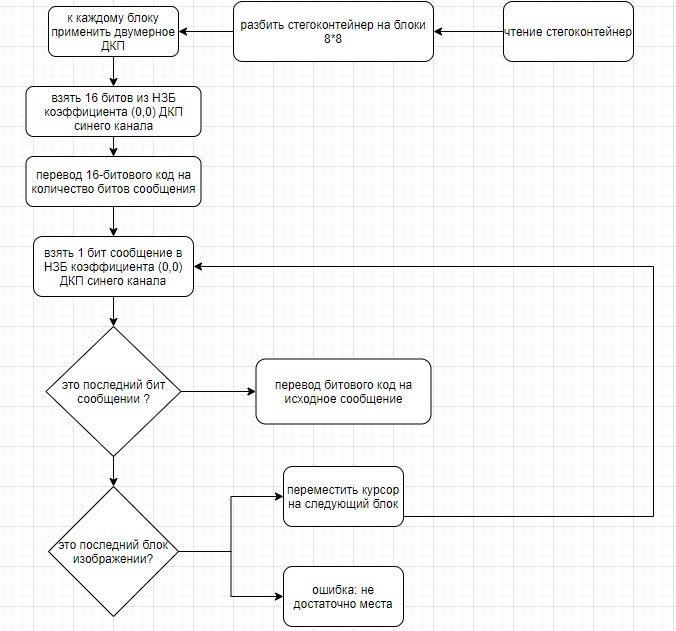
Встраивание выполнено по следующим приципам:

* использовать синие каналы писели исходного изображения
* встраивать сверху вниз и слево направо.



*Рус. 2: Блок-схема алгоритма встраивания секретного сообщения*

**Алгоритм извлечения секретного сообщения**



*Рус. 3: Блок-схема алгоритма извлечения секретного сообщения*

Исходным контейнером является изображение “container.bmp” с размером 320x256 пикселей в cover.bmp” с размером 320x256 пикселей в формате BMP (изображение в ссылке:

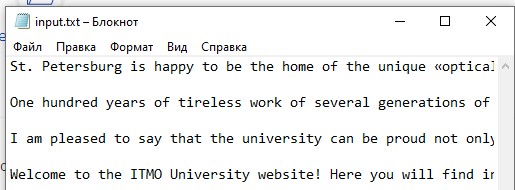
<https://drive.google.com/drive/u/0/folders/0ByO8U171Nlp7fjB5Z19oQnl3VUI5Y3hnVjFfLVNlSlpPazRpTGcydXJkWHpzQjdQc0llMEU>).



*Рyс 4. Изображенный контейнер*

Текстовое исходное сообщение (информация из ссылки :

[https://en.itmo.ru/en/page/41/Rectors\_Greeting.htm)](https://en.itmo.ru/en/page/41/Rectors_Greeting.htm).



Для написания программы использовался язык Python v.3.8.2, так как мне нравится работа со строками в данном языке. Питон медленнее C/C++, но в данном случае мы не заинтересованы в космической скорости работы алгоритма, с отые доли секунд никак не влияют на эффективность программ.

Программа делится на 3 модули:

1. **lab3.py** содержатся основые функции: чтение информации из командной строки, чтение и запись с файлами .
2. **LSB-DCT.py** включает две функции: встраивание инфромации в изображение и извлечения из него .
3. Модуль **analysis.py** написан для построения график зависимости PSNR от размера встраивания.

**Запуска встраивания-извлечения**

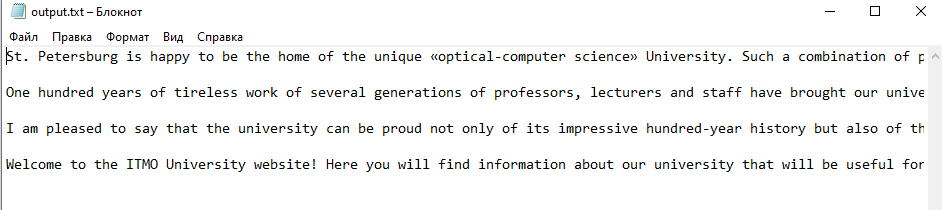
Вход: файл monalisa.bmp, текстовой файл input.txt.

Выход: outmonalisa.bmp



## *Рисунок 5.* *Запуска программы*

Текстовое выходное сообщение - файл output.txt:



*Рис. 6: Файл output.txt*

Выходной стегоконтейнер – файл outmonalisa.bmp:



*Рис. 7: Выходной стегоконтейнер*

Пиковое отношение сигнала к шуму:

*Рис. 8: Вычисление PSNR*

Запускать программу для встраивания в другой коэффициент

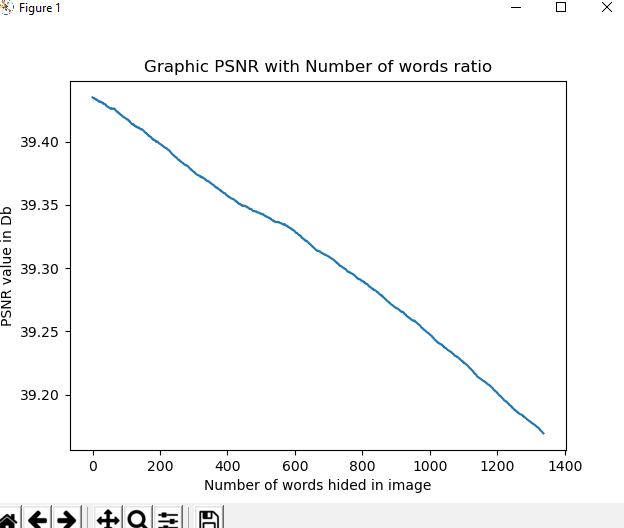
Если выполнено встраивание на коэффициент (2,2) (у нас в работе встраивание на коэффициент 0,0), то получается результат на рисунке 9 и 10

**Построение графика PSNR:**

В этом работе, мы построим графику значений PSNR исходного изображения и изображения с информацией, встроенные по наростающей одно слово, пять, десять, .. до конца текстового файла.

Вход: файл container.bmp, текстовой файл in\_text.txt.

Выход: графика PSNR исходного изображения и изображения

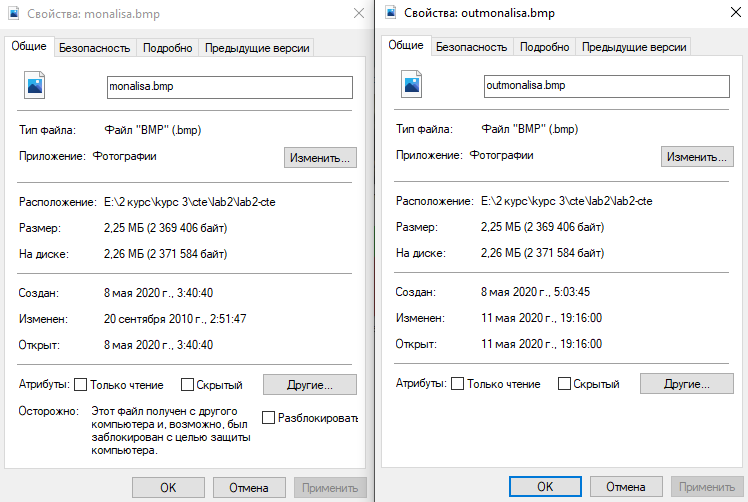


*Рисунок 9. графика PSNR исходного изображения и изображения*

Можно видеть, что это значение идет от бесконечного и продолжает падать, когда мы вставляем больше символа.

**Оценка целесообразности**

После встраивания, размер файла не меняется



*Рисунок 9. Сравнение размера файлов*

Вывод : При выполнении данной лабораторной работы мною был изучен метод LSB-DCT стеганографии. Я научился применять его и проводить последующую оценку их применению. Можно сделать вывод, что кочество изображения заданого метода только среднее и такой метод целесообразен для небольшего секретного сообщения .

**Список использованной литературы:**

1. Стеганография. Метод LSB :2018 г

URL : [https://ghostbasenji.blogspot.com/2018/08/steganography-methodLSB.html.](https://ghostbasenji.blogspot.com/2018/08/steganography-method-LSB.html)

1. Метод сокрытия информации в квантованных коэффициентах дискретного косинус преобразования : 2016 г

URL: <http://ainjournal.ru/doc/847079.html>.

1. BMP

URL : <https://ru.wikipedia.org/wiki/BMP>.

1. Стеганография и другие альтернативные методы сокрытия информации -

2013г

URL : <https://www.bibliofond.ru/view.aspx?id=657009>.

1. Стеганография [Электронный ресурс]: 2011 г. URL :<https://habr.com/ru/post/114597/>.

# Приложение

# LSB\_DCT.py

from cv2 import cv2

import sys,numpy as np,itertools

class StegaException(Exception):

    pass

class LSB\_DCT\_steg():

    def \_\_init\_\_(self,img):

        self.image = img

        self.orirow,self.oricol,self.chan = img.shape

        # Добавляем некоторый пиксель, чтобы делить строку и столбец на 8

        if self.orirow % 8 != 0 or self.oricol %8 != 0:

            newRow = self.orirow+8-self.orirow%8

            newCol = self.oricol+8-self.oricol%8

            self.image = cv2.resize(self.image,(newCol,newRow))

        self.row,self.col,\_ = self.image.shape

        # разбить изображение на 8 \* 8 блоков

        self.blocks = self.break8()

        self.quant = np.array([[16,11,10,16,24,40,51,61],

                    [12,12,14,19,26,58,60,55],

                    [14,13,16,24,40,57,69,56],

                    [14,17,22,29,51,87,80,62],

                    [18,22,37,56,68,109,103,77],

                    [24,35,55,64,81,104,113,92],

                    [49,64,78,87,103,121,120,101],

                    [72,92,95,98,112,100,103,99]])

    # разбить изображение на 8 \* 8 блоков

    def break8(self):

        bImage,\_,\_ = cv2.split(self.image)

        bImage = np.float32(bImage)

        imgBlocks = [bImage[j:j+8, i:i+8]

                    for (j,i) in itertools.product(range(0,self.row,8),range(0,self.col,8))]

        return imgBlocks

    # вернуть все группы, в которых сумма всей ширины = self.col

    def chunkRows(self,blocks,n):

        m = int(n)

        for i in range(0, len(blocks), m):

            yield blocks[i:i + m]

    def reshape(self,blocks):

        \_,gImage,rImage = cv2.split(self.image)

        bImage = []

        for chunk in self.chunkRows(blocks,self.col/8):

            for numRow in range(8):

                for block in chunk:

                    bImage.extend(block[numRow])

        bImage = np.array(bImage).reshape(self.row,self.col)

        bImage = np.uint8(bImage)

        img = cv2.merge((bImage,gImage,rImage))

        return img

    # изменить значение в двоичную форму с фиксированным размером кусочка

    def binary\_value(self,value,bitsize):

        binVal = bin(value)[2:]

        if len(binVal) > bitsize:

            raise StegaException("Your message contain unexpected character!!!")

        while len(binVal) < bitsize:

            binVal = '0' + binVal

        return binVal

    def binary\_text(self,mess):

        res = ""

        l = len(mess)

        res += self.binary\_value(l,16)

        for ch in mess:

            c = ord(ch)

            res += self.binary\_value(c,8)

        return res

    def embed(self,mess):

        # вычислить блоки коэффициентов DCT из изображения

        DCT\_blocks = [np.round(cv2.dct(block)) for block in self.blocks]

        # разделить каждый коэффициент на величину квантования

        quantizedDCT = [np.round(DCT\_block/self.quant) for DCT\_block in DCT\_blocks]

        # quantizedDCT = [np.round(DCT\_block/1) for DCT\_block in DCT\_blocks]

        # перевести сообщение в двоичную форму

        mess = self.binary\_text(mess)

        messIndex = 0

        # вставлять каждый бит сообщения в 1-й коэффициент каждого блока

        for block in quantizedDCT:

            DC = int(block[0,0])

            if DC % 2 == 0:

                DC ^= int(mess[messIndex])

            else:

                DC ^= int(mess[messIndex]) ^ 1

            block[0,0] = np.float32(DC)

            messIndex += 1

            if messIndex == len(mess): break

        if messIndex < len(mess)-1: raise StegaException("not enough spaces to embed")

        # умножить каждый коэффициент на значение квантования

        DCT\_blocks = [block \* self.quant for block in quantizedDCT]

        # DCT\_blocks = [block \* 1 for block in quantizedDCT]

        # рассчитать обратное DCT, чтобы получить значения изображения

        Img\_blocks = [np.round(cv2.idct(block)) for block in DCT\_blocks]

        # изменить форму блоков в форму изображения

        img = self.reshape(Img\_blocks)

        # img = cv2.resize(img,(self.oricol,self.orirow))

        return img

    # прочитать все вставленные биты в блоках

    def read\_bits(self,blocks):

        res = ""

        for block in blocks:

            c = int(block[0,0])

            if c % 2 == 0: res += '0'

            else: res += '1'

        return res

    def extract(self):

        # вычислить блоки коэффициентов DCT из изображения

        DCT\_blocks = [np.round(cv2.dct(block)) for block in self.blocks]

        # разделить каждый коэффициент на величину квантования

        quantizedDCT = [np.round(DCT\_block/self.quant) for DCT\_block in DCT\_blocks]

        # quantizedDCT = [np.round(DCT\_block/1) for DCT\_block in DCT\_blocks]

        length = self.read\_bits(quantizedDCT[:16])

        length = int(length,2)

        cursor = 16

        mess = ""

        for \_ in range(length):

            c = self.read\_bits(quantizedDCT[cursor:cursor+8])

            c = int(c,2)

            ch = chr(c)

            mess += ch

            cursor += 8

            if cursor >= len(quantizedDCT):

                raise StegaException("Your image hasn't been embed!!")

        return mess

    def statistic(self):

        # вычислить блоки коэффициентов DCT из изображения

        DCT\_blocks = [np.round(cv2.dct(block)) for block in self.blocks]

        # разделить каждый коэффициент на величину квантования

        quantizedDCT = [np.round(DCT\_block/self.quant) for DCT\_block in DCT\_blocks]

        import matplotlib.pyplot as plt

        histogram = []

        for block in quantizedDCT:

            for i in range(8):

                for j in range(8):

                    val = float(block[i,j])

                    histogram.append(val)

        plt.hist(histogram,bins = 100)

        plt.show()

# analysis.py

from cv2 import cv2

import numpy as np

class PSNRException(Exception):

    pass

class Analysis():

    def \_\_init\_\_(self,originImg,Img):

        self.origin = originImg

        self.changed = Img

        orirow,oricol = originImg.shape[:2]

        # Добавляем некоторый пиксель, чтобы делить строку и столбец на 8

        if orirow % 8 != 0 or oricol %8 != 0:

            newRow = orirow+8-orirow%8

            newCol = oricol+8-oricol%8

            self.origin = cv2.resize(originImg,(newCol,newRow))

        self.oheight, self.owidth, self.ochannel = self.origin.shape

        self.osize = self.oheight \* self.owidth \* self.ochannel

        self.cheight, self.cwidth, self.cchannel = Img.shape

        self.csize = self.cheight \* self.cwidth \* self.ochannel

        self.MAXi = 255 # maximum posible channel value of the image = 2^8-1

        self.maskONE = 0b00000001 # use bitwise OR to make LSB 1

    def MSE(self): # calculate MSE = 1/size \* sum((I(i,j)-K(i,j))^2) with i = 1..n;j = 1..m

        mse = np.mean((self.origin - self.changed)\*\*2)

        return mse

    def PSNR(self): # calculate PSNR = 10\*log10(MAXi^2/MSE)

        mse = np.mean((self.origin - self.changed)\*\*2)

        res = self.MAXi\*\*2/mse

        res = 10 \* np.log(res) / np.log(10)

        return res

    def Detec(self):

        for curheight in range(self.oheight):

            for curwidth in range(self.owidth):

                for curchan in range(self.ochannel):

                    I = int(self.origin[curheight,curwidth][curchan])

                    K = int(self.changed[curheight,curwidth][curchan])

                    self.origin[curheight,curwidth][curchan] = (I & self.maskONE) \* 255

                    self.changed[curheight,curwidth][curchan] = (K & self.maskONE) \* 255

        cv2.imwrite('tmp1.bmp',self.origin)

        cv2.imwrite('tmp2.bmp',self.changed)

class AnalysisException(Exception):

    pass

#lab3.py

from LSB\_DCT import \*

from analysis import \*

import argparse

#config аргументы командной строки

parse = argparse.ArgumentParser(description= "lab3 staganography- LTD N3349")

parse.add\_argument('-i',metavar='ifile',type=str,dest= "in\_text",help= "name of input text file")

parse.add\_argument('-ii',metavar='inimg',type=str,dest= "in\_img",help= "name of input image file",required= True)

parse.add\_argument('-o',metavar='ofile',type=str,dest= "out\_file",help= "name of output image/text file")

parse.add\_argument('option',metavar='opts',type=str,help= "name of option (embed/extract/PSNR/detection)")

args = parse.parse\_args()

# вставлять текстовые данные в изображение методом LSB-DCT

if args.option == "embed":

    # читать текст для вставки

    text = open(args.in\_text,"r",encoding="UTF-8").read()

    # получить данные и запустить LSB-DCT stegano

    data\_in = cv2.imread(args.in\_img)

    steg = LSB\_DCT\_steg(data\_in)

    data\_out = steg.embed(text)

    # записать данные изображения в файл

    cv2.imwrite(args.out\_file,data\_out)

# извлекать текстовые данные из изображения методом LSB-DCT

elif args.option == "extract":

    # получить данные и запустить инвертированный LSB-DCT стегано

    data\_in = cv2.imread(args.in\_img)

    steg = LSB\_DCT\_steg(data\_in)

    text = steg.extract()

    # записать извлеченный текст в файл

    open(args.out\_file,"w",encoding="UTF-8").write(text)

# рассчитать PSNR 2 изображения

elif args.option == "PSNR":

    # получить 2 входных изображения и запустить PSNR

    data1 = cv2.imread(args.in\_img)

    data2 = cv2.imread(args.out\_file)

    analize = Analysis(data1,data2)

    PSNR = analize.PSNR()

    # показать PSNR

    print("PSNR of your 2 images is:",end = " ")

    print(PSNR)

# нарисовать PSNR-изображение с изображением, вставленным символами 1,2, ....

elif args.option == "graph":

    # читать вводимый текст

    text = open(args.in\_text,"r",encoding="UTF-8").read()

    # получить данные входных изображений

    data\_in = cv2.imread(args.in\_img)

    # data\_out = cv2.imread(args.out\_file)

    in\_text = ""

    PSNRs = []

    numWords = []

    for i in range(len(text)):

        in\_text += text[i]

        print(in\_text)

        steg = LSB\_DCT\_steg(data\_in) # метод init LSB-DCT

        data\_out = steg.embed(in\_text)

        data\_in = cv2.imread(args.in\_img)

        analize = Analysis(data\_in,data\_out)

        # сохранить выходные данные в массив

        numWords.append(i)

        PSNR = analize.PSNR()

        PSNRs.append(PSNR)

    # рисовать PSNR графику

    import matplotlib.pyplot as plt

    plt.plot(numWords,PSNRs)

    plt.xlabel("Number of words hided in image")

    plt.ylabel("PSNR value in Db")

    plt.title("Graphic PSNR with Number of words ratio")

    plt.show()

# запустить простую атаку на метод LSB-DCT

elif args.option == "detection":

    # получить данные входных изображений

    data\_in = cv2.imread(args.in\_img)

    steg = LSB\_DCT\_steg(data\_in)

    steg.statistic()