Цель работы:

Приобретение навыков встраивания и извлечения информации в цифровых изображениях с помощью метода наименее значимых битов (LSB).

Задача работы:

Программно реализовать метод встраивания и извлечения сообщений в изображение формата «.bmp», в частности, в наименее значимые биты. Построить блок-схему алгоритма. Рассчитать параметр PSNR для исходного изображения и изображения со встроенной информации. Построить график зависимости PSNR от количества встроенной информации. Провести оценку целесообразности встраивания.

Теоретическая часть

Стеганография в цифровых изображениях – раздел стеганографии, изучающий проблему сокрытия данных в цифровых изображениях. В отличие от криптографии, задача стеганографии – скрыть сам факт наличия скрытого сообщения [1].

Пространственные методы манипулируют значениями в пространственной области (пикселями). К одному из основных методов относится метод LSB (англ. Least Significant Bit – Наименее значимый бит) [1].

Данный метод заключается в выделении наименее значимых бит изображения-контейнера с последующей их заменой на биты сообщения. Поскольку замене подвергаются лишь наименее значимые биты, разница между исходным изображением-контейнером и контейнером, содержащим скрытые данные невелика и обычно незаметна для человеческого глаза. Метод LSB применим лишь к изображениям в форматах без сжатия (например, BMP) или со сжатием без потерь (например, GIF), так как для хранения скрытого сообщения используются наименее значимые биты значений пикселей, при сжатии с потерями эта информация может быть утеряна. Форматы без сжатия имеют очень большой размер и могут вызвать подозрение, по этому для стеганографии чаще используют другие форматы [1].

Практическая часть

Согласно задачам данной лабораторной работы, необходимо программно реализовать механизм встраивания и извлечения сообщения в изображение. Для выполнения этих задач был выбрал язык программирования Python 3.8.

Имеется цветное изображение, состоящее из пикселей, представленными сочетанием трех цветовых компонентов (субпикселей) RGB (красный, зеленый, синий). Каждый субпиксель имеет интенсивность, выраженную восьмибитным числом, в десятичном виде от 0 до 255 включительно, где 255 – максимальная интенсивность. Задача программного модуля – разобрать имеющееся изображение на субпиксели, провести побитовое встраивание сообщения, заведомо переведенного в двоичную строку, в наименее значимые 8-ые биты, затем собрать изображение обратно в формат «.bmp».

Преобразование сообщения, состоящего из кириллических букв и пробелов в двоичный код, осуществляется со стандартизированной размерностью символа в 7 бит.

Демонстрация работы программы:

Рисунок 1. Исходное изображение.

Рисунок 2. Изображение со встроенным сообщением.

```
Для встраивания введите 1, для извлечения введите 2: 1
Введите сообщение для встраивания в текст: Сообщение для встраивания
Встроено сообщение "Сообщение для встраивания" в изображение "img_encoded.bmp"
Параметр PSNR равен 45.735 dB
```

Рисунок 3. Демонстрация вывода программы при встраивании сообщения.

```
Для встраивания введите 1, для извлечения введите 2: 2
Извлечено сообщение "Сообщение для встраивания" из изображения "img_encoded.bmp"
```

Рисунок 4. Демонстрация вывода программы при извлечении сообщения.

Для большего понимания работы программы представляется блок-схема программы:

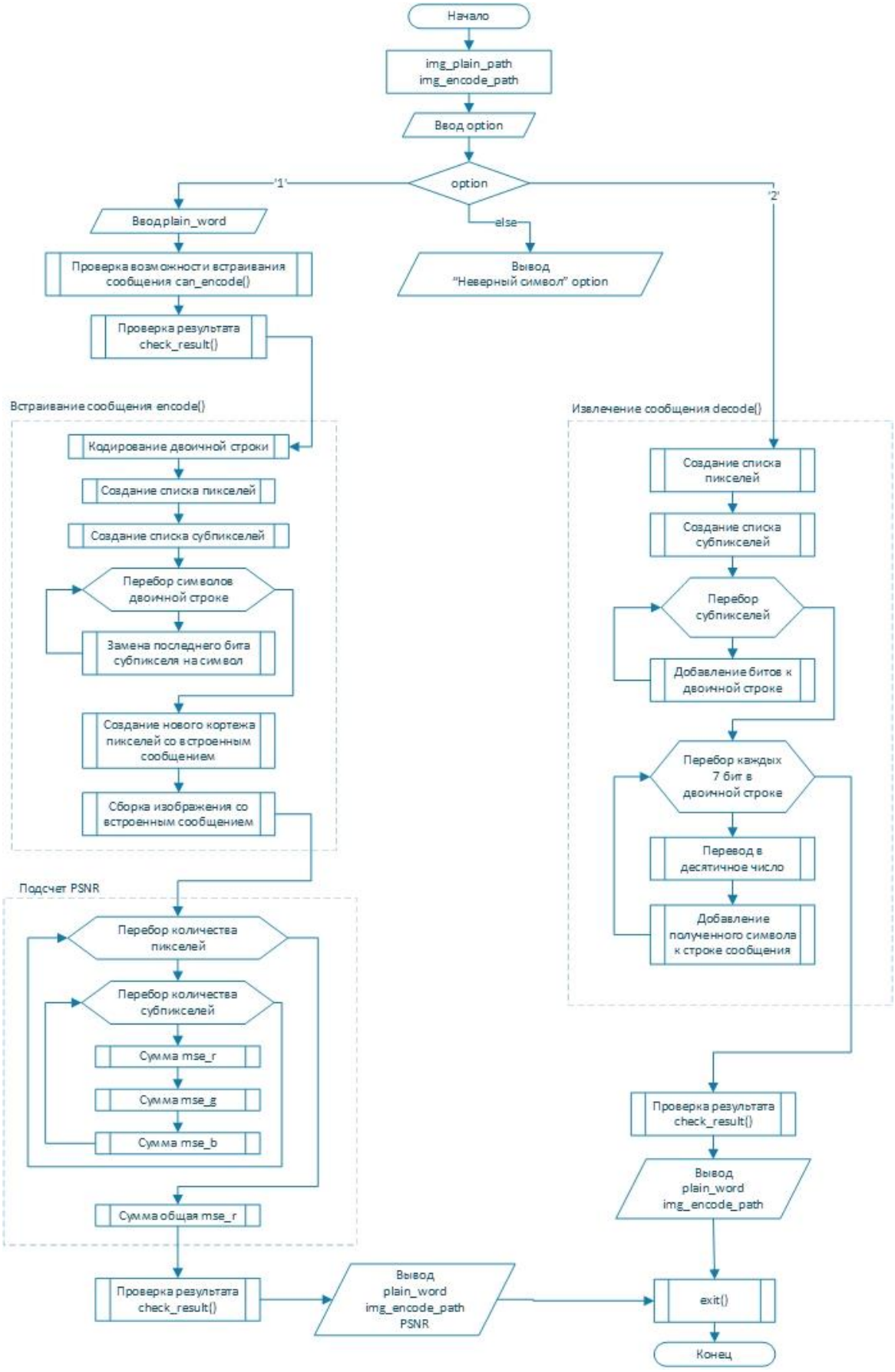

Рисунок 5. Блок-схема программы.

Был программно реализован расчет параметра PSNR для исходного изображения и изображения со встроенной информацией.
Встроим сообщение «стеганография» в изображение. Параметр PSNR равен: 48.798 dB

Встроим по нарастающей 1, 5, 10, 20, 30, 40, 50 слов «стеганография» в изображение. Построим график зависимости параметра PSNR от количества встраиваемых слов.

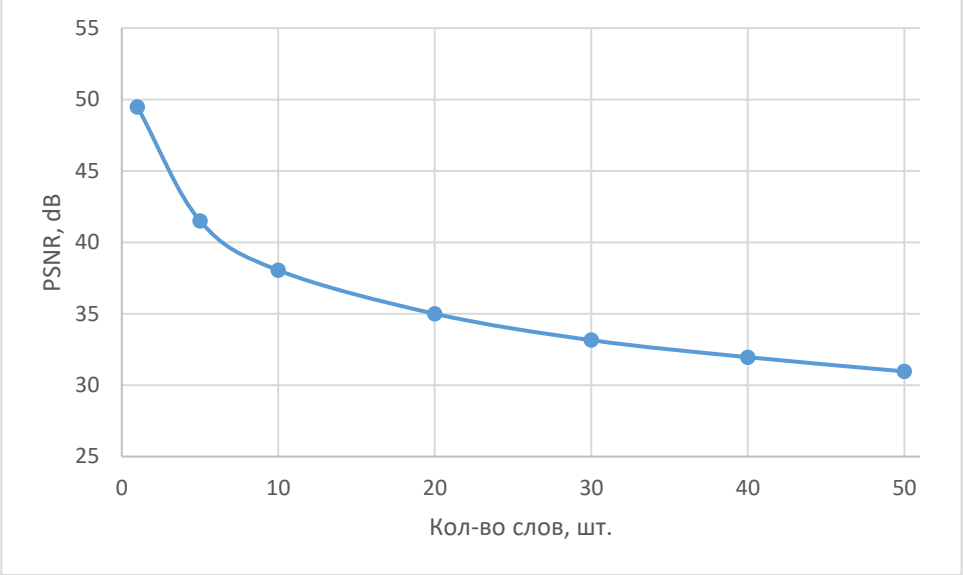

Рисунок 6. График зависимости параметра PSNR от количества встраиваемых слов.

График зависимости параметра PSNR от количества встраиваемых слов получился постоянно убывающим, напоминающим убывающий экспоненциальный график.

Это говорит о том, что чем больше длина встраиваемого сообщения в текст, тем больше шума наблюдается на изображении, так как параметр PSNR есть отношение максимально возможного сигнала к уровню шума. Также можно заметить, что с увеличением длины встраиваемого сообщения прирост количества шума снижается.

Так как встраивание производилось только в последний бит значения интенсивности субпикселя, то визуально данные шумы незаметны. При таком подходе в изображение с размером 64x64 пикселя возможно встроить сообщение длиной до 1750 символов. Поэтому данный метод встраивания является довольно целесообразным, в том числе и для передачи сообщения по открытым каналам связи.

Вывод:

В результате работы над программной реализацией метода встраивания и извлечения сообщения в изображение были приобретены навыки сокрытия информации в цифровых изображениях с помощью метода наименее значимых битов (LSB).

Данный метод является довольно целесообразным, в том числе и для использования в открытых каналах связи для незаметной передачи сообщения.

Список использованной литературы:

1. «СРАВНЕНИЕ МЕТОДОВ СТЕГАНОГРАФИИ В ИЗОБРАЖЕНИЯХ» // Сейеди С. А., Садыхов Р. Х // Информатика. 2013;(1):66-75.
2. «Цифровая стеганография» // Грибунин В.Г. // ООО "СОЛОН-Пресс" 2002 г.

Приложение 1. Листинг программного кода.

```
import math
from PIL import Image

img_plain_path = 'img_plain.bmp'
img_encode_path = 'img_encoded.bmp'
img_in = img_out = 0

# Встраивание сообщения в изображение
def encode(plain_word, img_plain_path, img_encode_path):
        # Кодирование двоичной строки
        cypher_word = ''
        for i in plain_word:
                if i == ' ':
                        cypher_word += '1111111'
                else:
                        cypher_word += format(ord(i)-1024, '07b')
        # Создание списка пикселей
        img_in = Image.open(img_plain_path)
        img_in_res = img_in.size
        pixels = list(img_in.getdata())
        img_in.close()
        # Создание списка субпикселей
        pixels_bin = []
        for p in pixels:
                [pixels_bin.append(bin(sp)[2:].rjust(8,'0')) for sp in p]
        # Встраивание двочной строки
        for sp in range(len(pixels_bin)):
                if sp < len(cypher_word):
                        pixels_bin[sp] = str(pixels_bin[sp][:7] + cypher_word[sp])
                elif sp == len(cypher_word):
                        for end in range(sp,sp+7):
                                pixels_bin[end] = str(pixels_bin[end][:7] + '0')
                        break
        # Создание нового кортежа пикселей со встроенным сообщением
        i = 0
        pixels_new = []
        for p in range(int(len(pixels_bin)/3)):
                pix = []
                for sp in range(3):
                        pix.append(int(pixels_bin[i],2))
                        i+=1
                pix = tuple(pix)
                pixels_new.append(pix)
        pixels_new = tuple(pixels_new)
        # Создание изображения со встроенным сообщением
        img_out = Image.new('RGB', img_in_res)
        img_out.putdata(pixels_new)
        img_out.save(img_encode_path)
        img_out.close()
        # Рассчет параметра PSNR
        img_in = Image.open(img_plain_path)
        img_a = list(img_in.getdata())
        img_out = Image.open(img_encode_path)
        img_b = list(img_out.getdata())
        mse_r  = mse_g = mse_b = mse_t = psnr = 0
        for y in range(len(img_a)):
                for x in range(3):
                        if x == 0:
                                mse_r += (int(img_b[y][x])-int(img_a[y][x]))**2
                        if x == 1:
                                mse_g += (int(img_b[y][x])-int(img_a[y][x]))**2
                        if x == 2:
                                mse_b += (int(img_b[y][x])-int(img_a[y][x]))**2
        mse_t = (mse_r + mse_g + mse_b)/(3*len(img_a))
        psnr = round(10*math.log10((255^2)/mse_t),3)
        img_in.close()
        img_out.close()
        return psnr

# Извлечение сообщения из изображения
def decode(img_encode_path):
        # Создание списка пикселей
        img_in = Image.open(img_encode_path)
        img_in_res = img_in.size
        pixels = list(img_in.getdata())
        img_in.close()
        # Создание списка субпикселей
        pixels_bin = []
        for p in pixels:
                [pixels_bin.append(bin(sp)[2:].rjust(8,'0')) for sp in p]
```

```
        # Извлечение двоичной строки
        cypher_word = ''
        bit_count = zero_count = 0
        for sp in pixels_bin:
                if sp[7] == '0':
                        cypher_word += '0'
                        zero_count += 1
                        bit_count += 1
                elif sp[7] == '1':
                        cypher_word += '1'
                        bit_count += 1
                if bit_count == 7 and zero_count != 7:
                        bit_count = 0
                        zero_count = 0
                elif bit_count == 7 and zero_count == 7:
                        cypher_word = cypher_word[:-7]
                        break
        # Декодирование двоичной строки
        string = ''
        plain_word = ''
        for i in range(len(cypher_word)):
                string += cypher_word[i]
                if len(string) == 7:
                        if string == '1111111':
                                plain_word += ' '
                        else:
                                plain_word += chr(int(string, 2)+1024)
                        string = ''
        return plain_word

# Проверка возможности встраивания сообщения
def can_encode(plain_word, img_plain_path):
        img_in = Image.open(img_plain_path)
        img_in_res = img_in.size
        if ((len(plain_word)*7)+7) > (img_in_res[0]*img_in_res[1]*3):
                return -1
        img_in.close()
        return 0

# Проверка результата выполнения функции
def check_result(res, msg):
        if res == -1:
                print(msg)
                exit1()

# Завершение работы при ошибке
def exit1():
        if img_in != 0:
                img_in.close()
        if img_out != 0:
                img_out.close()
        print ('exit(-1)')
        exit(-1)

# Выбор встраивания/извлечения
option = input('Для встраивания введите 1, для извлечения введите 2: ')
if option not in '12':
        print ('Неверный символ: ', option)
        exit1()

# Встраивание
if option == '1':

        plain_word = input('Введите сообщение для встраивания в текст: ')
        res = can_encode(plain_word, img_plain_path)
        check_result(res, 'Нехватка пикселей для встраивания')

        res = encode(plain_word, img_plain_path, img_encode_path)
        check_result(res, 'Не удалось выполнить встраивание')

        print('Встроено сообщение \"', plain_word, '\" в изображение \"', img_encode_path,
                '\"', sep='')
        print('Параметр PSNR равен', res, 'dB')

# Извлечение
elif option == '2':

        res = decode(img_encode_path)
        check_result(res, 'Не удалось выполнить извлечение')
        print('Извлечено сообщение \"', res, '\" из изображения \"', img_encode_path, '\"',
                sep='')

exit(0)
```