Серт Серкан, группа 8

Лабораторная работа №1

Метод LSB (Least Significant Bit)

Вариант№3

Использование метода LSB базируется на невосприимчивости человеческих органов чувств к малозначительным изменениям в контейнерах, обладающих психовизуальной избыточностью. При встраивании данных в пространственной области графических контейнеров метод LSB реализует замену наименее значимых бит (НЗБ) значений яркости или цветности отдельных пикселей контейнера битами скрываемого сообщения

Цель работы:

Реализовать LSB-алгоритм. В качестве метрик для оценки искажений заполненных контейнеров использовать **μmax D , μSNR , μPSNR** . Построить зависимости вероятности ошибок при извлечении скрытых данных от объема скрываемой информации.

Код программы:

```
from PIL import Image
import numpy as np
import warnings
warnings.filterwarnings(action='once')
encoding: str = 'utf-8'

class LSB:
    def __init__(self, old_image_path: str, new_image_path: str):
        self.__empty_image_path: str = old_image_path
        self.__full_image_path: str = new_image_path
        self.__occupancy: int = 0

    @staticmethod
    def str_to_bits(message: str) -> list:
        result = []
        for num in list(message.encode(encoding=encoding)):
            result.extend([(num >> x) & 1 for x in range(7, -1, -1)])

        return result

    @staticmethod
    def bits_to_str(bits: list) -> str:
        chars = []
        for b in range(len(bits) // 8):
            byte = bits[b * 8:(b + 1) * 8]
            chars.append(chr(int(''.join([str(bit) for bit in byte]), 2)))
```

```
        return ''.join(chars)

    def embed(self, message: str,new_image_path: str):
        img = Image.open(self.__empty_image_path).convert('RGB')
        picture = np.asarray(img, dtype='uint8')
        picture = picture.astype('uint8')
        img.close()
        picture_shape = picture.shape
        height, width, depth = picture.shape[0], picture.shape[1],
picture.shape[2]
        message_bits = LSB.str_to_bits(message)
        if len(message_bits) > height * width * depth:
            raise ValueError('Message greater than capacity!')
        message_bits = np.asarray(message_bits)
        bits_length = message_bits.shape[0]
        picture = picture.reshape(-1)
        picture[:bits_length] = ((picture[:bits_length] >> 1) << 1) |
message_bits
        picture = picture.reshape(picture_shape)
        print('bits',message_bits)
        print('bits_leng',bits_length)
        self.__occupancy = bits_length
        Image.fromarray(picture).save(new_image_path, 'PNG')  # Сохраняем
новое изображение
        self.__occupancy = bits_length
        Image.fromarray(picture).save(self.__full_image_path, 'PNG')

    def recover(self) -> str:
        img = Image.open(self.__full_image_path).convert('RGB')
        picture = np.asarray(img, dtype='uint8')
        picture = picture.astype('uint8')
        img.close()
        recovered_message = picture.reshape(-1)[:self.__occupancy] & 0x01
        return LSB.bits_to_str(list(recovered_message))

    @property
    def occupancy(self) -> int:
        return self.__occupancy

def metrics(empty_image: str, full_image: str) -> None:
    img = Image.open(empty_image).convert('RGB')
    empty = np.asarray(img, dtype='uint8')
    img.close()
    img = Image.open(full_image).convert('RGB')
    full = np.asarray(img, dtype='uint8')
    img.close()

    MAX_D = np.max(np.abs(empty.astype(int) - full.astype(int)))
```

```
    SNR = np.sum(empty * empty) / np.sum((empty - full) ** 2)
    H, W = empty.shape[0], empty.shape[1]
    MSE = np.sum((empty - full) ** 2) / (W * H)
    max_pixel = 255.0
    PSNR = 20 * np.log10(max_pixel / np.sqrt(MSE))
    sigma = np.sum((empty - np.mean(empty)) * (full - np.mean(full))) / (H *
W)
    UQI = (4 * sigma * np.mean(empty) * np.mean(full)) / \
          ((np.var(empty) ** 2 + np.var(full) ** 2) * (np.mean(empty) ** 2 +
np.mean(full) ** 2))
    print(f'Максимальное абсолютное отклонение (MAX): {MAX_D}')
    print(f'Отношение сигнал-шум (SNR): {SNR}')
    print(f'Среднее квадратичное отклонение (MSE): {MSE}')
    print(f'Пиковое отношение сигнал-шум (PSNR): {PSNR}')
    print(f'Универсальный индекс качества (УИК): {UQI}')

def extract_with_capacity(full_image_path, capacity):
    lsb = LSB(full_image_path, full_image_path)  # Используем тот же путь для
извлечения
    message = lsb.recover()  # Извлекаем скрытое сообщение
    extracted_message = message[:capacity]  # Получаем часть сообщения с
заданным объемом
    return extracted_message

def main():
    old_image = 'input/SERT_.png'
    new_image = 'output/sert_.png'

    with open('message.txt', mode='r', encoding=encoding) as file:
        message = file.read()
    capacities = [len(message)]  # Разные объемы сообщений для тестирования
    for capacity in capacities:
        new_image_path = 'output/sert_capacity_{}.png'.format(capacity)
        lsb = LSB(old_image, new_image_path)
        lsb.embed(message[:capacity], new_image_path)  # Вставляем сообщение и
сохраняем новое изображение
        recovered_message = lsb.recover()
        error_probability = sum(1 for a, b in zip(message[:capacity],
recovered_message) if a != b) / capacity
        print(f'Объем информации: {capacity}, Вероятность ошибки:
{error_probability}')
        print('сообщение:\t{}'.format(recovered_message))
        metrics(old_image, new_image)

if __name__ == '__main__':
    main()
```

Результат работы программы:

Исходное изображение (пустой контейнер)

Изображение со встроенным сообщением (заполненный контейнер)

Результаты работы программы (введено сообщение «My secret messages»):

```
bits [0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0
 1 1 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 0 0 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 0 0
 1 1 0 1 0 1 0 0 1 1 0 0 0 0]
bits_leng 88
Объем информации: 11, Вероятность ошибки: 0.0
сообщение:      10846323850
Максимальное абсолютное отклонение (MAX): 1
Отношение сигнал-шум (SNR): 2817195.093023256
Среднее квадратичное отклонение (MSE): 0.0001194444444444444
Пиковое отношение сигнал-шум (PSNR): 87.35914406055612
Универсальный индекс качества (УИК): 0.0007829034970413483
PS C:\Users\serkan\Desktop\okul ödevler\stg\Task1_Sert>
```