Труфанов Илья Евгеньевич, группа 11-2

Лабораторная работа №1

Вариант №3

Метод LSB

Задание

Реализовать LSB-алгоритм. В качестве метрик для оценки искажений заполненных контейнеров использовать μ_{maxD}, μ_{SNR}, μ_{PSNR}. Построить зависимости вероятности ошибок при извлечении скрытых данных от объема скрываемой информации.

Код программы

modules.py

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

def text_to_bits(text):
    """Преобразует текст в битовую строку (8 бит на символ)."""
    return ''.join(format(ord(c), '08b') for c in text)

def lsb_embed(image, data):
    h, w = image.shape[:2]
    channels = image.shape[2] if image.ndim == 3 else 1
    capacity = h * w * channels  # Используем только 1 бит на пиксель

    if len(data) > capacity:
        raise ValueError("Объём данных превышает ёмкость изображения.")

    flat = image.flatten()
    new_flat = flat.copy()

    for i in range(len(data)):
        new_flat[i] = (new_flat[i] & 0b11111110) | int(data[i])  # Заменяем
только 1 младший бит

    stego = new_flat.reshape(image.shape)
    return stego

def lsb_extract(stego, data_length):

    flat = stego.flatten()
    extracted_bits = ''.join(str(pixel & 1) for pixel in
flat[:data_length])  # Извлекаем только 1 бит на пиксель
```

```
    return extracted_bits

def compute_snr(original, stego):
    """Вычисляет отношение сигнал/шум (SNR) в децибелах между оригиналом и
стегоизображением."""
    signal = np.sum(original.astype(np.float64) ** 2)
    noise = np.sum((original.astype(np.float64) - stego.astype(np.float64))
** 2)
    if noise == 0:
        return float('inf')
    snr = 10 * np.log10(signal / noise) #Приведение в децибелы
    #snr = signal / noise
    return snr

def compute_maxD(original, stego):
    """Вычисляет максимальное абсолютное отклонение."""
    return np.max(np.abs(original.astype(np.float64) -
stego.astype(np.float64)))

def compute_psnr(original, stego):
    """Вычисляет пиковое отношение сигнал-шум."""
    max_pixel_value = float(np.max(original))
    mse = np.mean((original.astype(np.float64) - stego.astype(np.float64))
** 2)
    if mse == 0:
        return float('inf')
    return (float(original.shape[0] * original.shape[1]) * (max_pixel_value
** 2)) / mse

def plot_error_vs_capacity(original, text):
    """Строит график зависимости вероятности ошибок от объема скрываемых
данных."""
    data_lengths = np.linspace(0, len(text_to_bits(text)), 10, dtype=int)
    errors = []

    for data_length in data_lengths:
        data = text_to_bits(text[:data_length // 8])
        stego = lsb_embed(original, data)
        extracted_bits = lsb_extract(stego, len(data))
        extracted_text = ''.join(chr(int(extracted_bits[i:i+8], 2)) for i
in range(0, len(extracted_bits), 8))
        error_rate = sum(a != b for a, b in zip(text[:len(extracted_text)],
extracted_text)) / len(text)
        errors.append(error_rate)

    plt.plot(data_lengths, errors, marker='o')
    plt.xlabel("Объем скрываемой информации (биты)")
```

```
    plt.ylabel("Вероятность ошибки")
    plt.title("Зависимость вероятности ошибок от объема скрываемых данных")
    plt.grid()
    plt.show()
```

lsb_main.py #для запуска кода

```
import cv2
from modules import text_to_bits, lsb_embed, compute_maxD, compute_psnr,
compute_snr, plot_error_vs_capacity, lsb_extract

if __name__ == '__main__':
    original = cv2.imread('1.png')
    if original is None:
        print("Ошибка: не удалось загрузить 1.png")
    else:
        text = "secret message"
        data = text_to_bits(text)

        stego = lsb_embed(original, data)
        cv2.imwrite('stego.png', stego)
        print("Данные встроены и стегоизображение сохранено как stego.png")

        maxD = compute_maxD(original, stego)
        snr = compute_snr(original, stego)
        psnr = compute_psnr(original, stego)

        print("μ_maxD:", maxD)
        print("μ_SNR:", snr, "dB")
        print("μ_PSNR:", psnr)

        extracted_bits = lsb_extract(stego, len(data))
        extracted_text = ''.join(chr(int(extracted_bits[i:i+8], 2)) for i
in range(0, len(extracted_bits), 8))
        print("Извлечённый текст:", extracted_text)

        plot_error_vs_capacity(original, text)
```

Результат выполнения программы

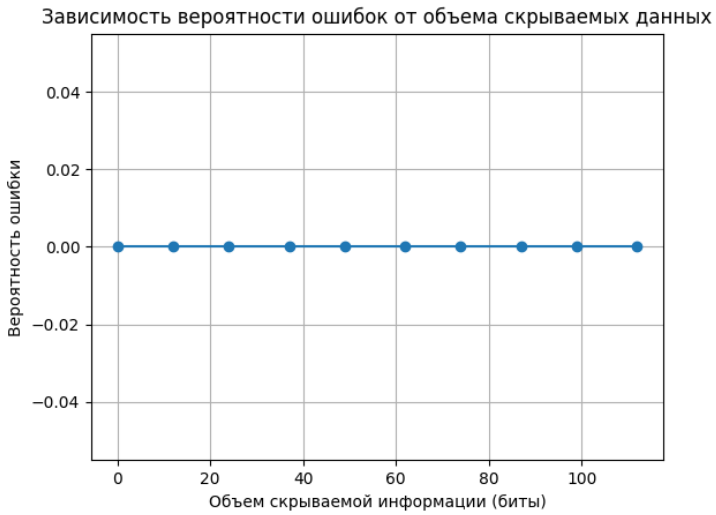
Для проверки работы алгоритма было взято следующее изображение:

```bash
#!/bin/bash

count=1
timeout=1
text=""

while [[ $# -gt 0 ]]; do
    case "$1" in
        -n) count="$2"; shift 2;;
        -t) timeout="$2"; shift 2;;
        --) shift; text="$@"; break;;
        *) echo "Wrong argument: $1"; exit 1;;
    esac
done

if [[ -z "$text" ]]; then
    echo "No text provided."
    exit 1
fi

for (( i=1; i<=count; i++ )); do
    echo "$text"
    sleep "$timeout"
done
```

После работы метода LSB было получено второе изображение-контейнер:

```bash
#!/bin/bash

count=1
timeout=1
text=""

while [[ $# -gt 0 ]]; do
    case "$1" in
        -n) count="$2"; shift 2;;
        -t) timeout="$2"; shift 2;;
        --) shift; text="$@"; break;;
        *) echo "Wrong argument: $1"; exit 1;;
    esac
done

if [[ -z "$text" ]]; then
    echo "No text provided."
    exit 1
fi

for (( i=1; i<=count; i++ )); do
    echo "$text"
    sleep "$timeout"
done
```

Также был получен график зависимости вероятности ошибок от объема скрываемых данных:



Вероятность ошибки будет расти если исходный текст не будет совпадать с извлеченным, к примеру если сохранить конечное изображение в JPEG с сжатием.

Метрики и извлеченный текст:

μ_maxD: 1.0

μ_SNR: 78.32839473855299 dB

μ_PSNR: 270443466037378.72

Извлечённый текст: secret message

Заключение

В ходе лабораторной работы был изучен LSB-алгоритм. Реализованы методы встраивания и извлечения текстового сообщения, а также проведена оценка искажений с помощью метрик: максимального абсолютного отклонения (maxD), отношения сигнал-шум (SNR) и пикового отношения сигнал-шум (PSNR). Результаты подтвердили эффективность метода для сокрытия информации при минимальных искажениях изображения.