МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №13 НА ТЕМУ:**

**ИССЛЕДОВАНИЕ СТЕГАНОГРАФИЧЕСКОГО МЕТОДА НА ОСНОВЕ ПРЕОБРАЗОВАНИЯ НАИМЕНЕЕ ЗНАЧАЩИХ БИТОВ**

                                                                 Выполнил студент 3 курса 5 группы

Дмитрук Илья Игоревич

Минск 2024

# Задание 1.

Метод НЗБ основывается на ограниченных способностях зрения или слуха человека, вследствие чего людям тяжело различать незначительные вариации цвета или звука. Рассмотрим это на примере 24-битного растрового RGB-изображения. Как известно, каждая точка кодируется тремя байтами. Каждый байт определяет интенсивность красного (Red), зеленого (Green) и синего (Blue) цветов. Совокупность интенсивностей цвета в каждом из трех каналов определяет оттенок пикселя. Младшие биты (выделены бледным, справа) дают незначительный «вклад» в изображение по сравнению со старшими. Замена одного или даже нескольких младших битов для человеческого глаза будет почти незаметна, поскольку реально человек может различать около полторы сотни цветовых оттенков. Таким образов в каждый пиксель изображения можно поместить 3 бита текста. Для начала был реализован метод последовательной подстановки LSB. Программная реализация данного алгоритма представлена в листинге 1.1.

public static Bitmap PosledovatelnyiSteganographyLSB(Bitmap originalImage, byte[] message)

{

int height = originalImage.Height, width = originalImage.Width;

int messageIndex = 0, bitIndex = 0;

Bitmap NewImg = new Bitmap(originalImage);

for (int y = 0; y < height; y++)

{

for (int x = 0; x < width; x++)

{

if (messageIndex < message.Length)

{

Color pixel = originalImage.GetPixel(x, y);

int red = pixel.R;

int green = pixel.G;

int blue = pixel.B;

byte currentByte = message[messageIndex];

int currentBit = (currentByte >> (7 - bitIndex)) & 0x01;

red = (red & 0xFE) | currentBit;

Color stegoPixel = Color.FromArgb(red, green, blue);

NewImg.SetPixel(x, y, stegoPixel);

bitIndex++;

if (bitIndex >= 8)

{

bitIndex = 0;

messageIndex++;

}

}

}

}

return NewImg;

}

Листинг 1.1 – Функция, выполняющая алгоритм последовательной подстановки LSB

Результат вставки и извлечения битов сообщения в изображения представлен на рисунке 1.1.

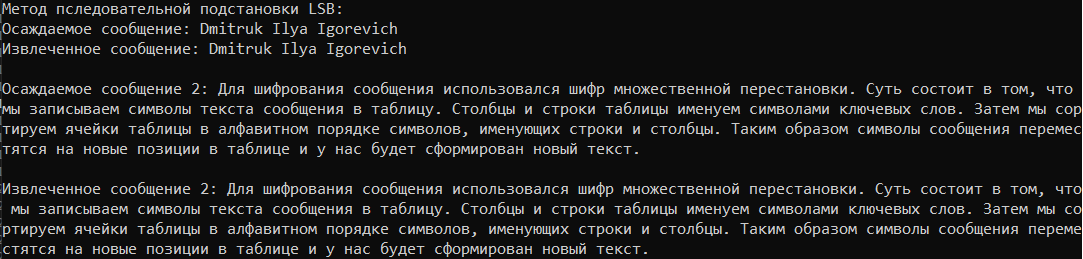


Рисунок 1.1 – Результат встраивания и извлечения битов сообщения

После выполнения встраивания битов сообщения произошло изменение битов изображения. В результате поменялось само изображения. Исходное изображение представлено на рисунке 1.2, а на рисунке 1.3 представлено изображение со встроенным текстом.



Рисунок 1.2 – Исходное изображение

  
  
Рисунок 1.3 – Изображение со встроенным текстом

Как мы видим, изображение практически не поменялось. Также была создана цветовая матрица. Результат создания цветовой матрицы представлен на рисунке 1.4.

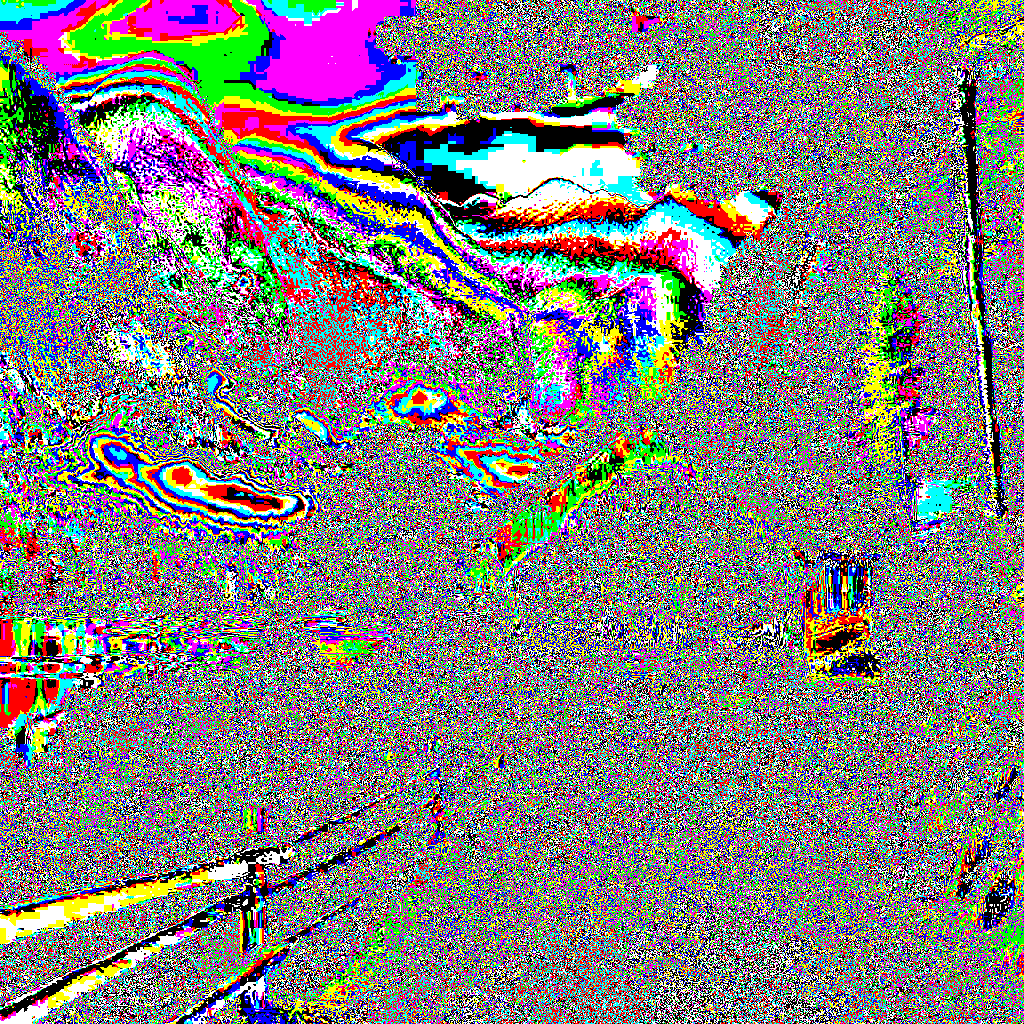


Рисунок 1.4 – Результат создания цветовой матрицы со встроенным текстом

# Задание 2.

Далее необходимо реализовать второй подход размещения битового потока сообщения. Для этого был создан метод реализующий данный функционал алгоритма LSB с вертикальным подходом. Программная реализация метода LSB с подходом вертикальной подстановки представлена в листинге 2.1.

public static Bitmap VerticalSteganographyLSB(string text, Bitmap image)

{int flag = 1, IndexOfSymbol = 0, SymbolNumb = 0, zeros = 0;

long pixelIndex = 0;

for (int i = 0; i < image.Height; i++)

{for (int j = 0; j < image.Width; j++)

{Color pixel = image.GetPixel(j, i);

int modifiedR = pixel.R - pixel.R % 2; int modifiedG = pixel.G - pixel.G % 2; int modifiedB = pixel.B - pixel.B % 2;

for (int k = 0; k < 3; k++)

{if (pixelIndex % 8 == 0)if ((pixelIndex - 1) % 3 < 2) image.SetPixel(j,i,Color.FromArgb(modifiedR, modifiedG, modifiedB));

return image; }

if (IndexOfSymbol >= text.Length)

flag = 0; else SymbolNumb = text[IndexOfSymbol++];

}

}return image;

}

Листинг 2.1 – Программная реализация метода LSB с подходом вертикальной постановки

Результат вставки и извлечения битов сообщения в изображения представлен на рисунке 2.1.

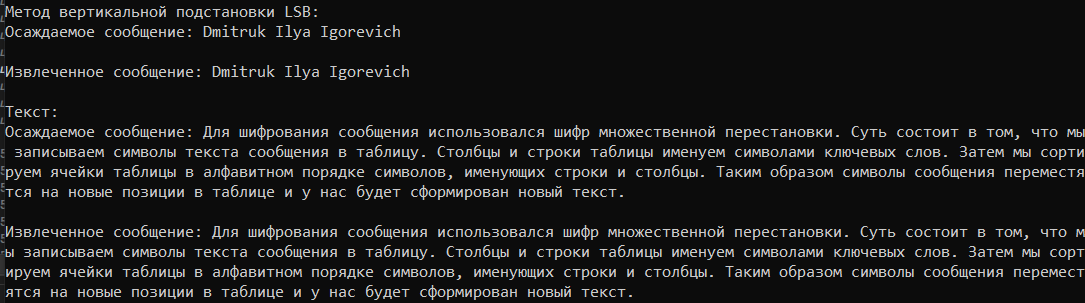


Рисунок 2.1 – Результат встраивания и извлечения битов сообщения

После выполнения встраивания битов сообщения произошло изменение битов изображения. В результате поменялось само изображения. Исходное изображение представлено на рисунке 2.2, а на рисунке 2.3 представлено изображение со встроенным текстом.



Рисунок 2.2 – Исходное изображение

  
  
Рисунок 2.3 – Изображение со встроенным текстом

Как мы видим, изображение практически не поменялось. Также была создана цветовая матрица. Результат создания цветовой матрицы представлен на рисунке 2.4.

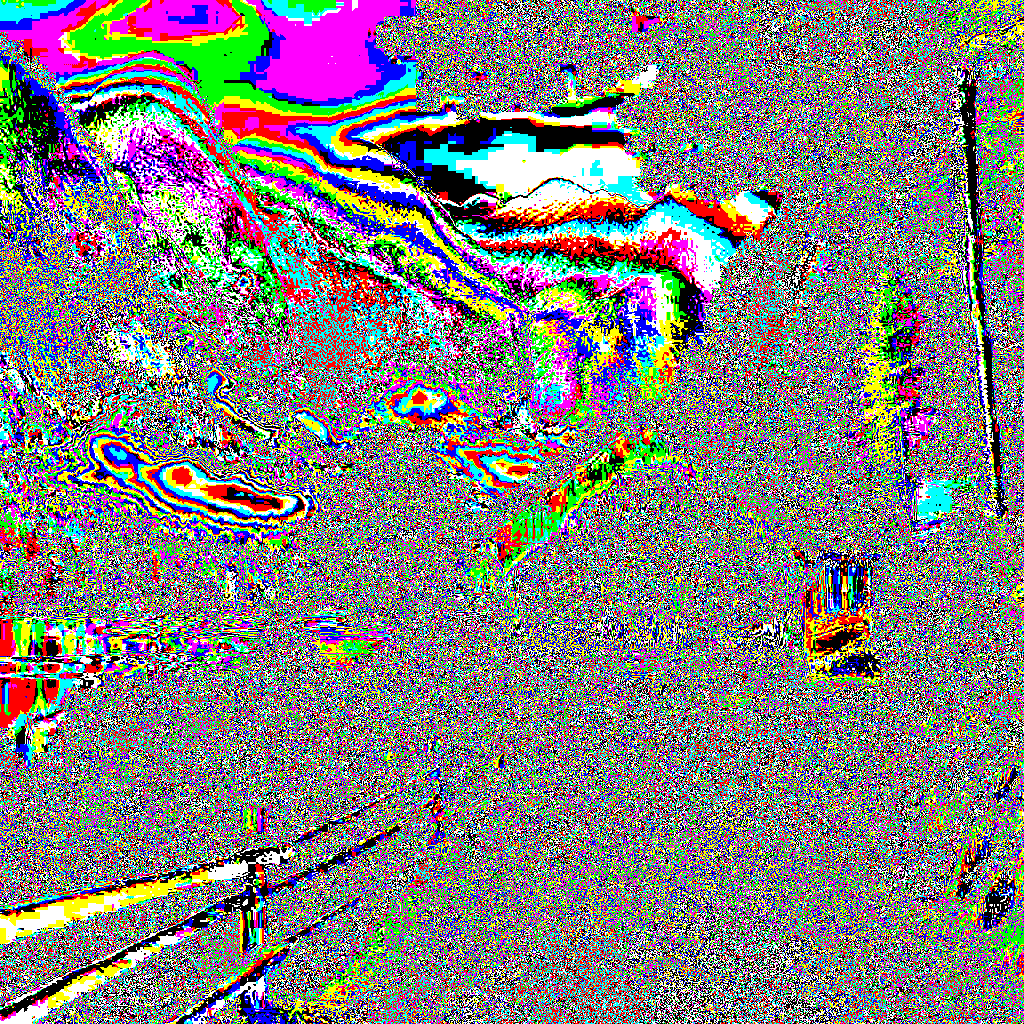


Рисунок 2.4 – Результат создания цветовой матрицы со встроенным текстом

**Вывод**

В результате выполнения данной лабораторной работы был изучен стеганографический алгоритм LSB позволяющий спрятать информацию внутри картинки так чтоб случайный человек даже и не понял, что она несет в себе информацию. Он строится на основе изменения младших битов пикселей изображения, в результате чего человеческому глазу данные изменения не заметны. Были созданы два подхода по осаждению текста вертикальный и последовательный.