Міністерство освіти і науки України
Харківський національний університет ім. В. Н. Каразіна
Факультет комп’ютерних наук
Кафедра безпеки інформаційних систем і технологій

Лабораторна робота №2

з навчальної дисципліни

«Стеганографія»

Виконала:

Студентка групи КБ-41
Кононченко А. В.

Перевірив:

Доцент
Нарєжній О. П.

Харків – 2020

Лабораторна робота №2

на тему:

«Приховування даних у просторовій області нерухомих зображень методом блокового вбудовування, метод квантування та методом “хреста”»

Мета роботи: закріпити теоретичні знання за темою «Приховування даних у просторовій області нерухомих зображень методом блокового вбудовування, методом квантування та методом "хреста"», набути практичних вмінь та навичок щодо розробки стеганографічних систем, дослідити властивості стеганографічних методів, що засновані на низькорівневих властивостях зорової системи людини (ЗСЛ).

Хід роботи

1 Реалізація алгоритму приховування і вилучення повідомлень методом блокового приховування

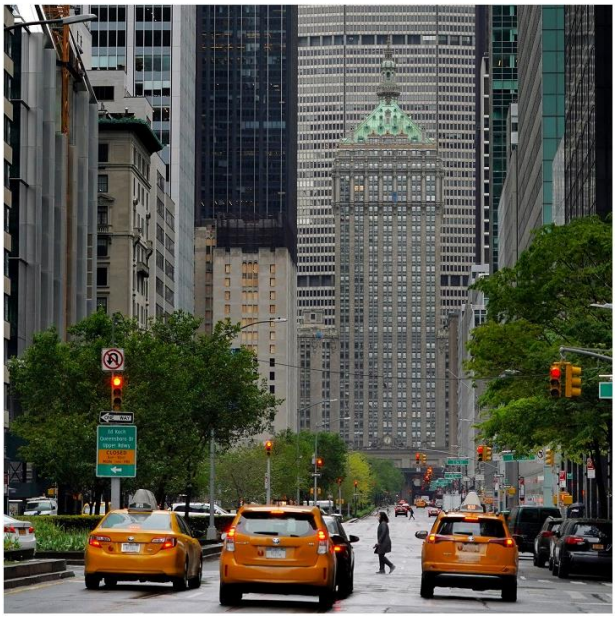


Рисунок 1.1 – Початкове зображення

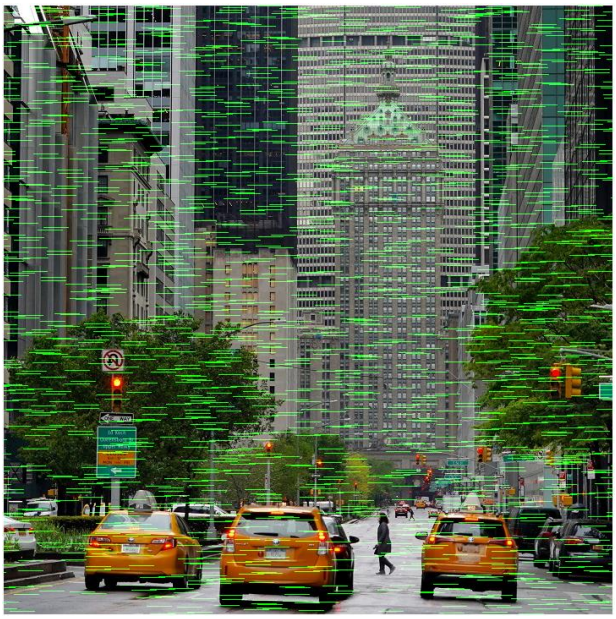


Рисунок 1.2 – Область, у яку буде проведено запис бітів повідомлення:

Повідомлення для приховування:

Text1 Text2 Text3

```
Hagrid looked at Harry with warmth and respect blazing in his eyes, but Harry, instead of feeling pleased and proud, felt quite sure there had been a horrible mistake. A wizard? Him? How could he possibly be? He'd spent his life being clouted by Dudley, and bullied by Aunt Petunia and Uncle Vernon; if he was really a wizard, why hadn't they been turned into warty toads every time they'd tried to lock him in his cupboard? If he'd once defeated the greatest sorcerer in the world, how come Dudley had always been able to kick him around like a football?
```

Біти повідомлення будуть приховуватися у каналі зеленого кольору зображення.

Вилучена інформація:

```
Hagrid looked at Harry with warmth and respect blazing in his eyes, but Harry, instead of feeling pleased and proud, felt quite sure there had been a horrible mistake. A wizard? Him? How could he possibly be? He'd spent his life being clouted by Dudley, and bullied by Aunt Petunia and Uncle Vernon; if he was really a wizard, why hadn't they been turned into warty toads every time they'd tried to lock him in his cupboard? If he'd once defeated the greatest sorcerer in the world, how come Dudley had always been able to kick him around like a football?
```

Програмна реалізація

Функція розрахунку псевдовипадкового інтервалу між блоками для вбудовування:

```
function step(x, k) {
    let count = 0;
    for (let i = 0; i < x.length; i++) {
        if (x[i] === '1') {
            count++;
        }
    }
    return k * count;
}
```

Функція вбудовування інформаційних бітів у пікселі зображення:

```
function putPixelsInBlocks(arr, st, key, input) {
    let nextSt = 0;
    for (let i = 0, index = 0; i < arr.length; i++) {
        if (i === st) {
            nextSt = st + 30;
            for (let j = st; j < nextSt; j++) {
                if (typeof input[index] === 'undefined') {
                    break;
                }
                arr[j] = replaceChars(arr[j], LSB, input[index]);
                index++;
                if (index === input.length) {
                    break;
                }
            }
            st += step(i.toString(2), key) + 30;
        }
    }
    return arr;
}
```

Функція кодування зображення:

```
let encode = function () {
    let key = parseInt(prompt('Enter key for encoding: ', '7'));
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input); //отримуємо текст у
двійковому вигляді
    let binaryLength = intToBinary(input.length);//запам’ятовуємо
довжину повідомлення у двійковому вигляді
    let imageData = ctx.getImageData(0, 0, width,
height);//отримуємо пікселі зображженя
    let encodeImageData =
arrayToBinary(imageData.data);//приводимо пікселі до двійкового
вигляду
    let greenChannel =
arrayToBinary(getGreenChannel(imageData.data));//дістаємо канал
зеленого кольору та приводимо його до двійкового вигляду
```

```
    let k = encodeImageData.length / 10000 / key;//розраховуємо
секретний ключ
    k = parseInt(k.toString().split('.')[0]);

    greenChannel = putPixelsInBlocks(greenChannel, startMark, k,
input, encodeImageData);//вбудовуємо інформаційні біти в канал
зеленого кольору

    for (let i = 0, index = 0; i < encodeImageData.length; i +=
4) {
        encodeImageData[i + 1] = greenChannel[index]; // green
        index++;
    }//вносимо зміни зеленого кольору до масиву пікселів
зображення
    for (let i = 0; i < binaryLength.length; i++) {
        encodeImageData[i] = replaceChars(encodeImageData[i],
LSB, binaryLength[i]);
    }//вбудовуємо довжину повідомлення у початок зображення

    encodeImageData =
parseFromBinary(encodeImageData);//приводимо бінарний вигляд
пікселів зображення до десяткового вигляду відповідно до моделі
RGB
    imageData.data.set(encodeImageData);//приймаємо внесені зміни
    ctx.putImageData(imageData, 0, 0);//у зображенні
    alert("Your data was successfully encoded!");
};
```

Функція вилучення інформаційних бітів з пікселів зображення:

```
function getBitsInBlocks(arr, st, key, inputLength,
imageDataLength) {
    let output = '', nextSt = 0;
    for (let i = 0, index = 0; i < arr.length; i++) {
        if (i === st) {
            nextSt = st + 30;
            for (let j = st; j < nextSt; j++) {
                output += getChar(arr[j], LSB);
                index++;
            }
            st += step(i.toString(2), key) + 30;
            if (index === inputLength) {
                break;
            }
        }
    }
    return output;
}
```

Функція декодування зображення:

```
let decode = function () {
    let key = parseInt(prompt('Enter key for decoding: ', '7'));
    let tmp = document.getElementsByClassName('input-wrapper');
    let output = document.getElementById('messageOutput');
    let decodeImageData = arrayToBinary(ctx.getImageData(0, 0,
width, height).data);//отримуємо двійковий вигляд масиву пікселів
зображення
    let greenChannel =
getGreenChannel(decodeImageData);//дістаємо канал зеленого
кольору
    let k = decodeImageData.length / 10000 / key;//обраховуємо
значення секретного ключа
    k = parseInt(k.toString().split('.')[0]);
    let inputLength = '';
    for (let i = 0; i < 24; i++) {
        inputLength += getChar(decodeImageData[i], LSB);
    }//вилучаємо довжину вбудованого повідомлення
    inputLength = parseInt(inputLength, 2);
    let res = getBitsInBlocks(greenChannel, startMark, k,
inputLength, decodeImageData.length);//вилучаємо інформаційні
біти вбудованого повідомлення
    output.innerText += convertBinaryToText(res);//конвертуємо
двійкове повідомлення у текст
};
```

2 Реалізація алгоритму приховування і вилучення повідомлень методом квантування

Даний метод використовує в якості секретного ключа таблицю квантування, яка містить значення яскравості від -255 до 255 та відповідні їм бінарні значення, розраховані псевдовипадковим чином. Бінарні значення обчислюються на основі обраного для вбудовування каналу кольору, представленого у двійковому вигляді. При цьому для кожного елементу масиву кольору розраховується сума за модулем 2 з булевим додаванням до результату одиниці при кожному третьому елементі. Фрагмент таблиці квантування представлений на рис. 2.1.

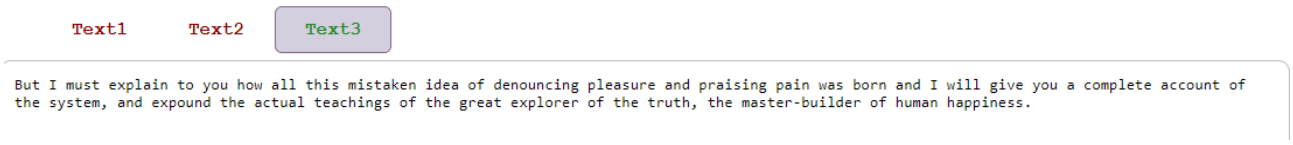
-255	0	…		…		…	
-254	0	-6	0			242	1
-253	1	-5	0			243	0
-252	0	-4	1			244	0
-251	0	-3	0			245	1
-250	1	-2	0			246	0
-249	0	-1	1			247	0
-248	0	0	0			248	1
-247	1	1	0			249	0
-246	0	2	1			250	0
-245	0	3	0			251	1
-244	1	4	0			252	0
-243	0	5	1			253	0
-242	0	6	0			254	1
…		…				255	0

Рисунок 2.1 – Фрагмент згенерованої таблиці квантування

Рисунок 2.2 – Початкове зображення

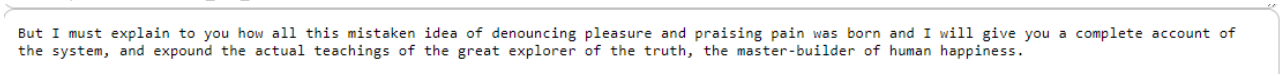
Рисунок 2.3 – Зображення після вбудовування повідомлення

Повідомлення для приховування:



Біти повідомлення будуть приховуватися у каналі синього кольору зображення.

Вилучена інформація:



Програмна реалізація

Функція вбудовування інформаційних бітів у пікселі зображення:

```
function putPixels(arr, st, input, imageDataArr, quantTable) {
    let key = parseInt(prompt('Enter key for decoding: ', '6'));

    for (let i = 0, index = 0; i < imageDataArr.length; i++) {
        if (i === st) {
            let delta = arr[i] - arr[i + 1];
            let tmp = quantTable.find(i => i[0] === delta);
            if (typeof tmp === 'undefined') {
                alert('Your key is too big, so not all information will
be encoded..');
                break;
            }
            if (tmp[1] !== parseInt(input[index])) {
                let bias = 0;
                for (let j = quantTable.indexOf(tmp); j <
quantTable.length; j++) {
                    if (quantTable[j][1] === parseInt(input[index])) {
                        if (arr[i] + bias > 255) {
                            do {
                                arr[i]--;
                                j--;
                            } while (quantTable[j][1] !==
parseInt(input[index]));
                        }
                        arr[i] += bias;
                        break;
                    }
                    bias++;
                }
            }
            st += step(i.toString(2), key);
            index++;
        }
        if (index === input.length) {
            break;
        }
    }
    return arr;
}
```

Для вбудовування інформаційних бітів знаходиться різниця сусідніх бітів обраного каналу кольору. Далі виконується пошук різниці у таблиці квантування

та порівняння закріпленого за нею бінарного значення з бітом повідомлення. Якщо значення не співпадають, необхідно провести пошук наступного в таблиці такого бінарного значення, аби інформаційний біт з ним виявився однаковим. У даному випадку використовується пошук вправо, а у ситуації виходу за границі таблиці (тобто піксель набуває значення більше 255) – зворотній пошук (вліво).

Після того, як таке значення знайшлось, потрібно збільшити (або зменшити) поточний піксель на кількість перебраних значень таблиці, тобто зробити його таким, щоб відповідав таблиці квантування.

Функція кодування зображення:

```
function encode() {
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input);//отримуємо текст у
двійковому вигляді
    let imageData = ctx.getImageData(0, 0, width,
height);//отримуємо пікселі зображженя

    let changedPixels = putPixels(blueChannel, startMark, input,
imageData.data, quantTable);//вбудовуємо у канал синього кольору
біти повідомлення

    for (let i = 0, index = 0; i < imageData.data.length; i += 4)
{
        imageData.data[i + 2] = changedPixels[index++];
    }
    console.log(imageData.data);//вносимо необхідні зміни у масив
пікселів зображення

    ctx.putImageData(imageData, 0, 0);//приймаємо внесені зміни
    alert("Your data was successfully encoded!");
}
```

Функція вилучення інформаційних бітів з пікселів зображення:

```
function getPixels(arr, st, key, imageData, quantTable) {
    let output = '', inputLength = '';
    for (let i = 0; i < 48; i += 2) {
        let delta = arr[i] - arr[i + 1];
        let tmp = quantTable.find(item => item[0] === delta);
        inputLength += tmp[1];
    }
    inputLength = parseInt(inputLength, 2);

    for (let i = 0, index = 0; i < arr.length; i++) {
        if (i === st) {
            let delta = arr[i] - arr[i + 1];
            let tmp = quantTable.find(i => i[0] === delta);
            output += tmp[1];

            st += step(i.toString(2), key);
            index++;
        }
```

```
        if (index === inputLength) {
            break;
        }
    }
    return output;
}
```

Для реалізації вилучення бітів повідомлення із зображення необхідно лише провести пошук значення різниці двох сусідніх пікселів обраного каналу кольору в таблиці квантування. Прикріплене за шуканою дельтою бінарне значення і буде вилученим інформаційним бітом.

Функція декодування зображення:

```
function decode() {
    let key = parseInt(prompt('Enter key for decoding: ', '6'));
    let tmp = document.getElementsByClassName('input-wrapper');
    let output = document.getElementById('messageOutput');
    let decodeImageData = ctx.getImageData(0, 0, width,
height);//отримання даних зображення
    let blueChannel =
getBlueChannel(decodeImageData.data);//отримання каналу синього
кольору
    let decodedMessage = getPixels(blueChannel, startMark, key,
decodeImageData.data, quantTable);//вилучення з каналу синього
кольору вбудованих даних
    output.innerText +=
convertBinaryToText(decodedMessage);//конвертація вилучених
бінарних даних в текст
}
```

3 Реалізація алгоритмів вбудовування та вилучення повідомлень методом «хреста» (метод Куттера-Джордана-Боссена)

При реалізації даного методу вилучення вбудованої інформації носить ймовірнісний характер, що призводить до спотворень повідомлення. Рівень помилок, що вносяться, залежить від параметру λ – константи, яка визначає «енергію» вбудованого сигналу. Вбудовування секретного повідомлення відбувається у канал синього кольору.

Рисунок 3.1 – Початкове зображення

Рисунок 3.2 – Зображення з після вбудовування повідомлення

На рис. 3.2 відображено зображення закодоване при λ = 0.2. При збільшенні значення λ збільшується рівень внесених спотворень (рис. 3.3).

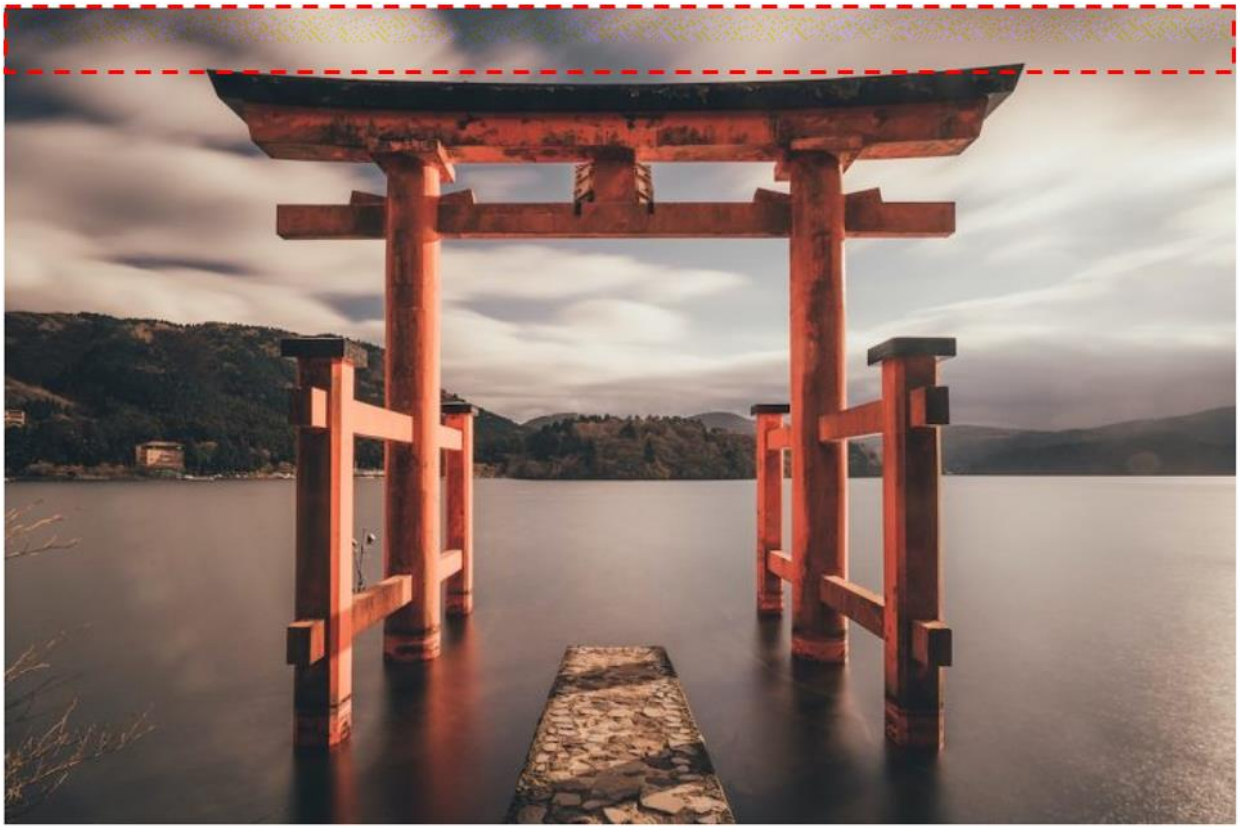


Рисунок 3.3 – Внесені спотворення при λ = 0.8

Вбудована та вилучена інформація:

Вбудоване повідомлення	Вилучене повідомлення
Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. The use of steganography can be combined with encryption as an extra step for hiding or protecting data.	Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. The use of steganography can be combined with encryption as an extra step for hiding or protecting data.

Графіки залежності правильності вилучення інформаційних бітів та внесених спотворень від величини λ відображені на рис. 3.4 та рис. 3.5 відповідно.

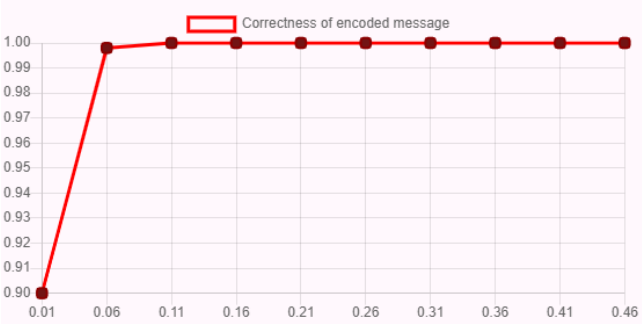


Рисунок 3.4 – Графік залежності правильності вилучених біт від величини λ

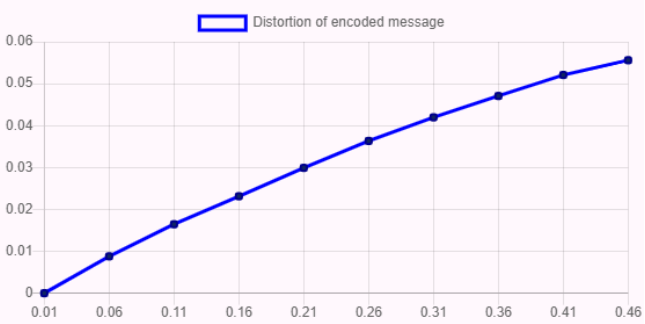


Рисунок 3.5 – Графік залежності внесених спотворень від величини λ

Необхідно зазначити, що правильність вилученого повідомлення залежить від самого зображення. Нижче приведені графіки залежності для різних типів зображень.

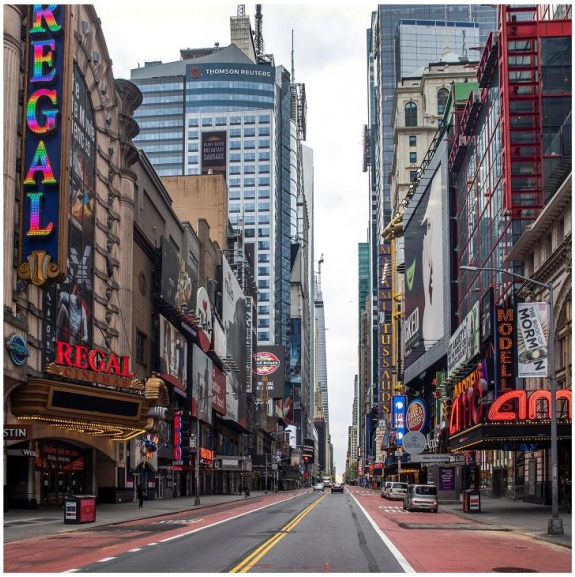


Рисунок 3.6 – Контрастне зображення

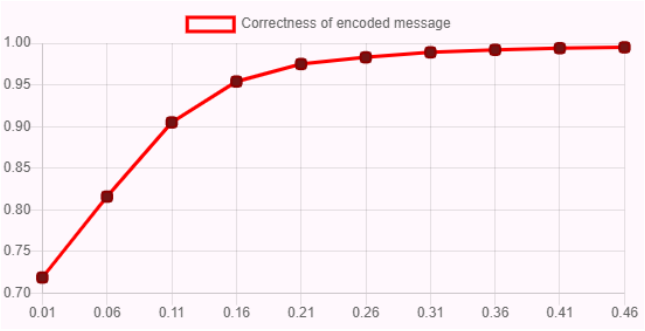


Рисунок 3.7 – Графік залежності правильності вилученого повідомлення від величини λ

Рисунок 3.8 – Чорно-біле зображення

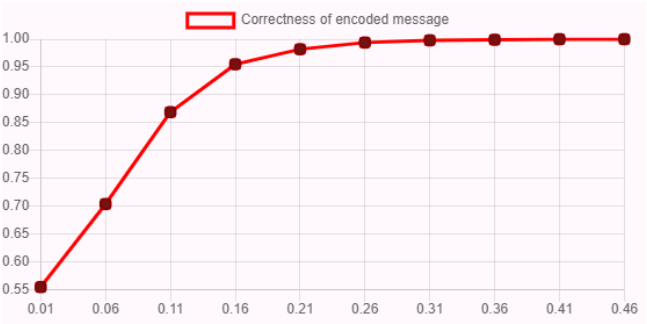


Рисунок 3.9 – Графік залежності правильності вилученого повідомлення від величини λ

Рисунок 3.10 – Зображення з різкими перепадами кольорів (контрастне зображення)

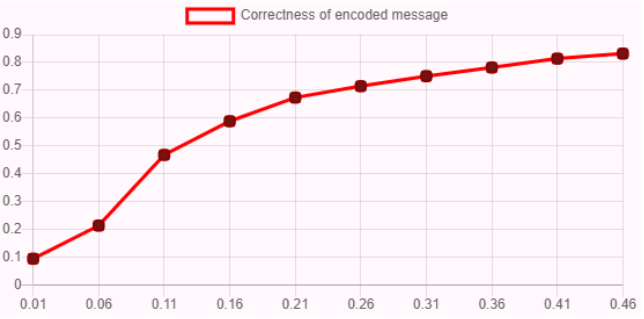


Рисунок 3.11 – Графік залежності правильності вилученого повідомлення від величини λ

З приведених графіків можна зробити висновок, що найвищої ймовірності правильного вилучення секретного повідомлення можливо досягти при використанні більш монотонних зображень, бажано з природними для ока людини переходами (наприклад, пейзажі).

Програмна реалізація

Константи:

```
let lambda = 0.3,//енергія вбудованого сигналу
    sigma = 5;//розмір «крила хреста»
```

Функція вбудовування інформаційних бітів у пікселі зображення:

```
function putBits(input, imageData, lambda) {
   for (let j = sigma, index = 0; j < height - sigma; j++) {
        for (let i = j; i < width - sigma; i += sigma) {
            if (index === input.length) {
                break;
            }
            let color = getRGB(imageData, j * width + i);
            let br = 0.29890 * color[0] + 0.58662 * color[1] +
0.11448 * color[2];//розрахунок повнокольрової яскравості
            let newBlue = parseInt(((color[2] + ((2 *
input[index] - 1)) * br * lambda)).toString());//розрахунок
нового значення синього кольору
            imageData[j * width + i] = [color[0], color[1],
newBlue < 0 ? 0 : Math.min(newBlue, 255)];//внесення змін у
піксель кольору
            index++;
        }
    }
    let binaryLength = intToBinary  (input.length);
    for (let i = imageData.length - 24, index = 0; i <
imageData.length; i++) {
        let tmp = make8bit(getRGB(imageData, i)[1].toString(2));
        imageData[i][1] = parseInt(replaceChars2(tmp, LSB,
binaryLength[index++]), 2);//вбудовування довжини повідомлення,
що приховується
    }
    return imageData;
}
```

Функція кодування зображення:

```
let encode = function () {
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input););//отримуємо текст у
двійковому вигляді
    let inputArr = [];
    for (let i = 0; i < input.length; i++) {
        inputArr.push(parseInt(input[i]));
    }
    let imageData = ctx.getImageData(0, 0, width,
height);//отримуємо пікселі зображженя
    let imagePixels = dividePixels(imageData.data);
    let changed = putBits(inputArr, imagePixels,
lambda););//вбудовуємо у канал синього кольору біти повідомлення
    changed = mergePixels(changed);
    for (let i = 0; i < imageData.data.length; i++) {
        imageData.data[i] = changed[i];
```

```
    });//вносимо необхідні зміни у масив пікселів зображення
    ctx.putImageData(imageData, 0, 0););//приймаємо зміни
    alert("Your data was successfully encoded!");
}
```

Функція вилучення інформаційних бітів з пікселів зображення:

```
function getPixels(imageData) {
    let inputLength = '';
    for (let i = imageData.length - 24; i < imageData.length;
i++) {
        let tmp = make8bit(getRGB(imageData, i)[1].toString(2));
        inputLength += getChar(tmp, LSB);
    } //дістаємо довжину вбудованого повідомлення
    inputLength = parseInt(inputLength, 2);
    let output = [], count = 0;
    for (let j = sigma; j < height - sigma; j++) {
        for (let i = j; i < width - sigma; i += sigma) {
            if(count === inputLength){
                break;
            }
            let color = getRGB(imageData, j * width +
i);//отримуємо колір в моделі RGB
            let sum = 0.0;
            for (let k = i - sigma; k <= i + sigma; k++) {
                if (i !== k) {
                    sum += getRGB(imageData, j * width + k)[2];
                }
            }
            for (let k = j - sigma; k <= j + sigma; k++) {
                if (j !== k) {
                    sum += getRGB(imageData, k * width + i)[2];
                }
            }//проходимося по значенням «хресту» і сумуємо їх
            sum /= 4 * sigma;//за формулою обчислюємо
передбачуване значення синього кольору
            let diff = color[2] - sum;//знаходимо різницю між
поточним та прогнозованим значеннями синього кольору
            output.push(diff > 0 ? '1' : '0');//якщо різниця
більше «0», вилучаємо «1», інакше – «0»
            count++;
        }
    }
    return output.join('');
}
```

Функція декодування зображення:

```
let decode = function () {
    let tmp = document.getElementsByClassName('input');
    let output = document.getElementById('messageOutput');
    let decodeImageData = ctx.getImageData(0, 0, width,
height);//отримання даних зображення
    let decodeImageDataArr = dividePixels(decodeImageData.data);
    let decodedMessage =
getPixels(decodeImageDataArr););//вилучення з каналу синього
кольору вбудованих даних
    output.innerText +=
convertBinaryToText(decodedMessage););//конвертація вилучених
бінарних даних в текст
}
```

Додаткові функції:

```
function getRGB(imageData, index) {
    for (let i = 0; i < imageData.length; i++) {
        if (i === index) {
            return imageData[i];
        }
    }
} //отримання і-го пікселю у вигляді RGB моделі

function dividePixels(imageData) {
    let res = [];
    for (let i = 0; i < imageData.length; i += 4) {
        let tmp = [imageData[i], imageData[i + 1], imageData[i +
2]];
        res.push(tmp);
    }
    return res;
} //розбиття масиву даних зображення на пікселі у вигляді RGB
моделі

function mergePixels(imageDataArr) {
    let res = [];
    for (let i = 0; i < imageDataArr.length; i++) {
        res.push([].concat(imageDataArr[i], 255));
    }
    return res.flat();
} //об’єдання масиву RGB у масив даних зображення
```

Висновки

При виконанні лабораторної роботи було реалізовано три методи приховування даних у просторовій області нерухомих зображень: метод блокового приховування, метод квантування та метод Куттера-Джордана-Боссена. Було досліджено ймовірнісні характеристики правильного вилучення повідомлення та рівня внесених спотворень для методу «хреста».