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

Лабораторна робота №4

з навчальної дисципліни

«Стеганографія»

Виконала:

Студентка групи КБ-41
Кононченко А. В.

Перевірив:

Доцент
Нарєжній О. П.

Харків – 2020

Лабораторна робота №4

на тему:

«Приховування даних у частотній області нерухомих зображень на основі кодування різниці абсолютних значень коефіцієнтів дискретно-косинусного перетворення»

Мета роботи: закріпити теоретичні знання з теми «Приховування даних у частотній області нерухомих зображень», набути практичних вмінь та навичок з розробки стеганографічних систем, дослідити властивості стеганографічних методів, заснованих на низькорівневих властивостях зорової системи людини (ЗСЛ), зокрема частотної чутливості.

Хід роботи

1. Реалізація алгоритмів прямого та зворотного дискретно-косинусного перетворення. Дослідження ефекту частотної чутливості зорової системи людини.

Функція прямого дискретно-косинусного перетворення:

```
function directDCT(inputData) {
    let outputData = new matrix(N, N);
    for (let u = 0; u < N; u++) {
        for (let v = 0; v < N; v++) {
            let sum = 0.0;
            for (let x = 0; x < N; x++) {
                for (let y = 0; y < N; y++) {
                    sum += inputData[x][y] * Math.cos(((2 * x + 1)
/ (2.0 * N)) * u * Math.PI) * Math.cos(((2 * y + 1) / (2.0 * N)) *
v * Math.PI);
                }
            }
            sum *= c[u][v] / 4.0;
            outputData[u][v] = parseInt(sum.toString());
        }
    }
    return outputData;
}
```

Функція зворотнього дискретно-косинусного перетворення:

```
function inverseDCT(inputData) {
    let outputData = new matrix(N, N);
    for (let x = 0; x < N; x++) {
        for (let y = 0; y < N; y++) {
            let sum = 0.0;
            for (let u = 0; u < N; u++) {
                for (let v = 0; v < N; v++) {
                    sum   +=   c[u][v]   *   inputData[u][v]   *
Math.cos(((2 * x + 1) / (2.0 * N)) * u * Math.PI) * Math.cos(((2 *
y + 1) / (2.0 * N)) * v * Math.PI);
                }
            }
```

```
                sum /= 4.0;
                outputData[x][y] = parseInt(sum.toString());
            }
        }
        return outputData;
    }
```

Функція розбиття зображення на блоки:

```
function imageToBlock() {
    let nn = height / N, mm = width / N;
    let r = new matrix(nn, mm);
    for (let x = 0; x < mm; x++) {
        for (let y = 0; y < nn; y++) {
            let RR = new matrix(N, N);
            for (let i = 0; i < N; i++) {
                for (let j = 0; j < N; j++) {
                    RR[i][j] = redMatrix[i + x * N][j + y * N];
                }
            }
            r[x][y] = RR;
        }
    }
    return r;
}
```

Функція виконання ДКП для кожного блоку:

```
function useDCTtoBlocks(){
    let nn = height / N, mm = width / N, blockDCT = new matrix (nn,
mm);
    for (let i = 0; i < mm; i++) {
        for (let j = 0; j < nn; j++) {
            blockDCT[i][j] = directDCT(blockedImage[i][j])
        }
    }
    return blockDCT;
}
```

Приклад блоків контейнера-зображення в просторовій та частотній областях:

(in…	0	1	2	3	4	5	6	7
0	81	93	94	93	95	95	96	98
1	80	92	93	92	94	94	95	97
2	80	92	93	92	94	94	95	96
3	80	91	92	91	94	93	95	95
4	78	89	90	89	91	91	93	94
5	77	88	88	86	90	90	91	93
6	74	85	85	83	87	87	88	89
7	70	80	80	78	81	81	83	84

Блок зображення

(i…	0	1	2	3	4	5	6	7
0	709	-27	-10	-14	-10	-9	-5	1
1	31	0	-1	-1	0	0	0	0
2	-12	0	1	0	0	0	0	0
3	7	-1	1	0	0	0	0	0
4	-3	0	0	0	0	0	0	0
5	3	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	-1	0	0	0	0	0	0	0

Результат прямого ДКП над блоком зображення

Функція переходу до моделі YCbCr:

```
function toYCbCrModel(pixelsRGB) {
    let YCbCrModel = [];
    for (let i = 0; i < pixelsRGB.length; i++) {
        YCbCrModel.push([
            parseInt((0.299 * pixelsRGB[i][0] + 0.587 *
pixelsRGB[i][1] + 0.114 * pixelsRGB[i][2]).toString()),
            parseInt((-0.169 * pixelsRGB[i][0] - 0.331 *
pixelsRGB[i][1] + 0.5 * pixelsRGB[i][2] + 128).toString()),
            parseInt((.5 * pixelsRGB[i][0] - 0.4187 *
pixelsRGB[i][1] - 0.081 * pixelsRGB[i][2] + 128).toString())
        ]);
    }
    return YCbCrModel;
}
```

Функція повернення від моделі YCbCr до RGB:

```
function toRGB(pixelsYCbCr) {
    let RGBmodel = [];
    for (let i = 0; i < pixelsYCbCr.length; i++) {
        RGBmodel.push([
            parseInt(((pixelsYCbCr[i][0] + 1.402 *
(pixelsYCbCr[i][2] - 128))).toString()),
            parseInt(((pixelsYCbCr[i][0] - 0.33414 *
(pixelsYCbCr[i][1] - 128) - 0.71414 * (pixelsYCbCr[i][2] -
128))).toString()),
            parseInt(((pixelsYCbCr[i][0] + 1.772 *
(pixelsYCbCr[i][1] - 128))).toString())
        ]);
    }
    return RGBmodel;
}
```

Представлення зображення у різних кольорових моделях:

Оригінальне зображення в RGB

Зображення в моделі YCbCr

Повернення до RGB з YCbCr

Для дослідження ефекту частотної чуттєвості зорової системи людини змінимо величини коефіцієнтів дискретно-косинусного перетворення в низькочастотній і високочастотній області зображення. Внесені зміни будемо оцінювати візуально, для чого виконаємо зворотне дискретно-косинусне перетворення над зміненим масивом коефіцієнтів.

Функція внесення змін у високочастотній області:

```
function changeHighFrequency() {
    let contWrapper1 = document.getElementById('canvas1');
    let container12 = document.createElement('canvas');
    container12.id = 'container12';
    contWrapper1.appendChild(container12);
    container12.height = height;
    container12.width = width;
    let ctx12 = container12.getContext('2d');
    let divided = dividePixels(imagePixels);
    let YCbCr = toYCbCrModel(divided); //RGB to YCbCr transform
    let merged = mergePixels(YCbCr);
    let Ychannel = getRedChannel(merged); // getting Y from YCbCr
    let blockedImage = imageToBlock(toMatrix(Ychannel));
    let blockedImageWithDCT = useDCTtoBlocks(blockedImage);
    for (let i = 0; i < blockedImageWithDCT.length; i++) {
        for (let j = 0; j < blockedImageWithDCT[i].length; j++) {
            blockedImageWithDCT[i][j][0][0]                    +=
(parseInt((blockedImageWithDCT[i][j][0][0] * 0.3).toString()));
        }
    }
    Let                  blockedImageWithIDCT                  =
useIDCTtoBlocks(blockedImageWithDCT);
    let changedYchannel = flatBlocked(blockedImageWithIDCT);
    for (let i = 0, index = 0; i < merged.length; i++) {//write
changes to YCbCr arr
        if (i % 4 === 0) {
            merged[i] = changedYchannel[index];
            index++;
        }
    }
    let div = dividePixels(merged);
    let backToRGB = toRGB(div);
    let backToRGBarr = mergePixels(backToRGB);
    console.log(backToRGBarr);

    imageDataDCT.data.set(backToRGBarr);
    ctx12.putImageData(imageDataDCT, 0, 0);
}
```

Приклад внесених змін у високочастотну область одного з блоків зображення:

(i…	0	1	2	3	4	5	6	7
0	488	-20	-10	-15	-10	-11	-6	0
1	16	0	-2	-1	0	0	0	0
2	-7	1	0	0	0	0	0	0
3	4	-1	0	0	0	0	0	0
4	-2	0	0	0	0	0	0	0
5	3	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0
7	-1	0	0	0	0	0	0	0

Коефіцієнти ДКП оригінального блоку зображення

(i…	0	1	2	3	4	5	6	7
0	634	-20	-10	-15	-10	-11	-6	0
1	16	0	-2	-1	0	0	0	0
2	-7	1	0	0	0	0	0	0
3	4	-1	0	0	0	0	0	0
4	-2	0	0	0	0	0	0	0
5	3	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0
7	-1	0	0	0	0	0	0	0

Коефіцієнти ДКП зміненого блоку зображення

(i…	0	1	2	3	4	5	6	7
0	52	64	65	64	66	64	65	67
1	52	64	64	63	65	63	64	66
2	52	64	65	63	64	63	64	65
3	52	63	64	63	65	63	66	66
4	50	61	62	61	63	62	64	65
5	51	62	62	60	62	62	63	65
6	49	60	60	58	61	61	62	63
7	47	57	57	55	57	56	58	58

Оригінальний блок зображення зі ЗДКП

(i…	0	1	2	3	4	5	6	7
0	70	82	83	82	83	82	83	84
1	70	81	82	81	82	81	82	84
2	70	82	82	81	82	81	82	83
3	70	81	82	80	82	81	83	84
4	69	80	80	79	81	80	81	83
5	69	80	80	78	80	79	81	82
6	67	78	77	76	78	78	79	81
7	65	75	75	73	75	74	76	77

Змінений блок зображення зі ЗДКП

Візуальне порівняння оригінального зображення та зображення, кожен блок якого піддався змін у високочастотній області:

Оригінальне зображення

Зображення зі змінами

Функція внесення змін у низькочастотній області:

```
function changeLowFrequency() {
    imageDataDCT.data.set(originalPixels);
    let contWrapper2 = document.getElementById('canvas2');
    /**Отрисовка оригинального канваса **/
    let originCont = document.createElement('canvas');
    let originCtx = originCont.getContext('2d');
    contWrapper2.appendChild(originCont);
    originCont.height = height;
    originCont.width = width;
    originCtx.putImageData(imageDataDCT, 0, 0);
    /**Отрисовка изменённого канваса **/
    let container22 = document.createElement('canvas');
    container22.id = 'container22';
    contWrapper2.appendChild(container22);
    let ctx22 = container22.getContext('2d');
    container22.height = height;
    container22.width = width;
    let divided = dividePixels(imagePixels);
    let YCbCr = toYCbCrModel(divided); //RGB to YCbCr transform
    let merged = mergePixels(YCbCr);
    console.log(merged);

    let Ychannel = getRedChannel(merged); // getting Y from YCbCr
    let blockedImage = imageToBlock(toMatrix(Ychannel));
    let blockedImageWithDCT = useDCTtoBlocks(blockedImage);
    for (let i = 0; i < blockedImageWithDCT.length; i++) {
        for (let j = 0; j < blockedImageWithDCT[i].length; j++) {
            blockedImageWithDCT[i][j][N - 1][N - 1] +=
blockedImageWithDCT[i][j][N - 1][N - 1];
        }
    }
    let blockedImageWithIDCT =
useIDCTtoBlocks(blockedImageWithDCT);
    let changedYchannel = flatBlocked(blockedImageWithIDCT);
    for (let i = 0, index = 0; i < merged.length; i++) {//write
changes to YCbCr arr
        if (i % 4 === 0) {
            merged[i] = changedYchannel[index];
            index++;
        }
    }
    let div = dividePixels(merged);
    let backToRGB = toRGB(div);
    let backToRGBarr = mergePixels(backToRGB);
    imageDataDCT.data.set(backToRGBarr);
    ctx22.putImageData(imageDataDCT, 0, 0);
}
```

Приклад внесених змін у низькочастотну область одного з блоків зображення:

(i…	0	1	2	3	4	5	6	7
0	488	-20	-10	-15	-10	-11	-6	0
1	16	0	-2	-1	0	0	0	0
2	-7	1	0	0	0	0	0	0
3	4	-1	0	0	0	0	0	0
4	-2	0	0	0	0	0	0	0
5	3	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0
7	-1	0	0	0	0	0	0	0

Коефіцієнти ДКП оригінального блоку зображення

(i…	0	1	2	3	4	5	6	7
0	488	-20	-10	-15	-10	-11	-6	0
1	16	0	-2	-1	0	0	0	0
2	-7	1	0	0	0	0	0	0
3	4	-1	0	0	0	0	0	0
4	-2	0	0	0	0	0	0	0
5	3	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0
7	-1	0	0	0	0	0	0	0

Коефіцієнти ДКП зміненого блоку зображення

(i…	0	1	2	3	4	5	6	7
0	52	64	65	64	66	64	65	67
1	52	64	64	63	65	63	64	66
2	52	64	65	63	64	63	64	65
3	52	63	64	63	65	63	66	66
4	50	61	62	61	63	62	64	65
5	51	62	62	60	62	62	63	65
6	49	60	60	58	61	61	62	63
7	47	57	57	55	57	56	58	58

Оригінальний блок зображення зі ЗДКП

(i…	0	1	2	3	4	5	6	7
0	52	64	65	64	66	64	65	67
1	52	64	64	63	65	63	64	66
2	52	64	65	63	64	63	64	65
3	52	63	64	63	65	63	66	66
4	50	61	62	61	63	62	64	65
5	51	62	62	60	62	62	63	65
6	49	60	60	58	61	61	62	63
7	47	57	57	55	57	56	58	58

Змінений блок зображення зі ЗДКП

Візуальне порівняння оригінального зображення та зображення, кожен блок якого піддався змін у низькочастотній області:

Оригінальне зображення

Зображення зі змінами

2. Реалізація алгоритмів вбудовування та вилучення повідомлень до частотної області зображень (метод Коха–Жао)

Функція вбудовування інформаційних бітів у зображення:

```
function encode() {
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input);
    let imageData = ctx.getImageData(0, 0, width, height);
    let divided = dividePixels(imageData.data);
    let YCbCr = toYCbCrModel(divided); //RGB to YCbCr transform
    let merged = mergePixels(YCbCr);
    let Ychannel = getRedChannel(merged); // getting Y from YCbCr
    let YchannelMatrix = toMatrix(Ychannel);
    let blockedImage = imageToBlock(YchannelMatrix);
    let blockedImageWithDCT = useDCTtoBlocks(blockedImage);
    let index = 0;
    for (let i = 0; i < Math.min(blockedImageWithDCT.length, input.length);
i++) {
        let i2 = parseInt((N / 3).toString()), i1 = i2 + i2;
        for (let j = 0; j < blockedImageWithDCT[i].length; j++) {
            if (input[index] === '1') {
                if (blockedImageWithDCT[i][j][i1][i2] > 0) {
                    blockedImageWithDCT[i][j][i1][i2] =
                            Math.abs(blockedImageWithDCT[i][j][i2][i1]) + Pr
                } else {
                    blockedImageWithDCT[i][j][i1][i2] =
                           -Math.abs(blockedImageWithDCT[i][j][i2][i1]) - Pr
                }
                index++;
            } else {
                if (blockedImageWithDCT[i][j][i2][i1] > 0) {
                    blockedImageWithDCT[i][j][i2][i1] =
                            Math.abs(blockedImageWithDCT[i][j][i1][i2]) + Pr
                } else {
                    blockedImageWithDCT[i][j][i2][i1] =
                           -Math.abs(blockedImageWithDCT[i][j][i1][i2]) - Pr
                }
                index++;
            }
        }
    }
    let blockedImageWithIDCT = useIDCTtoBlocks(blockedImageWithDCT);
    let changedYchannel = flatBlocked(blockedImageWithIDCT);
    for (let i = 0, index = 0; i < merged.length; i++) {
        if (i % 4 === 0) {
            merged[i] = changedYchannel[index];
            index++;
        }
    }
    let div = dividePixels(merged);
    let backToRGB = toRGB(div);
    let backToRGBarr = mergePixels(backToRGB);
    imageData.data.set(backToRGBarr);
    ctx.putImageData(imageData, 0, 0);
}
```

Функція вилучення інформаційних бітів із зображення:

```
let decode = function () {
    let tmp = document.getElementsByClassName('input-wrapper');
    console.log(tmp[0].children.length);
    let output;
    if (tmp[0].children.length <= 2) {
        output = document.createElement('textarea');
        output.id = 'messageOutput';
        tmp[0].appendChild(output);
    } else {
        output = document.getElementById('messageOutput');
    }
    let decodeImageData = ctx.getImageData(0, 0, width, height);
    console.log('imageData for decoing: ', decodeImageData.data)

    let divided = dividePixels(decodeImageData.data);
    let YCbCr = toYCbCrModel(divided); //RGB to YCbCr transform
    let merged = mergePixels(YCbCr);
    console.log('image in YCbCr: ', merged);
    let Ychannel = getRedChannel(merged); // getting Y from YCbCr
    let YchannelMatrix = toMatrix(Ychannel);
    let blockedImage = imageToBlock(YchannelMatrix);
    console.log('YCbCr image in blocks: ', blockedImage);
    let blockedImageWithDCT = useDCTtoBlocks(blockedImage);
    console.log('YCbCr with DCT: ', blockedImageWithDCT)
    let mess = [];
    let i1 = parseInt((N / 3).toString()), i2 = i1 + i1;
    for (let i = 0; i < blockedImageWithDCT.length; i++) {
        for (let j = 0; j < blockedImageWithDCT[i].length; j++) {
            if (Math.abs(blockedImageWithDCT[i][j][i2][i1]) >
Math.abs(blockedImageWithDCT[i][j][i1][i2])){
                mess.push('1');
            } else {
                mess.push('0');
            }
        }
    }
    output.innerText += convertBinaryToText(mess.join(''));
}
```

Візуальне порівняння оригінального зображення та зображення зі вбудованим повідомленням:

Оригінальне зображення

Зображення після вбудовування повідомлення

Повідомлення для приховування:

```
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.
```

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.
```

3. Реалізація стеганоатаки на основі використання алгоритму стискання JPEG та дослідження її можливостей

Для реалізації атаки використовується вбудована можливість JavaScript – стискання при завантаженні зображення.

Розмір оригінального зображення:	43,7 КБ (44 802 байт)
Розмір завантаженого зображення:	10,3 КБ (10 567 байт)

Отримані емпіричні результати відображені на рис. 3.1 - 3.2.

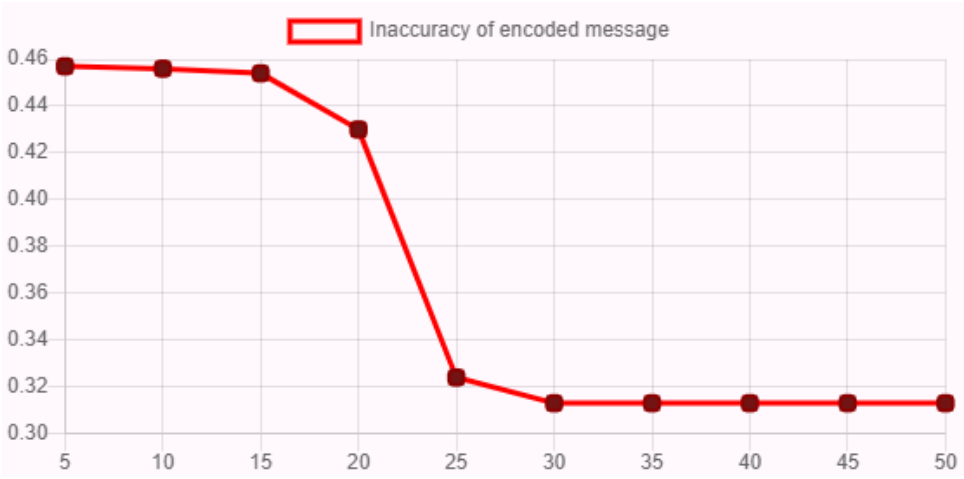

Рисунок 3.1 – Графік залежності помилкового вилучення інформаційних бітів до змін порогового значення *Pr*

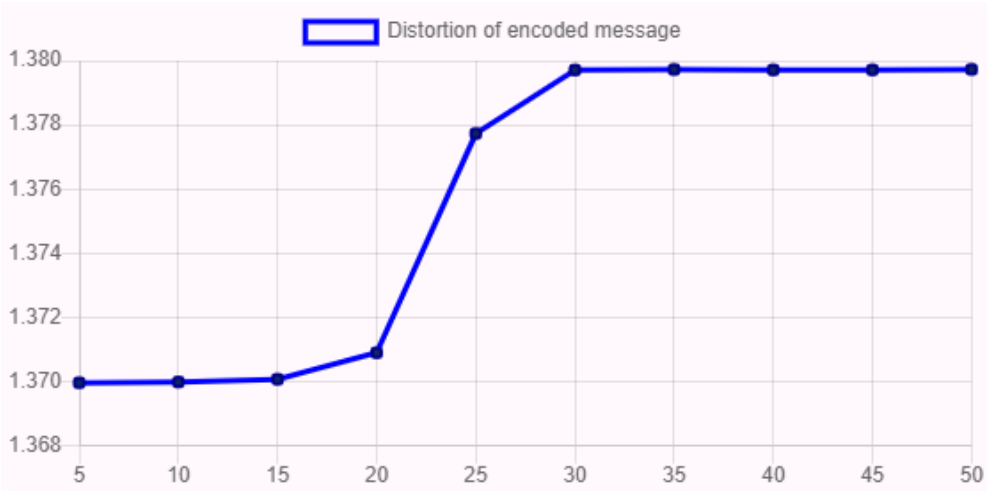

Рисунок 3.2 – Графік залежності внесених спотворень до змін порогового значення *Pr*

4. Реалізація вдосконалених алгоритмів вбудовування та вилучення повідомлень до частотної області зображень (метод Бенгама–Мемона–Ео–Юнга)

Функція вбудовування інформаційних бітів у зображення:

```
function encode() {
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input);
    let imageData = ctx.getImageData(0, 0, width, height);
    let divided = dividePixels(imageData.data);
    let YCbCr = toYCbCrModel(divided); //RGB to YCbCr transform
    let merged = mergePixels(YCbCr);
    let Ychannel = getRedChannel(merged); // getting Y from YCbCr
    let YchannelMatrix = toMatrix(Ychannel);
    let blockedImage = imageToBlock(YchannelMatrix);
    let blockedImageWithDCT = useDCTtoBlocks(blockedImage);
    let index = 0;
    for (let i = 0; i < Math.min(blockedImageWithDCT.length, input.length);
i++) {
        let i2 = parseInt((N / 3).toString()), i1 = i2 + i2, i3 = ((i1+i2)/2);
        for (let j = 0; j < blockedImageWithDCT[i].length; j++) {
            if (input[index] === '1') {
                if (blockedImageWithDCT[i][j][i3][i3] > 0) {
                    blockedImageWithDCT[i][j][i3][i3] =
                        Math.max(Math.abs(blockedImageWithDCT[i][j][i2][i1]),
                        Math.abs(blockedImageWithDCT[i][j][i1][i2])) + Pr;
                } else {
                    blockedImageWithDCT[i][j][i3][i3] = -
                        Math.max(Math.abs(blockedImageWithDCT[i][j][i2][i1]),
                        Math.abs(blockedImageWithDCT[i][j][i1][i2])) + Pr;
                }
                index++;
            } else {
                if (blockedImageWithDCT[i][j][i3][i3] > 0) {
                    blockedImageWithDCT[i][j][i3][i3] =
                        Math.min(Math.abs(blockedImageWithDCT[i][j][i2][i1]),
                        Math.abs(blockedImageWithDCT[i][j][i1][i2])) - Pr;
                } else {
                    blockedImageWithDCT[i][j][i3][i3] = -
                        Math.min(Math.abs(blockedImageWithDCT[i][j][i2][i1]),
                        Math.abs(blockedImageWithDCT[i][j][i1][i2])) - Pr;
                }
                index++;
            }
        }
    }
    let blockedImageWithIDCT = useIDCTtoBlocks(blockedImageWithDCT);
    let changedYchannel = flatBlocked(blockedImageWithIDCT);
    for (let i = 0, index = 0; i < merged.length; i++) {//write changes to
YCbCr arr
        if (i % 4 === 0) {
            merged[i] = changedYchannel[index];
            index++;
        }
    }
    let div = dividePixels(merged);
    let backToRGB = toRGB(div);
    let backToRGBarr = mergePixels(backToRGB);
    imageData.data.set(backToRGBarr);
    ctx.putImageData(imageData, 0, 0);
}
```

Функція вилучення інформаційних бітів із зображення:

```
let decode = function () {
    let tmp = document.getElementsByClassName('input-wrapper');
    let output;
    if (tmp[0].children.length <= 2) {
        output = document.createElement('textarea');
        output.id = 'messageOutput';
        tmp[0].appendChild(output);
    } else {
        output = document.getElementById('messageOutput');
    }
    let decodeImageData = ctx.getImageData(0, 0, width, height);
    let divided = dividePixels(decodeImageData.data);
    let YCbCr = toYCbCrModel(divided); //RGB to YCbCr transform
    let merged = mergePixels(YCbCr);
    let Ychannel = getRedChannel(merged); // getting Y from YCbCr
    let YchannelMatrix = toMatrix(Ychannel);
    let blockedImage = imageToBlock(YchannelMatrix);
    let blockedImageWithDCT = useDCTtoBlocks(blockedImage);
    let mess = [];
    let i1 = parseInt((N / 3).toString()), i2 = i1 + i1, i3 = ((i1 + i2)/2);
    for (let i = 0; i < blockedImageWithDCT.length; i++) {
        for (let j = 0; j < blockedImageWithDCT[i].length; j++) {
            if (blockedImageWithDCT[i][j][i3][i3] >
                        Math.max(Math.abs(blockedImageWithDCT[i][j][i2][i1]),
                                 Math.abs(blockedImageWithDCT[i][j][i1][i2]))){
                mess.push('1');
            } else {
                mess.push('0');
            }
        }
    }
    output.innerText += convertBinaryToText(mess.join(''));
}
```

Візуальне порівняння оригінального зображення та зображення зі вбудованим повідомленням:

Оригінальне зображення

Зображення після вбудовування повідомлення

Повідомлення для приховування:

```
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.
```

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.
```

Висновки

При виконанні лабораторної роботи було реалізовано стеганографічні методи приховування даних у частотній області нерухомих зображень на основі кодування різниці абсолютних значень коефіцієнтів дискретно-косинусного перетворення. Було розроблено алгоритми приховування та вилучення даних із просторової області нерухомих зображень шляхом аналізу значень середніх частот спектру блоку. Проведено емпіричні дослідження щодо візуального виявлення частки спотворень, шо вносяться, залежно від модифікованої частоти. Проведено атаку стиснення на зображення, що містить вбудоване повідомлення у частотній області, та подальший аналіз залежності правильного вилучення інформаційних бітів та частки спотворень від величини порогового значення.