Министерство образования Республики Беларусь
Учреждение Образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра электронных вычислительных средств

Лабораторная работа № 5
«Сжатие изображений»

Проверил:
Рыбенков Е.В.

Выполнил:
ст. гр. 850701
Филипцов Д. А.

Минск 2021

1 Цель работы

Сжатия изображения на примере стандарта JPEG.

2 Задание

Выполнить анализ изображения с помощью ДКП из ЛР4 и получить 64 канальных сигнала. Выполнить квантование канальных сигналов. Выполнить синтез квантованного изображения с помощью ДКП из ЛР4. Оценить качество квантованного изображения.

Преобразовать квантованный канальный сигнал в вектор. Написать собственную функцию RLE. Символы и число их повторений хранить в разных массивах. Выполнить кодирование всех каналов (RLE-символов и RLE-повторов) раздельно с помощью алгоритма Хаффмана. Рассчитать минимально требуемое число бит информации для восстановления изображения (код и словарь Хаффмана).

3 Ход работы

MATLAB-код:

```
%% Квантование коэффициентов ДКП
J = rgb2gray(im2double(imread('kodim03.png')));
I = J(1:256, 1:256);
bi = 4;
for i = 1:32
    for j = 1:32
        I((i-1)*8+1:i*8, (j-1)*8+1:j*8) = dct2(I((i-
1)*8+1:i*8, (j-1)*8+1:j*8));
    end
end
J = zeros(size(I, 1), size(I, 2));
for i = 1:8
    for j = 1:8
        for ii = 1:32
            for jj = 1:32
                J((i-1)*32+ii, (j-1)*32+jj) = I((ii-1)*8+i,
(jj-1)*8+j);
            end
        end
    end
end
sigma = zeros(8, 8);
```

```
for i = 1:8
    for j = 1:8
        w = J((i-1)*32+1 : i*32, (j-1)*32+1 : j*32);
        sigma(i, j) = var(reshape(w', [1 numel(w)]));
    end
end
P = prod(nthroot(reshape(sigma', [1 numel(sigma)]), 64));
b = zeros(8, 8);
for i = 1:8
    for j = 1:8
        b(i, j) = bi + log2(sigma(i, j)/P)/2;
    end
end
b = round(b);
b(b < 0) = 0;
h = 2.^b;
for i = 1:32
    for j = 1:32
        for ii = 1:8
            for jj = 1:8
                I((i-1)*8+ii, (j-1)*8+jj) = round(J((ii-
1)*32+i, (jj-1)*32+j) * h(ii, jj))/h(ii, jj);
            end
        end
    end
end
for i = 1:32
    for j = 1:32
        I((i-1)*8+1:i*8, (j-1)*8+1:j*8) = idct2(I((i-
1)*8+1:i*8, (j-1)*8+1:j*8));
    end
end
J = rgb2gray(im2double(imread('kodim03.png')));
A = psnr(I, J(1:256, 1:256));
B_array(bi) = A;
%% Кодирование коэффициентов ДКП
J = rgb2gray(im2double(imread('kodim03.png')));
I = J(1:256, 1:256);
bi = 4;
for i = 1:32
    for j = 1:32
        I((i-1)*8+1:i*8, (j-1)*8+1:j*8) = dct2(I((i-
1)*8+1:i*8, (j-1)*8+1:j*8));
    end
end
J = zeros(size(I, 1), size(I, 2));
```

```
for i = 1:8
    for j = 1:8
        for ii = 1:32
            for jj = 1:32
                J((i-1)*32+ii, (j-1)*32+jj) = I((ii-1)*8+i,
(jj-1)*8+j);
            end
        end
    end
end
sigma = zeros(8, 8);
for i = 1:8
    for j = 1:8
        w = J((i-1)*32+1 : i*32, (j-1)*32+1 : j*32);
        sigma(i, j) = var(reshape(w', [1 numel(w)]));
    end
end
P = prod(nthroot(reshape(sigma', [1 numel(sigma)]), 64));
b = zeros(8, 8);
for i = 1:8
    for j = 1:8
        b(i, j) = bi + log2(sigma(i, j)/P)/2;
    end
end
b = round(b);
b(b < 0) = 0;
h = 2.^b;
for i = 1:32
    for j = 1:32
        for ii = 1:8
            for jj = 1:8
                I((ii-1)*32+i, (jj-1)*32+j) = round(J((ii-
1)*32+i, (jj-1)*32+j) * h(ii, jj))/h(ii, jj);
            end
        end
    end
end
min_numb_bits = 0;
for i = 1:8
    for j = 1:8
        channel = I((i-1)*32+1:i*32,(j-1)*32+1:j*32);
        vector = zigzag(channel);
        [symbols, rep] = RLE(vector);
        prob_symb = symb_prob(vector);
        min_numb_bits = min_numb_bits + Hofmannita(symbols,
prob_symb);
```

```
        prob_rpt = symb_prob(rep);
        min_numb_bits = min_numb_bits + Hofmannita(rep,
prob_rpt);
    end
end
psnr_array(bi) = min_numb_bits;

function prob = symb_prob(data)

symb = unique(data);
prob = zeros(1, length(symb));
N = length(data);

for i = 1:length(symb)
   prob(i) = length(data(data == symb(i)))/N;
end

end

function bits_numb = Hofmannita(RLE_vector, vector_prob)

bits_numb = 0;

if RLE_vector == RLE_vector(1)
   bits_numb = 1 + 1 + length(RLE_vector);
else
    RLE_vector_u = unique(RLE_vector);
    RLE_dict = huffmandict(RLE_vector_u, vector_prob);
    RLE_code = huffmanenco(RLE_vector, RLE_dict);

    bits_numb = length(RLE_code);
    bits = ceil(log2(length(RLE_dict)));
    bits_numb = bits_numb + bits*length(RLE_dict);

    for k = 1:length(RLE_dict)
       bits_numb =  bits_numb + length(RLE_dict{k,2});
    end
end

end

function [symbols, rep] = RLE(x)

counter = 0;
number = x(1,1);
k = 1;
symbols(k) = number;
```

```
for i = 1:size(x,1)
    for j = 1:size(x,2)
        if (x(i,j) ~= number)
            counter = 0;
            number = x(i,j);
            symbols(k+1) = number;
            k = k + 1;
        end

        if (x(i,j) == number)
            counter = counter + 1;
            rep(k) = counter;
        end
    end
end

end
```

Выполнение:

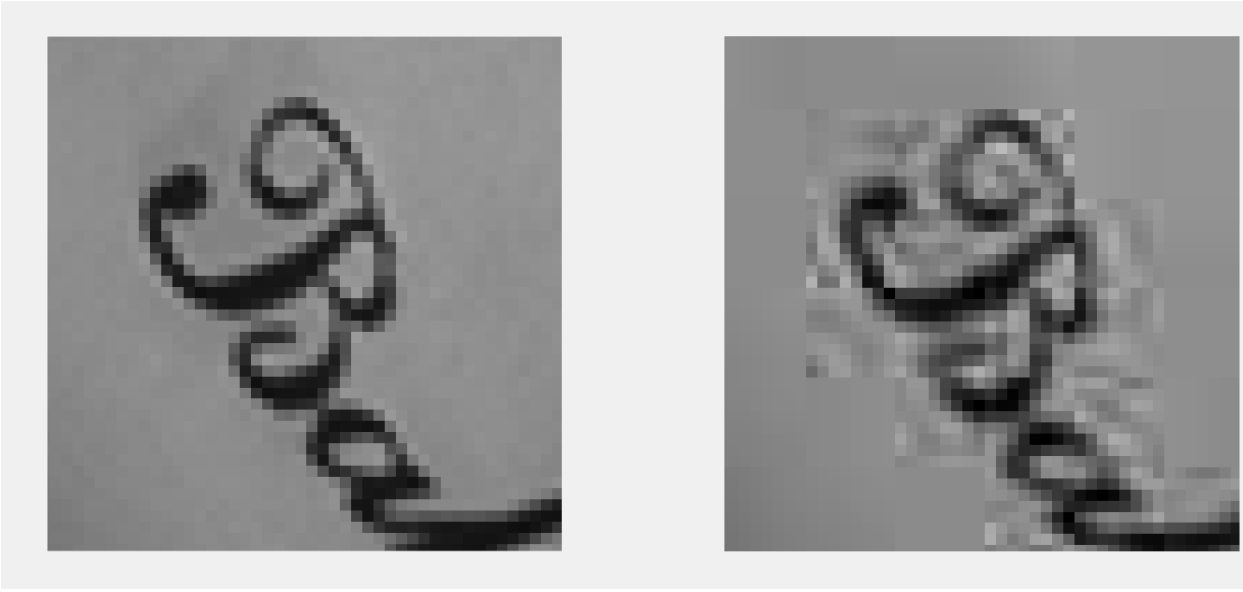

Рисунок 3.1 – Исходный и восстановленный после квантования при b = 1 фрагмент изображения

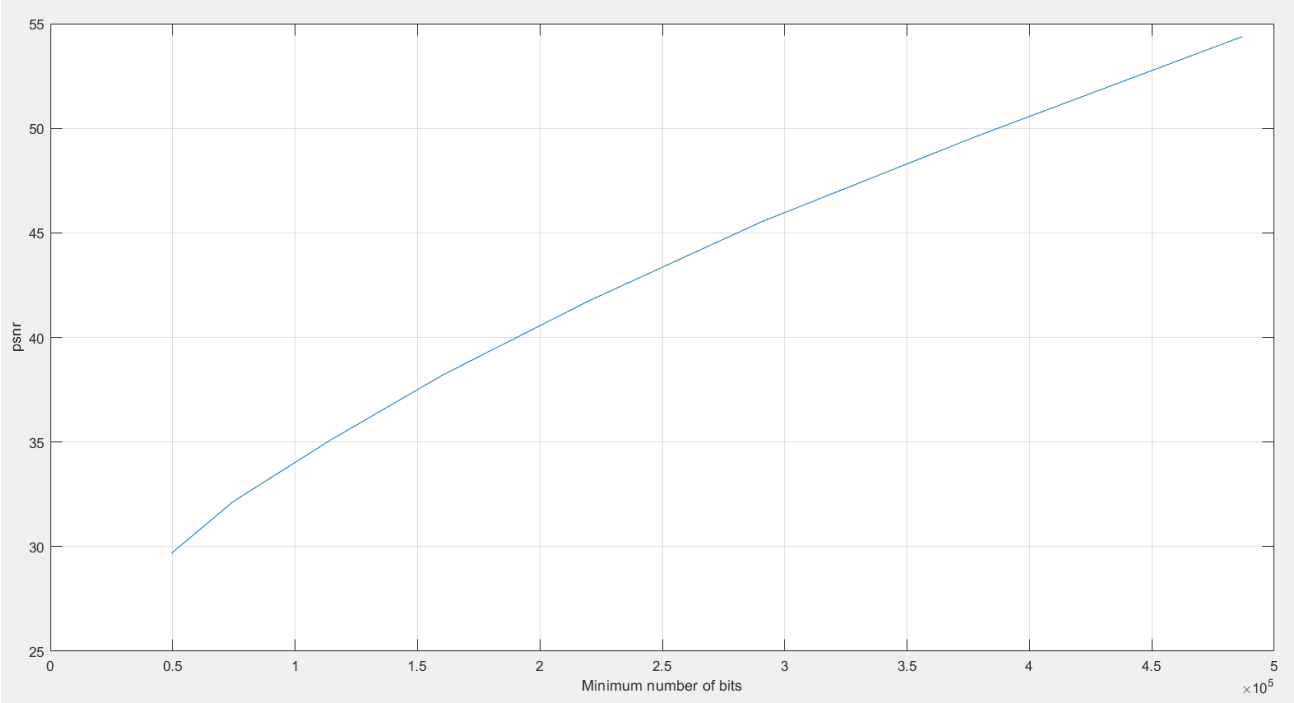

Рисунок 3.2 – Зависимость качества изображения от минимально требуемого числа бит информации

4 Вывод

В ходе лабораторной работы мы научились сжимать изображения на примере стандарта JPEG.