МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Университет ИТМО»

УТВЕРЖДАЮ .

Доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. Е. Бочарова

«\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 г.

ОТЧЕТ  
О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по курсу «Мультимедиа технологии»

по теме:

Стандарты сжатия изображений с потерей качества.

Стандарт JPEG

Студент гр. 3511 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Трофимов В.А.

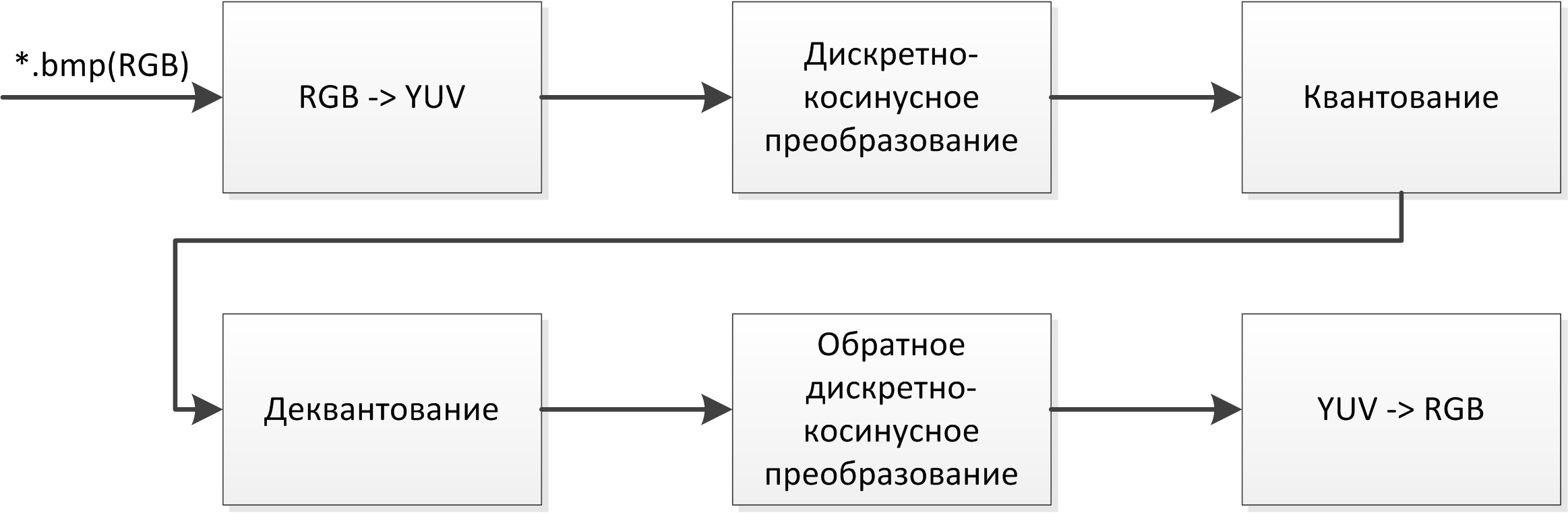
Студент гр. 3511 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шобей А.В.

Санкт-Петербург

2014

**Вариант 4**

# Схема работы кодера



# Оценка коэффициента сжатия

Оценивание коэффициента сжатия для компоненты происходит по формуле:

где – объем информации, необходимый для хранения сжатой компоненты,

, – энтропия потоков разностей коэффициентов постоянного тока,

, – энтропия потока длин серий нулей, – число серий нулей,

, – энтропия потока ненулевых коэффициентов, – число ненулевых коэффициентов,

, – энтропия потока числа ненулевых коэффициентов в блоках,

M, N – ширина и высота изображения.

Оценивание коэффициента сжатия для всего изображения (всех трех компонент) происходит по формуле:

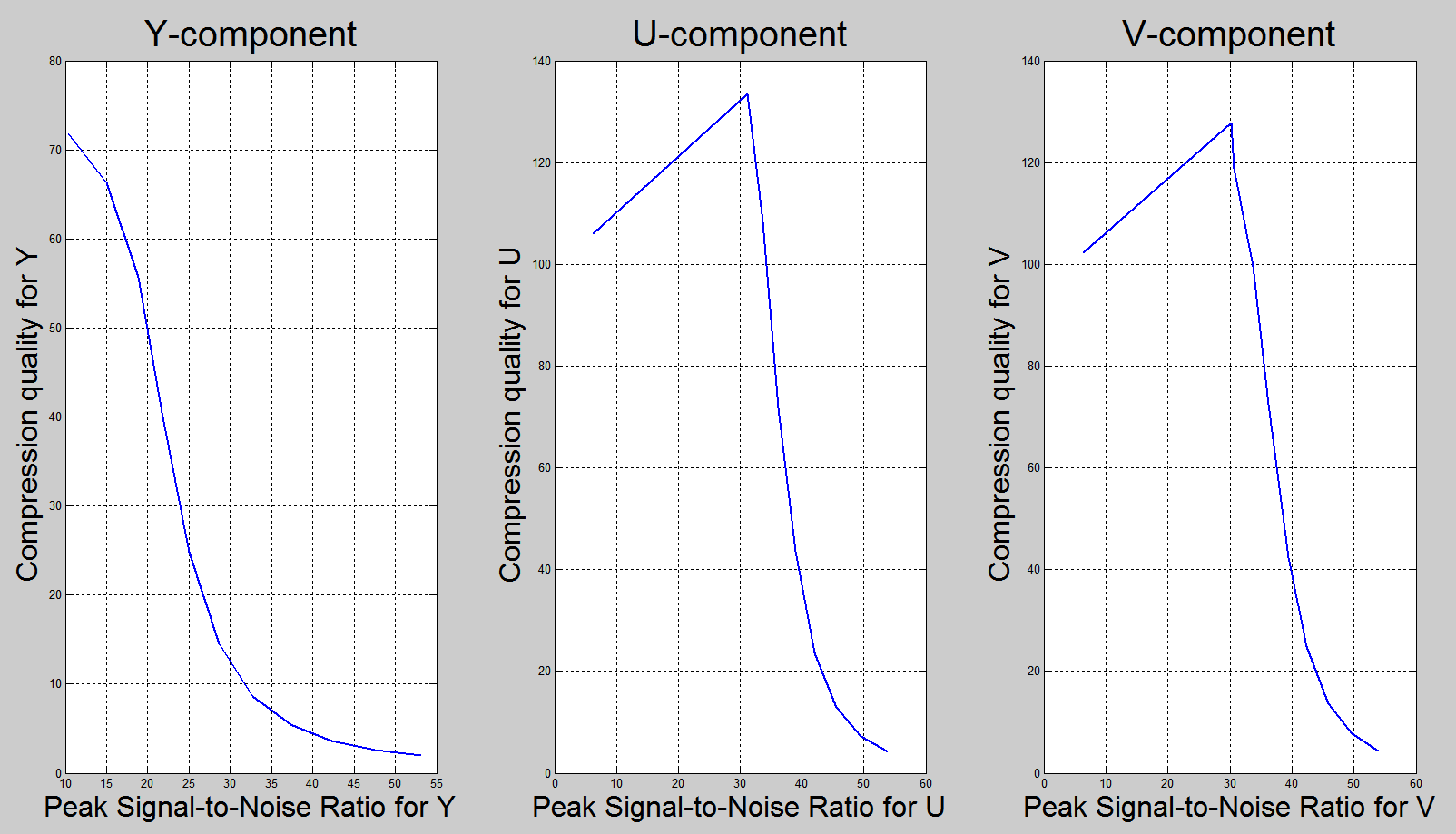
где – объем информации, необходимый для хранения сжатых компонент соответственно.

# Отношение сигнал/шум

Отношение сигнал/шум вычисляется по следующей формуле:

Где – переменная компоненты до ДКП, - переменная компоненты после применения ДКП и обратного ДКП, M, N – ширина и высота изображения.

# График зависимости степени сжатия от соотношения сигнал/шум при переборе StepA от 2 до 2048

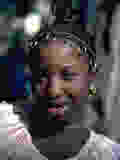


# Пример сжатых изображений



Сжатие 3.08

StepA = 2



Сжатие 50.39

StepA = 128



Сжатие 73.27

StepA = 256



Сжатие 98.69

StepA = 1024

# Программа

function lab3\_04\_trofiv\_shobey()

clc;

inputFileName = 'input.bmp';

outputFileName = 'restored';

initialStepA = 0;

step = 2;

repeats = 11;

PSNRY = zeros(1, repeats);

PSNRU = zeros(1, repeats);

PSNRV = zeros(1, repeats);

compresionY = zeros(1, repeats);

compresionU = zeros(1, repeats);

compresionV = zeros(1, repeats);

for i = 1 : repeats

stepA = initialStepA + step ^ i;

[PeakSNRY, PeakSNRU, PeakSNRV, comprY, comprU, comprV, totalCompr] = process(inputFileName, outputFileName, stepA);

fprintf('StepA: %.2f ', stepA);

fprintf('PeakSNRY: %.2f comprY: %.2f ', PeakSNRY, comprY);

fprintf('PeakSNRU: %.2f comprU: %.2f ', PeakSNRU, comprU);

fprintf('PeakSNRV: %.2f comprV: %.2f ', PeakSNRV, comprV);

fprintf('totalCompr: %.2f\n', totalCompr);

PSNRY(i) = PeakSNRY;

PSNRU(i) = PeakSNRU;

PSNRV(i) = PeakSNRV;

compresionY(i) = comprY;

compresionU(i) = comprU;

compresionV(i) = comprV;

end

figure;

subplot(1, 3, 1);

plot(PSNRY, compresionY, 'LineWidth', 2);

title('Y-component', 'FontSize', 30);

xlabel('Peak Signal-to-Noise Ratio for Y', 'FontSize', 24);

ylabel('Compression quality for Y', 'FontSize', 24);

grid on;

subplot(1, 3, 2);

plot(PSNRU, compresionU, 'LineWidth', 2);

title('U-component', 'FontSize', 30);

xlabel('Peak Signal-to-Noise Ratio for U', 'FontSize', 24);

ylabel('Compression quality for U', 'FontSize', 24);

grid on;

subplot(1, 3, 3);

plot(PSNRV, compresionV, 'LineWidth', 2);

title('V-component', 'FontSize', 30);

xlabel('Peak Signal-to-Noise Ratio for V', 'FontSize', 24);

ylabel('Compression quality for V', 'FontSize', 24);

grid on;

end

function [PeakSNRY, PeakSNRU, PeakSNRV, comprY, comprU, comprV, totalCompr] = process(inputFileName, outputFileName, stepAC)

[header, w, h, bitmap] = readBitmap(inputFileName);

[R, G, B] = bitmap2RGB(bitmap, w, h);

[Y, U, V] = RGB2YUV(R, G, B);

[Y, PeakSNRY, sizeY] = encode(Y, stepAC, w, h);

[U, PeakSNRU, sizeU] = encode(U, stepAC, w, h);

[V, PeakSNRV, sizeV] = encode(V, stepAC, w, h);

comprY = 8 \* w \* h / sizeY;

comprU = 8 \* w \* h / sizeU;

comprV = 8 \* w \* h / sizeV;

totalCompr = 24 \* w \* h / (sizeY + sizeU + sizeV);

[R, G, B] = YUV2RGB(Y, U, V);

outputFileName = strcat(outputFileName, num2str(stepAC), '.bmp');

bitmap = RGB2bitmap(R, G, B, w, h);

writeBitmap(outputFileName, header, bitmap);

end

function [compressedSignal, PeakSNR, size] = encode(signal, stepAC, w, h)

DC = [];

stepDC = 8;

zeroSeries = [];

nonZeroSeries = [];

nonZeroSeriesCountInBlock = [];

compressedSignal = [w, h];

for i = 1 : w / 8

for j = 1 : h / 8

startIIndex = (i - 1) \* 8 + 1;

endIIndex = (i - 1) \* 8 + 8;

startJIndex = (j - 1) \* 8 + 1;

endJIndex = (j - 1) \* 8 + 8;

currentBlock = signal(startIIndex : endIIndex, startJIndex : endJIndex);

currentBlock = dct2(currentBlock);

DC = [DC round((currentBlock(1, 1) / stepDC)) \* stepDC]; %#ok<\*AGROW>

currentBlock = round(currentBlock ./ stepAC);

AC = zigzag(currentBlock);

zeroCount = 0;

nonZeroCount = 0;

for k = 1 : length(AC)

if (AC(k) == 0)

zeroCount = zeroCount + 1;

else

nonZeroSeries = [nonZeroSeries AC(k)];

zeroSeries = [zeroSeries zeroCount];

zeroCount = 0;

nonZeroCount = nonZeroCount + 1;

end

end

nonZeroSeriesCountInBlock = [nonZeroSeriesCountInBlock nonZeroCount];

currentBlock = currentBlock .\* stepAC;

currentBlock = idct2(currentBlock);

compressedSignal(startIIndex : endIIndex, startJIndex : endJIndex) = currentBlock;

end

end

DC = DC(2 : length(DC)) - DC(1 : length(DC) - 1);

bDC = enthropy(DC) \* h \* w / 64;

bRunlen = enthropy(zeroSeries) \* length(zeroSeries);

bCoeff = enthropy(nonZeroSeries) \* length(nonZeroSeries);

bNum = enthropy(nonZeroSeriesCountInBlock) \* (h \* w / 64);

size = bDC + bRunlen + bCoeff + bNum;

signalDiff = signal - compressedSignal;

PeakSNR = 10 \* log10(65025 / sum(sum(signalDiff .\* signalDiff)) \* w \* h);

end

function [bitmap] = RGB2bitmap(R, G, B, w, h)

bitmap = zeros(w\*3, h);

for i = 1 : w

for j = 1 : h

bitmap((i - 1) \* 3 + 1, j) = B(i, j);

bitmap((i - 1) \* 3 + 2, j) = G(i, j);

bitmap((i - 1) \* 3 + 3, j) = R(i, j);

end

end

end

function [R, G, B] = YUV2RGB(Y, U, V)

G = Y - 0.714 \* (V - 128) - 0.334 \* (U - 128);

R = Y + 1.402 \* (V - 128);

B = Y + 1.772 \* (U - 128);

end

function [Y, U, V] = RGB2YUV(R, G, B)

Y = 0.299 \* R + 0.587 \* G + 0.114 \* B;

U = (B - Y) \* 0.5643 + 128;

V = (R - Y) \* 0.7132 + 128;

end

function [R, G, B] = bitmap2RGB(bitmap, w, h)

R = zeros(w, h);

G = zeros(w, h);

B = zeros(w, h);

for i = 1 : w

for j = 1 : h

B(i, j) = bitmap((i - 1) \* 3 + 1, j);

G(i, j) = bitmap((i - 1) \* 3 + 2, j);

R(i, j) = bitmap((i - 1) \* 3 + 3, j);

end

end

end

function [result] = zigzag(x)

result = [ x(1, 1), x(1, 2), x(2, 1), x(3, 1), x(2, 2), x(1, 3), x(1, 4), x(2, 3), ...

x(3, 2), x(4, 1), x(5, 1), x(4, 2), x(3, 3), x(2, 4), x(1, 5), x(1, 6), ...

x(2, 5), x(3, 4), x(4, 3), x(5, 2), x(6, 1), x(7, 1), x(6, 2), x(5, 3), ...

x(4, 4), x(3, 5), x(2, 6), x(1, 7), x(1, 8), x(2, 7), x(3, 6), x(4, 5), ...

x(5, 4), x(6, 3), x(7, 2), x(8, 1), x(8, 2), x(7, 3), x(6, 4), x(5, 5), ...

x(4, 6), x(3, 7), x(2, 8), x(3, 8), x(4, 7), x(5, 6), x(6, 5), x(7, 4), ...

x(8, 3), x(8, 4), x(7, 5), x(6, 6), x(5, 7), x(4, 8), x(5, 8), x(6, 7), ...

x(7, 6), x(8, 5), x(8, 6), x(7, 7), x(6, 8), x(7, 8), x(8, 7), x(8, 8) ];

end

function [result] = enthropy(x)

maxx = max(x);

minx = min(x);

result = 0;

count = zeros(maxx - minx + 1);

for i = 1 : length(x)

pos = x(i) - minx + 1;

count(pos) = count(pos) + 1;

end

for i = 1 : length(count)

if (count(i) ~= 0)

probability = count(i) / length(x);

result = result - (probability \* log2(probability));

end

end

end

function [header, w, h, bitmap] = readBitmap(inputFileName)

input = fopen(inputFileName, 'rb');

header = fread(input, 54, 'uint8');

w = header(19);

h = header(23);

bitmap = fread(input,[w \* 3, h], 'uint8');

fclose(input);

end

function [] = writeBitmap(outputFileName, header, bitmap)

output = fopen(outputFileName, 'wb');

fwrite(output, header, 'uint8');

fwrite(output, bitmap, 'uint8');

fclose(output);

end

# Вывод программы

StepA: 2.00 PeakSNRY: 53.17 comprY: 1.94 PeakSNRU: 53.87 comprU: 4.27 PeakSNRV: 53.91 comprV: 4.45 totalCompr: 3.08

StepA: 4.00 PeakSNRY: 47.58 comprY: 2.58 PeakSNRU: 49.55 comprU: 7.26 PeakSNRV: 49.64 comprV: 7.73 totalCompr: 4.58

StepA: 8.00 PeakSNRY: 42.34 comprY: 3.63 PeakSNRU: 45.49 comprU: 12.86 PeakSNRV: 45.87 comprV: 13.68 totalCompr: 7.04

StepA: 16.00 PeakSNRY: 37.39 comprY: 5.41 PeakSNRU: 41.99 comprU: 23.67 PeakSNRV: 42.42 comprV: 24.87 totalCompr: 11.22

StepA: 32.00 PeakSNRY: 32.71 comprY: 8.59 PeakSNRU: 39.01 comprU: 43.59 PeakSNRV: 39.40 comprV: 42.31 totalCompr: 18.41

StepA: 64.00 PeakSNRY: 28.65 comprY: 14.52 PeakSNRU: 36.13 comprU: 71.76 PeakSNRV: 36.21 comprV: 72.43 totalCompr: 31.06

StepA: 128.00 PeakSNRY: 24.97 comprY: 24.83 PeakSNRU: 33.64 comprU: 108.39 PeakSNRV: 33.72 comprV: 99.62 totalCompr: 50.39

StepA: 256.00 PeakSNRY: 21.75 comprY: 40.49 PeakSNRU: 31.78 comprU: 127.60 PeakSNRV: 30.72 comprV: 118.93 totalCompr: 73.27

StepA: 512.00 PeakSNRY: 18.82 comprY: 55.69 PeakSNRU: 31.12 comprU: 133.51 PeakSNRV: 30.16 comprV: 127.83 totalCompr: 90.17

StepA: 1024.00 PeakSNRY: 15.00 comprY: 66.29 PeakSNRU: 31.12 comprU: 133.51 PeakSNRV: 30.16 comprV: 127.83 totalCompr: 98.69

StepA: 2048.00 PeakSNRY: 10.42 comprY: 71.83 PeakSNRU: 6.29 comprU: 106.10 PeakSNRV: 6.35 comprV: 102.34 totalCompr: 90.58

# Вывод

В ходе работы было выявлено, что для заданного изображения возможно сжатие примерно в 31 раз, с удовлетворительным качеством (соотношение сигнал/шум при данном сжатии составляет 28.65). При большем сжатии наблюдается заметное цветное искажение, которое, при дальнейшем сжатии, приводит к полной неразличимости исходного изображения.