Санкт-Петербургский национальный исследовательский университет информационных технологий механики и оптики

Цифровая обработка сигналов

**Лабораторная работа №5**

**Студент:**

Черезов Игорь Юрьевич

**Группа:**

P3400

**Преподаватель:**

Тропченко Андрей Александрович

Санкт-Петербург

2020

# Цель работы

* Знакомство с алгоритмом JPEG в системе Matlab;
* Исследование эффективности JPEG-сжатия;
* Получение навыков реализации линейных ортогональных преобразований.

# Задание

1. Создайте функцию *compressRatio*, определяющую степень сжатия изображения;
2. Создайте функцию *directDCT*, осуществляющую прямое ДКП изображения по формуле 1;
3. Создайте функцию *invertDCT,* осуществляющую обратное ДКП изображения по формуле 5;
4. Используя полученное для данного задания изображение и предыдущий пример осуществите 12 итераций, на каждой из которых будет происходить постепенное обнуление коэффициентов ДКП в порядке, обратном зиг-загообразному. (т.е., начиная с высокочастотного). DC-коэффициент обнулять не нужно.  
   На каждой итерации а) при помощи функции compressRatio вычислите уровень сжатия полученного изображения, б) при помощи функции corr2 вычислите корреляцию Пирсона между исходным и сжатым изображением.
5. При помощи функции plot постройте графики зависимости а) между количеством задействованных коэффициентов и качеством изображения, б) между уровнем сжатия и качеством изображения. Оси графиков должны быть подписаны.
6. Используя график “а)” определите приблизительное положение высокочастотных, среднечастотных и низкочастотных коэффициентов ДКП. Используя график “б)” определите зависимость между уровнем сжатия и качеством полученного изображения.
7. Анализируя внешний вид изображений, полученных при высоких коэффициентах сжатия, определите главный недостаток формата JPEG.

# Выполнение

## Задание 1 CompressRatio

Код

% This function takes filepaths of a compressed and an uncompressed

%images and returns compression ratio equal to their size relation

function cr = compressRatio(compressed\_img\_path, og\_img\_path)

og\_size = imfinfo(og\_img\_path).FileSize;

compressed\_size = imfinfo(compressed\_img\_path).FileSize;

cr = og\_size/compressed\_size;

endfunction

## Задание 2 directDCT

Было написано несколько версий функции directDCT, однако дальнейшая работа с ними показала, что наиболее эффективным по времени способом оказался строчно-столбцовый метод. Для этого была написана функция, возвращающая матрицу преобразования ДКП, которая затем была использована в основной функции direct\_dct.

function mtx = dct\_mtx(n)

function val = c(i)

if i == 1

val = sqrt(1/n);

else

val = sqrt(2/n);

endif

endfunction

for k=1:n

for l=1:n

mtx(k,l) = c(k)\*cos((k-1)\*(l-0.5)\*pi/n);

endfor

endfor

endfunction

function dct = direct\_dct(block)

n = size(block)(1,1);

coef = dct\_mtx(n);

dct = coef\*block\*coef';

endfunction

## Задание 3 invertDCT

Аналогично прямому преобразованию, матрично-столбцовый метод оказался наиболее эффективным. Была использована та же функция для получения матрицы преобразования ДКП, которая была использована в основной функции.

function mtx = invert\_dct(block)

n = size(block)(1,1);

coef = dct\_mtx(n);

mtx = coef'\*block\*coef;

endfunction

## Задание 4

При автоматизации данного задания наибольшую трудность представило написание функции, заполняющей маску в зигзагообразном порядке. Далее простым циклом было сжато и разжато изображение по всем возможным маскам.

function [cmp, quality, cmp\_ratio, cor] = compress(img\_path, mask)

img = imread(img\_path);

img = rgb2gray(img);

%img = im2double(img);

%figure, imshow(img);

img = cast(img, "double");

cmp = blockproc(img, [8 8], @direct\_dct);

figure, imshow(cmp);

cmp = blockproc(cmp, [8 8], @quant);

cmp = blockproc(cmp, [8 8], @apply\_mask, mask);

cmp = blockproc(cmp, [8 8], @dequant);

figure, imshow(cmp);

cmp = blockproc(cmp, [8 8], @invert\_dct);

cmp = cast(cmp, "uint8");

figure, imshow(cmp);

%compute quality difference by calculating standart deviation

%from original image

path = "temp.jpg";

quality = size(img)(1,1)^2/sum(sum((img - cmp).^2));

imwrite(cmp, ["cmp" path]);

img = cast(img, "uint8");

imwrite(img, ["ucmp" path]);

cmp\_ratio = compressRatio(["cmp" path], ["ucmp" path]);

cor = corr2(cmp, img);

endfunction

## Задание 5

Функция plot\_all использует данные, собранные функцией compress, для построения графиков зависимости качества изображения от количества оставленных коэффициентов и от уровня сжатия.

function plot\_all(n)

coefs = zeros(n)(1, :);

coefs(1) = 1;

for i=1:n

coefs(i) = round(i\*64/n);

endfor

quality = zeros(n)(1, :);

cmp\_ratio = zeros(n)(1, :);

cor = zeros(n)(1, :);

path = "batman\_jl.png"

for i = 1:n

disp(ctime(time));

disp(i);

[cmp, quality(i), cmp\_ratio(i), cor(i)] = compress(path, make\_mask(coefs(i)));

endfor

figure, plot(coefs, quality, 'marker', 'x');

xlabel("Number of coefficients");

ylabel("Quality");

figure, plot(cmp\_ratio, quality, 'marker', 'x');

xlabel("Compression ratio");

ylabel("Quality");

endfunction

## Задание 6

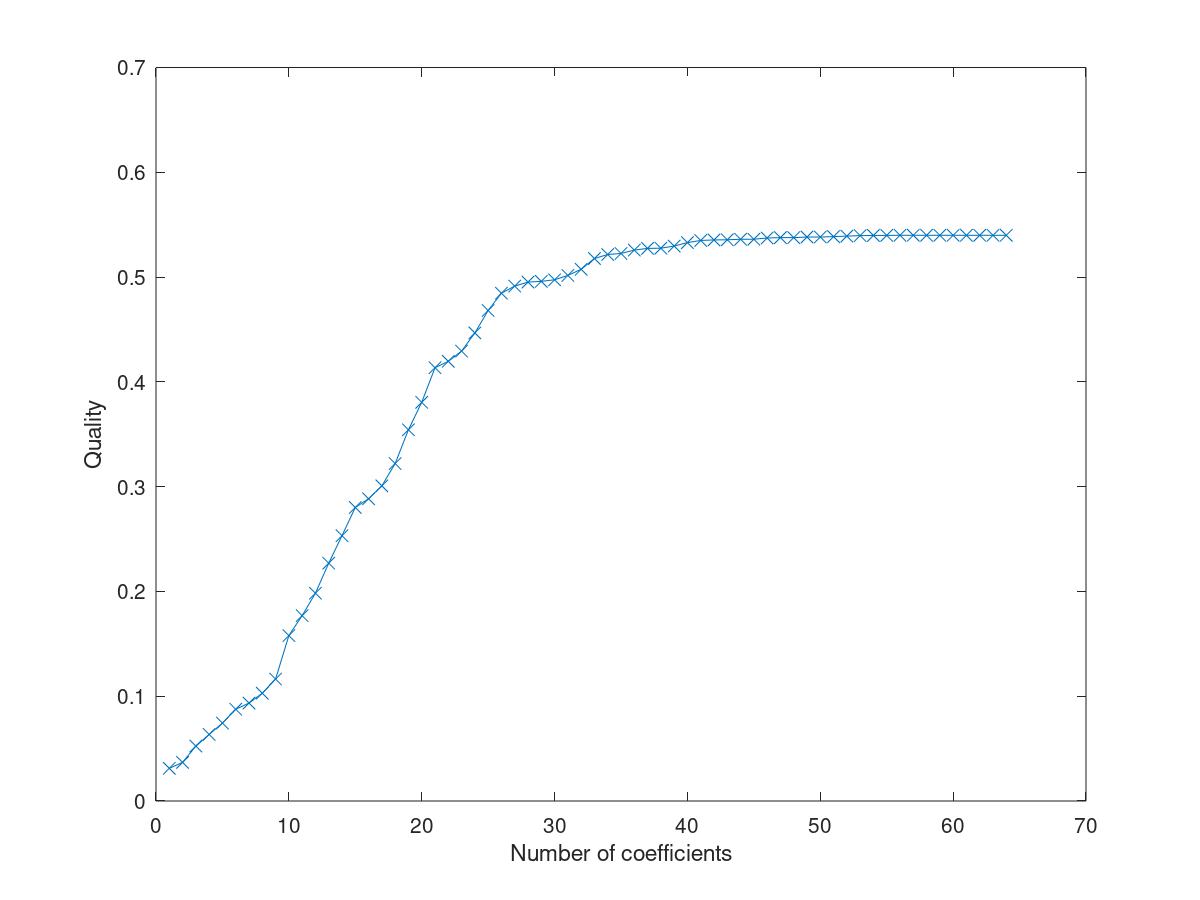


График График зависимости качества изображения от количества коэфициентов

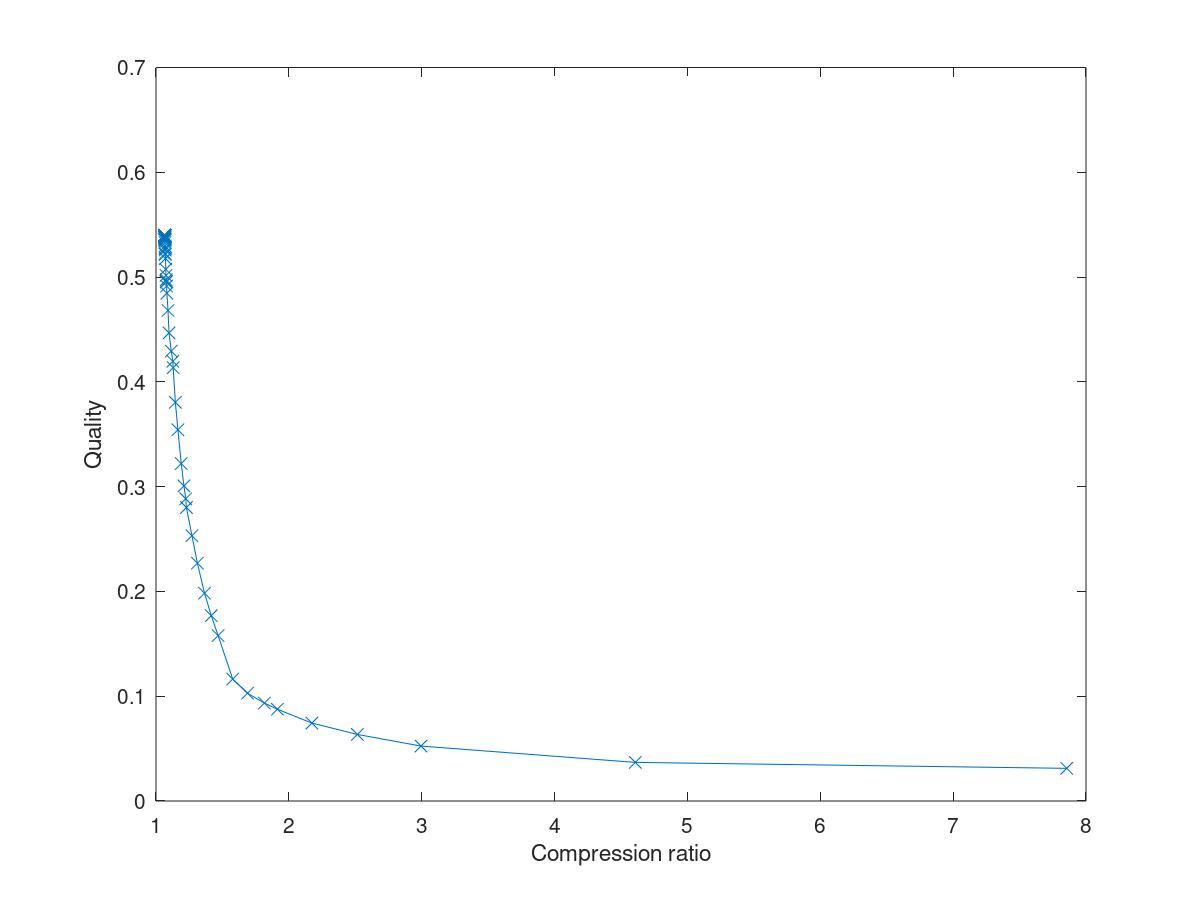


График График зависимости качества изображения от степени сжатия

Из графика 1 видно, что представленная функция ведёт себя различным образом при различном количестве коэффициентов. При количестве коэффициентов большем 34 график практически не изменяется, что говорит от том, что последние 30 коэффициентов являются высокочастотными. При количестве коэффициентов от 11 до 34 растёт значительно быстрее, чем от 1 до 10. Можно предположить, что 1-10 коэффициенты низкочастотные, а 11-34 – среднечастотные.

Из графика 2 видно, что высокая степень сжатия достижима лишь при сильной потере качества.

## Задание 7



Изображение Исходное изображение



Изображение Средняя степень сжатия



Изображение Высокая степень сжатия

При высоких степенях сжатия все пиксели в блоке имеют один цвет, что равносильно уменьшению размера изображения по количеству пикселей в 64 раза. При этом JPEG не позволяет достичь такого результата в результате работы своего алгоритма.

# Выводы

В ходе выполнения данной лабораторной работы был изучен принцип сжатия алгоритма JPEG. Был реализован алгоритм сжатия и сопутствующие ему прямое и обратное дискретные косинусные преобразования. Была визуализирована и анализирована зависимость качества изображения от количества оставленных коэффициентов ДКП и от степени сжатия изображения.

На основании полученных данных был сделан вывод о том, что алгоритм сжатия JPEG эффективен при низких степенях сжатия, так как качество изображения незначительно ухудшается при уменьшении количества ДКП коэффициентов в два раза, после чего начинает стремительно падать.