**4ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники

Лабораторная работа №5

по дисциплине «Цифровая обработка сигналов»

Выполнил:Ларочкин Г.И

Группа: P3400

Преподаватель: Тропченко А.А.

Санкт-Петербург

2020 г.

## Постановка задачи

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.

## Решение задачи

## Реализация directDCT и inverseDCT

Были реализованы две функции im2jpeg (directDCT) и jpeg2im (inverseDCT).



Рис.1 - исходное изображение



Рис. 2 – сжатое изображение

Part1.m:

f = imread('resources/gray.jpg');

figure, imshow(f)

g = im2jpeg(f);

f = jpeg2im(g);

figure, imshow(f)

im2jpeg.m:

function y = im2jpeg(x, mask)

if nargin < 2

mask = ones(8);

end

m = [16 11 10 16 24 40 51 61

12 12 14 19 26 58 60 55

14 13 16 24 40 57 69 56

14 17 22 29 51 87 80 62

18 22 37 56 68 109 103 77

24 35 55 64 81 104 113 92

49 64 78 87 103 121 120 101

72 92 95 98 112 100 103 99];

order = [1 9 2 3 10 17 25 18 11 4 5 12 19 26 33 ...

41 34 27 20 13 6 7 14 21 28 35 42 49 57 50 ...

43 36 29 22 15 8 16 23 30 37 44 51 58 59 52 ...

45 38 31 24 32 39 46 53 60 61 54 47 40 48 55 ...

62 63 56 64];

[xm, xn] = size(x);

x = double(x) - 128;

t = dctmtx(8);

y = blkproc(x, [8 8], 'P1 \* x \* P2', t, t');

y = blkproc(y, [8 8], 'round(x ./ P1)', m);

y = blkproc(y, [8 8], 'x .\* P1', mask);

y = im2col(y, [8 8], 'distinct');

xb = size(y, 2);

y = y(order, :);

eob = max(y(:)) + 1;

r = zeros(numel(y) + size(y, 2), 1);

count = 0;

for j = 1:xb

i = max(find(y(:, j)));

if isempty(i)

i = 0;

end

p = count + 1;

q = p + i;

r(p:q) = [y(1:i, j); eob];

count = count + i + 1;

end

r((count + 1):end) = [];

y = struct;

y.size = uint16([xm xn]);

y.numblocks = uint16(xb);

y.code = r;

jpeg2im.m:

function x = jpeg2im(y)

m = [16 11 10 16 24 40 51 61

12 12 14 19 26 58 60 55

14 13 16 24 40 57 69 56

14 17 22 29 51 87 80 62

18 22 37 56 68 109 103 77

24 35 55 64 81 104 113 92

49 64 78 87 103 121 120 101

72 92 95 98 112 100 103 99];

order = [1 9 2 3 10 17 25 18 11 4 5 12 19 26 33 ...

41 34 27 20 13 6 7 14 21 28 35 42 49 57 50 ...

43 36 29 22 15 8 16 23 30 37 44 51 58 59 52 ...

45 38 31 24 32 39 46 53 60 61 54 47 40 48 55 ...

62 63 56 64];

rev = order;

for k = 1:length(order)

rev(k) = find(order == k);

end

xb = double(y.numblocks);

sz = double(y.size);

xn = sz(2);

xm = sz(1);

x = y.code;

eob = max(x(:));

z = zeros(64, xb);

k = 1;

for j = 1:xb

for i = 1:64

if x(k) == eob

k = k + 1;

break;

else

z(i, j) = x(k);

k = k + 1;

end

end

end

z = z(rev, :);

x = col2im(z, [8 8], [xm xn], 'distinct');

x = blkproc(x, [8 8], 'x .\* P1', m);

t = dctmtx(8);

x = blkproc(x, [8 8], 'P1 \* x \* P2', t', t);

x = uint8(x + 128);

## Обнуление коэффициентов

На каждой итерации будет происходить постепенное обнуление коэффициентов ДКП в порядке, обратном зиг-загообразному.



Рис 3 – итерация 9 и 10



Рис 4 – итерация 11 и 12

Видно, что при каждой последующей итерации происходит потеря информации. В последней итерации остается всего лишь один коэффицент.

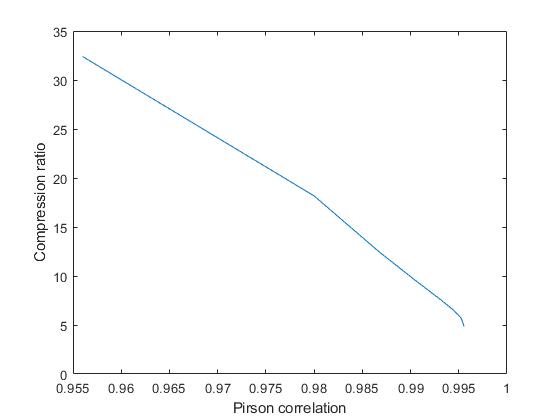
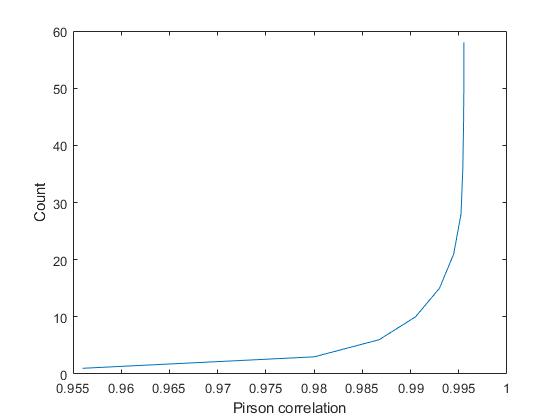


Рис 5 – зависимость количества коэффициентов от корреляции Пирсона, зависимость коэффициента сжатия от корреляции Пирсона

Видно, что в самом левом верхнем углу блока самые высокочастотные коэффициенты. А самые низкочастотные находятся в правом нижнем углу.

Part2.m:

f = imread('resources/gray.jpg');

figure, imshow(f)

cr\_array = zeros(12,1);

corr\_array = zeros(12,1);

count\_k = 1:12;

diagNumber = -5;

for i=1:12

mask = ones(8);

mask = triu(mask, diagNumber+i);

mask = flip(mask, 2);

f2 = im2jpeg(f, mask);

d1 = im2double(f); % compression ratio

d2 = f2.code; % compression ratio

cr1 = whos('d1'); % compression ratio

cr2 = whos('d2'); % compression ratio

cr = cr1.bytes / cr2.bytes; % compression ratio

f2 = jpeg2im(f2);

corr = corr2(f, f2);

cr\_array(i) = cr;

corr\_array(i) = corr;

count\_k(i) = sum(sum(mask));

figure, imshow(f2)

end

figure, plot(flip(corr\_array), flip(count\_k))

xlabel('Pirson correlation')

ylabel('Count')

figure, plot(flip(corr\_array), flip(cr\_array))

xlabel('Pirson correlation')

ylabel('Compression ratio')

## Анализ изображения при большом сжатии



Рис. 6 – изображение при большом сжатии

При использовании JPEG появляется большое количество артефактов. Однако если обнулять низкочастотные коэффициенты, то это позволит сжать картинку с минимальными потерями.