|  |
| --- |
| УНИВЕРСИТЕТ ИТМО |
| Лабораторная работа №5 по дисциплине «Цифровая обработка сигналов» |
| Сжатие изображений в системе Matlab |
| Группа Р3402 |
| **Выполнила: Орлова Кристина Александровна** |
| **Преподаватель: Тропченко Андрей Александрович** |

|  |
| --- |
| *22.01.20* |

**Цель**

* 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, обратному зиг-заг сканированию (начиная с высокочастотного, DC-коэффициент обнулять не нужно);
5. На каждой итерации с использованием функции compressRatio вычислите уровень сжатия полученного изображения и с использованием функции corr2 вычислите коэффициент корреляции Пирсона между исходным и сжатым изображением.
6. 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.
7. Определите приблизительное расположение высокочастотных, среднечастотных и низкочастотных коэффициентов ДКП. Определите зависимость между коэффициентом сжатия и качеством полученного изображения.
8. Анализируя внешний вид изображений, полученных при высоких коэффициентах сжатия, определите главный недостаток формата JPEG.

Исходный код

**\*compressRatio.m\***

function f = compressRatio(original, compressed)

f = numel(original) / numel(find(compressed ~= 0));

end

**\*directDCT.m\***

function f = directDCT(image, blockSize)

f = blkproc(image, [blockSize blockSize], 'dct2(x)');

end

**\*invertDCT.m\***

function f = invertDCT(image, blockSize)

f = blkproc(image, [blockSize blockSize], 'idct2(x)');

end

**\*quantization.m\***

function f = quantization(image, blockSize, mask)

f = blkproc(image, [blockSize blockSize], 'P1.\*x', mask);

end

**\*parseIndex.m\***

function [i, j] = parseIndex(index, n)

i = fix(index / n) + 1;

j = index - (i - 1) \* n;

if j == 0

i = i - 1;

j = n;

end

end

**\*iZigzag.m\***

function f = iZigzag(n)

operation = 1; %number of operation (one of four)

eF = 0; %end flag

k = 1; %new array's index

i = n;

j = n;

array = zeros(n);

while 1

index = (i - 1) \* n + j;

array(k) = index;

switch operation

case 1

if j > 1

j = j - 1;

else

if i > 1

i = i - 1;

else

eF = 1;

end

end

operation = operation + 1;

case 2

if i > 1 && j < n

i = i - 1;

j = j + 1;

else

operation = operation + 1;

k = k - 1;

end

case 3

if i > 1

i = i - 1;

else

if j > 1

j = j - 1;

else

eF = 1;

end

end

operation = operation + 1;

case 4

if j > 1 && i < n

j = j - 1;

i = i + 1;

else

operation = 1;

k = k - 1;

end

end

if eF == 1

break;

end

k = k + 1;

end

f = array;

end

**\*main.m\***

f = imread('cat.pgm');

figure, imshow(f);

f = im2double(f);

step = 5;

blockSize = 8;

mask = ones(blockSize);

zigzagOrder = iZigzag(blockSize);

B = directDCT(f, blockSize);

compressionRatio = zeros(1, 12);

correlationCoefficient = zeros(1, 12);

numberOfZeros = zeros(1, 12);

figure

for k = 1:12

for s = 1:step

index = zigzagOrder((k - 1) \* step + s);

[i, j] = parseIndex(index, blockSize);

mask(i, j) = 0;

end

B2 = quantization(B, blockSize, mask);

f2 = invertDCT(B2, blockSize);

compressionRatio(k) = compressRatio(f, B2);

correlationCoefficient(k) = corr2(f, f2);

numberOfZeros(k) = length(find(mask ~= 1));

disp(k);

subplot(3, 4, k), imshow(f2), title(numberOfZeros(k));

end

figure,

subplot(2, 1, 1), plot(numberOfZeros, correlationCoefficient),

ylabel('Коэффициент корреляции Пирсона'),

xlabel('Количество задействованных коэффициентов');

subplot(2, 1, 2), plot(compressionRatio, correlationCoefficient)

ylabel('Коэффициент корреляции Пирсона'),

xlabel('Степень сжатия');

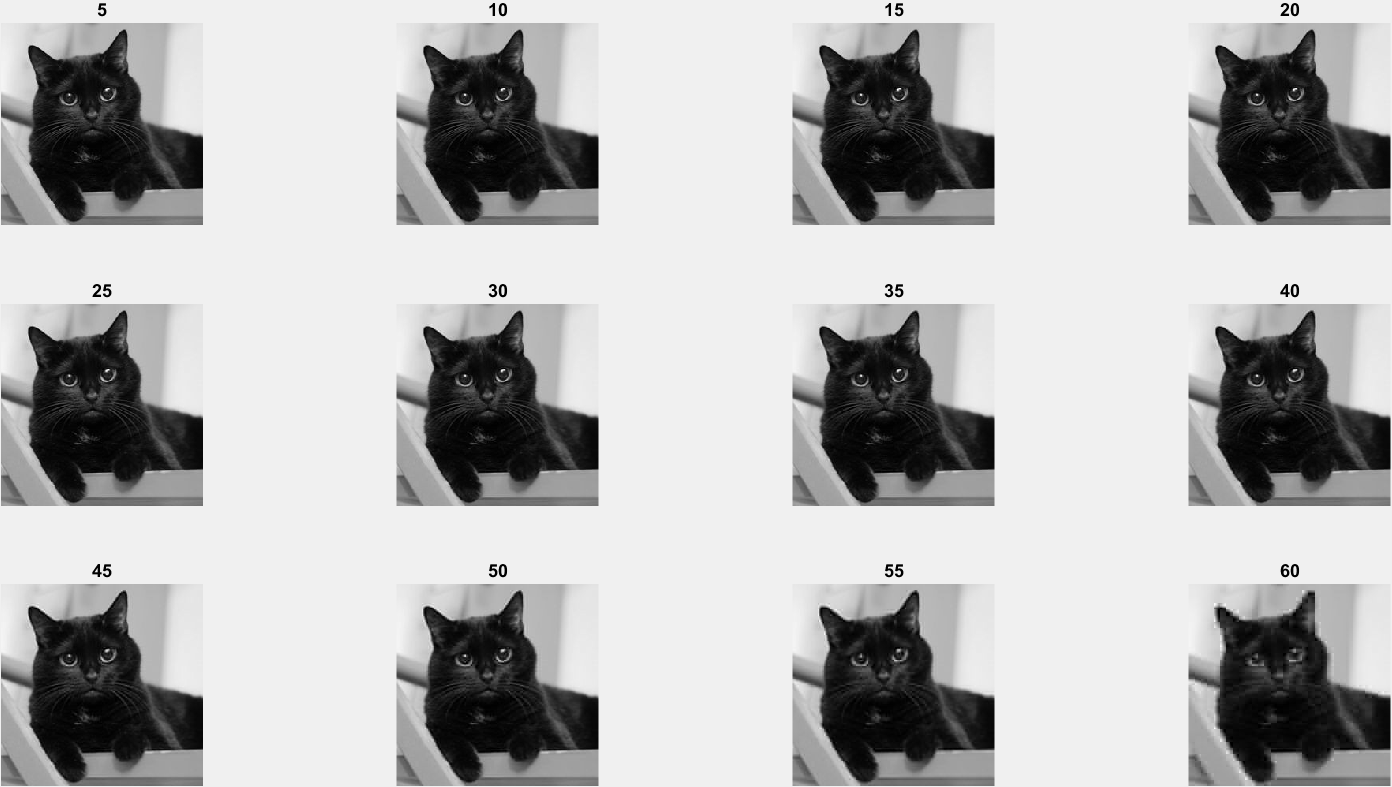
Результаты

Для демонстрации осуществлялось 12 итераций, на каждой из которых происходило постепенное обнуление коэффициентов ДКП в порядке, обратном зиг-заг сканированию. Был выбран шаг 5, то есть на каждой итерации обнулялось по 5 коэффициентов.

На Рисунке 2 каждое изображение, полученное на каждой итерации, подписано количеством обнуленных на этой итерации коэффициентов.



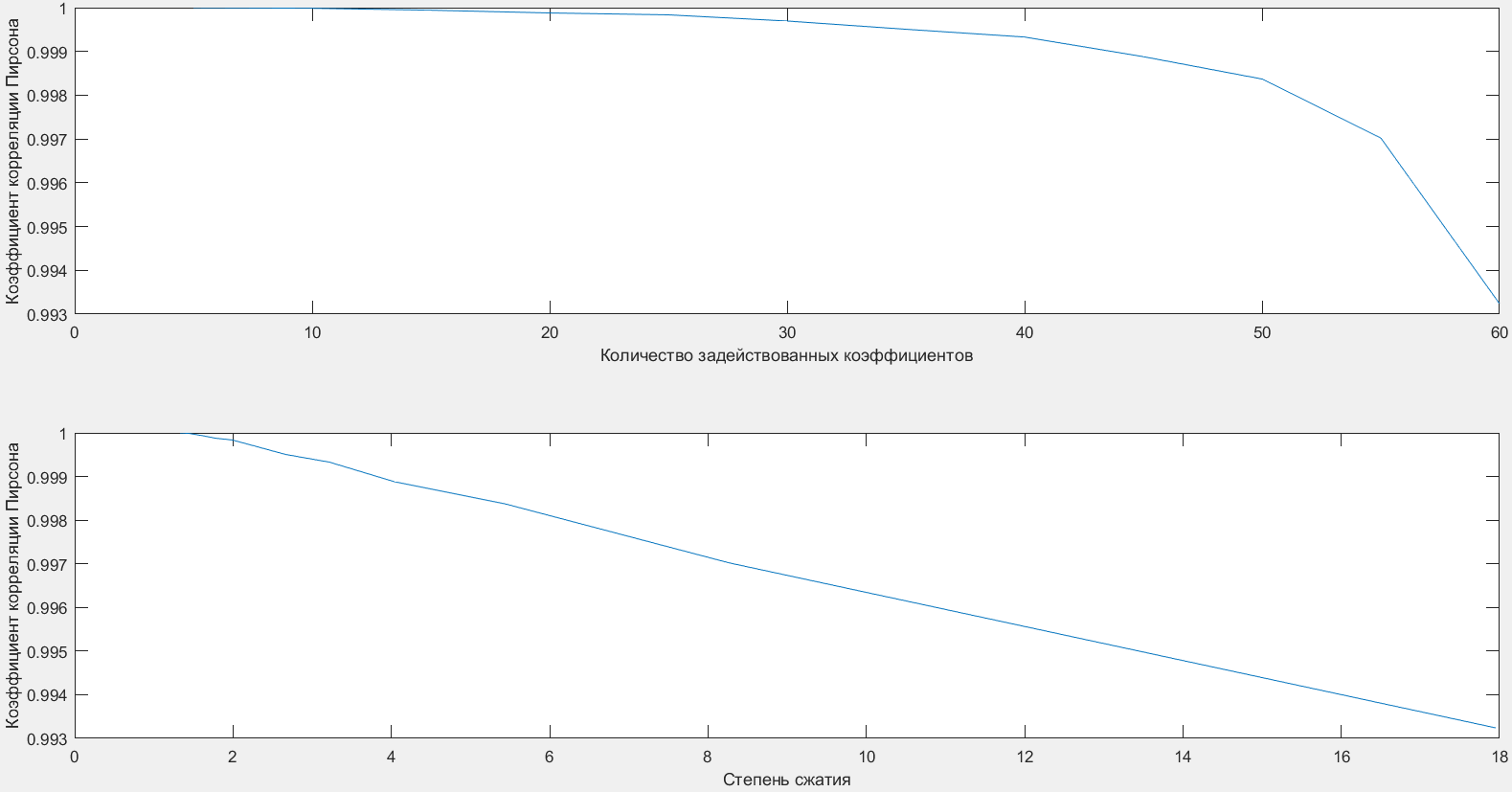
*Рисунок 1 - Исходное изображение*

****

*Рисунок 2 - Сжатые изображения*

Можно предположить, что низкочастотные коэффициенты - это DC-коэффициент и ближайшие к нему в порядке зиг-заг сканирования (допустим, первые 3). Среднечастотные коэффициенты - это следующие 10-15 коэффициентов, при обнулении которых мы начинаем видеть заметную разницу. Высокочастотные коэффициенты - пожалуй, все остальные.

На Рисунке 3 приведены графики зависимости а) между количеством задействованных коэффициентов и коэффициентом корреляции, б) между степенью сжатия и коэффициентом корреляции.



*Рисунок 3 - Графики*

По Рисунку 3 можно заметить, что качество изображения уменьшается с увеличением степени сжатия, что логично. Зависимость выглядит линейной, однако с некоторой вероятностью ограничена снизу значением, к которому стремится при увеличении степени сжатия до бесконечности.

**Выводы**

Главный недостаток формата JPEG состоит в том, что наибольшей степени сжатия можно добиться за счет операций над низкочастотными коэффициентами, в которых хранится больше всего информации об изображении. Однако обнуляя данные коэффициенты можно довести изображение до неузнаваемости и максимальной нечеткости, поэтому приходится довольствоваться операциями над высокочастотными коэффициентами, добиваясь лишь получения небольшой степени сжатия.