УНИВЕРСИТЕТ ИТМО

В.А. Батура
А.Ю. Тропченко
А.А. Тропченко

ОБРАБОТКА ИЗОБРАЖЕНИЙ В СИСТЕМЕ MATLAB: ЛАБОРАТОРНЫЕ РАБОТЫ

Санкт-Петербург
2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
УНИВЕРСИТЕТ ИТМО

В.А. Батура
А.Ю. Тропченко
А.А. Тропченко

ОБРАБОТКА ИЗОБРАЖЕНИЙ В СИСТЕМЕ MATLAB: ЛАБОРАТОРНЫЕ РАБОТЫ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 09.03.01, 09.03.04 в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

Санкт-Петербург
2019

Батура В.А., Тропченко А.Ю., Тропченко А.А. Обработка изображений в системе MATLAB: лабораторные работы– СПб: Университет ИТМО, 2019. – 41 с.

Рецензенты:
Поляков Владимир Иванович, кандидат технических наук, доцент, доцент (квалификационная категория "ординарный доцент") факультета программной инженерии и компьютерной техники, Университета ИТМО.

Целью учебного пособия является описание лабораторных работ, которые можно использовать для закрепления теоретических знаний, полученных в рамках лекционного курса. В пособии также приведены примеры тестовых вопросов, на которые необходимо знать ответы для успешной защиты лабораторных работ и написания рубежного контроля. Лабораторные работы выполняются с использованием современных средств обработки изображений – широкораспростаненного пакета программ математического моделирования MATLAB. Полученные навыки применения указанного пакета позволят студентам самостоятельно решать различные задачи моделирования обработки изображений и сигналов различной физической природы в ходе самостоятельных исследований и при работе над выпускной квалификационной работой

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Содержание

стр.

Введение

Цифровая обработка изображений охватывает широкий круг вопросов, связанных с изучением способов получения, представления, обработки, передачи, и хранения изображений.

Целью учебного пособия является описание лабораторных работ, которые можно использовать для закрепления теоретических знаний, полученных в рамках лекционного курса. В пособии также приведены примеры тестовых вопросов, на которые необходимо знать ответы для успешной защиты лабораторных работ и написания рубежного контроля.

Лабораторные работы выполняются с использованием современных средств обработки изображений – широко-распространенного пакета программ математического моделирования MATLAB. Полученные навыки применения указанного пакета позволят студентам самостоятельно решать различные задачи моделирования обработки изображений и сигналов различной физической природы в ходе самостоятельных исследований и при работе над выпускной квалификационной работой.

Лабораторная работа №1 «Сегментация изображений в системе Matlab» связана с применением различных методов выделения однородных локальных областей на изображении и решением тем самым задачи адаптивной сегментации. В этой же работе выполняется выделение контуров изображений с использованием различных градиентных фильтров и оценка эффективности их применения.

В лабораторной работе №2 «Сжатие изображений в системе Matlab» исследуется сжатие полутоновых и цветных изображений по стандарту JPEG и анализируется влияние различных параметров на качество восстанавливаемого изображения.

Лабораторная работа №3 «Восстановление изображений в системе Matlab» связана с исследованием процедуры шумоподавления при восстановлении цифровых изображений.

Лабораторная работа №4. «Сокрытие данных в изображении-стегоконтейнере» связана с исследованием методов сокрытия данных различными методами в изображении-стегоконтейнере, причем при ее выполнении студентам требуется самостоятельно разработать и запрограммировать соответствующие алгоритмы с использованием систеы Matlab.

Приведенный в пособии материал рекомендуется использовать при реализации дисциплины «Методы обработки и распознавания изображений» в объеме семестрового учебного курса образовательных программах бакалавриата по группам направлений подготовки «Компьютерные и информационные науки», «Информатика и вычислительная техника». «Программная инженерия»

1. Лабораторная работа № 1.
Сегментация изображений в системе Matlab

Цели лабораторной работы:

- Реализация методов сегментации изображений в системе Matlab;
- Исследование эффективности методов сегментации для различных видов изображений;
- Реализации градиентных фильтров для выделения границ изображения.

Matlab - это современный пакет прикладных программ, широко использующийся для облегчения и частичной автоматизации задач математического моделирования в различных областях инженерных вычислений. К настоящему времени практически стандартом стало использование Matlab для решения задач цифровой обработки изображений, моделирования различных систем автоматического управления и искусственных нейронных сетейю.

В состав Matlab дополнительно входит пакет Image Processing Toollbox (IPT), который предназначен для решения задач цифровой обработки изображений. Цикл описываемых лабораторных работ основан на использовании именно данного программного пакета.

1.1 Исследуемые методы сегментации

В данной лабораторной работе рассматриваются следующие методы сегментации изображения:

- Центроидное связывание;
- Метод слияния/расщепления изображения;
- Методы выделения границ на основе градиентных фильтров.

В качестве исходных графических файлов выбраны полутоновые изображения.

1.1.1 Центроидное связывание

Метод центроидного связывания относится к методу наращивания областей. На исходном изображении пользователь заранее размещает точки (центры кристаллизации), относительно которых будет происходить наращивание областей. Затем вокруг этих точек наращиваются области путем добавления к каждому центру тех соседних пикселов, которые по своей яркости близки к яркости точки -центра кристаллизации [1]. Пикселам каждой области присваивается определенная метка – идентификатор области (например, числом). В случае слияния областей пикселы объединенной новой области получают единую метку, что прводит к необходимости переобозначения их и корректировки таблицы меток.

Пакет IPT системы Matlab содержит большое разнообразие функций цифровой обработки изображения, в данной системе отсутствует готовая реализация метода центроидного связывания. Поэтому для ознакомления с данным методом сегментации была используется функция *regiongrow*, которая имеет следующий синтаксис [1]:

[g, NR, SI] = regiongrow(f, S, T)

где *f* – изображение, подлежащее сегментации; *S* – массив, определяющий местоположение центров кристаллизации, которые задаются путем присвоения элементу массива значения 1 (элементам массива *S*, не относящимся к центрам кристаллизации присваивается значение 0, при этом *S* может быть скаляром и тогда центрами кристаллизации будут все точки с данной яркостью); *T* – глобальный порог кристаллизации; *g* – сегментированное изображение, причем элементы каждой сегментированной области помечаются целым числом [1]; *NR* – число выделенных областей; *SI* – изображение, отображающее центры кристаллизации.

Рассмотрим пример использования данной функции. Для её использования необходимо изменить рабочую папку на директорию с файлом данной функции (*regiongrow.m*).

В качестве исходного изображения было взято полутоновое изображение, представленное на рисунке 1.1.

Пример:

```
f = imread('F:/M/face.png');        %Загружаем изображение
Mask = zeros(size(f));              %Создаем нулевой массив
Mask(38,79) = 1;                    %Выбираем точки кристаллизации
Mask(93,64) = 1;
Mask(92,117) = 1;
Mask(100,64) = 1;
[g NR SI]= regiongrow(f, Mask, 30);          %Сегментируем изображение
figure, imshow(f)
figure, imshow(g)          %Выводим на экран результат сегментации в виде бинарного изображения
figure, imshow(SI)         %Выводим на экран точки кристаллизации
```

Результат сегментации представлен на рисунке 1.2.

Рисунок 1.1. – исходное изображение

Рисунок 1.2. – результат сегментации

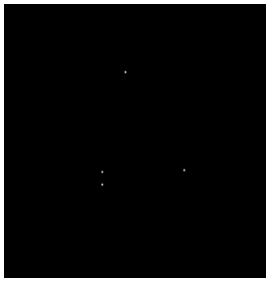


Рисунок 1.3. – точки кристаллизации

1.1.2 Метод разделения-слияния

Как видно из названия, метод подразделяется на два этапа: разделение изображения на ряд областей и последующее слияние части из них.

Разделение можно представить в виде квадродерева (рисунок 1.4) [1].

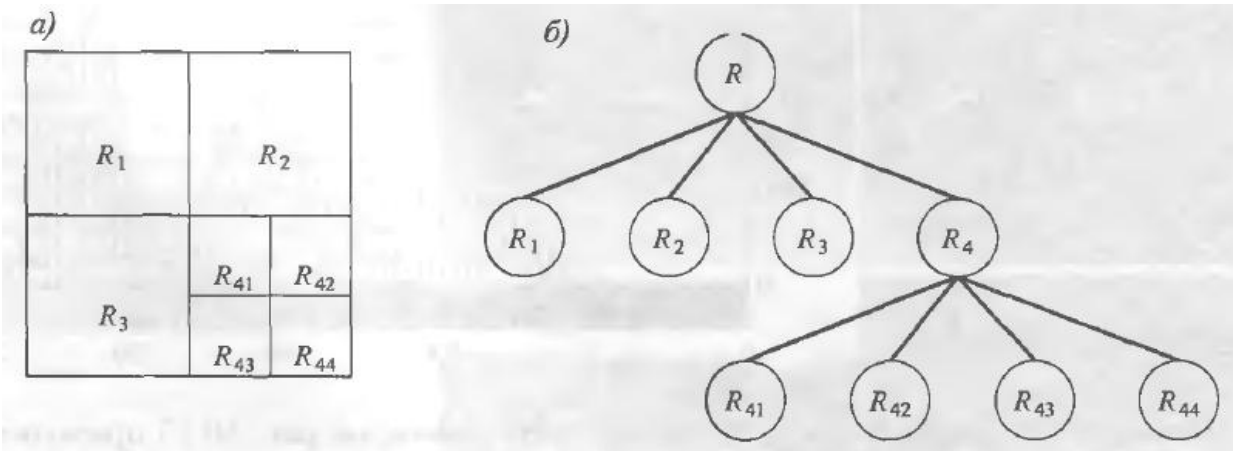


Рисунок 1.4

а) Деление изображения на части. б) Соответствующее квадродерево

Как видно из рисунка 1.4, исходное изображение *R* делится на части. Деление осуществляется в соответствие с правилом *P*.

Алгоритм деления в общем виде может быть записан следующим образом [1]:

- Любая квадратная область *Rj*, для которой *P(Rj) = FALSE*, разделяется на четыре квадратные непересекающиеся блока - четверти.
- Любые две соседние области *Rj* и *Rk*, для которых *P(Rj∪Rk) = TRUE*, объединяются в одну.
- Процедура завершается, если невозможно выполнить ни одну операцию слияния или разделения.

В пакете IPT предусмотрена функция разложения изображения по квадродереву *imdecomp*, которая имеет следующий синтаксис:

S = qtdecomp(f, @split_test, parameters)

где *f* – исходное изображение; *S* – разреженная матрица, которая представляет собой структуру квадродерева; (при этом элемент матрицы *S(k, m)*, не равный нулю, является верхним левым углом квадратного блока разложения и размер этого блока равен *S(k, m)* [1]); *split_test* – любая функция, определяющая допустимость разделения данной области на части; parameters – параметры, необходимые функции *split_test*.

Для получения значений блоков, получившихся в результате разделения изображения, в пакете IPT предусмотрена функция *qtgetblk*, которая имеет следующий синтаксис:

[vals, r, c]=qtgetblk(f, S, m)

где *vals* представляет собой массив, содержащий все блоки фиксированного размера m, получившиеся в результате сегментации полутонового изображения f с помощью функции *qtdecomp*. Параметр *S* определяет разреженный массив, описывающий квадро-дерево с результатами сегментации. Векторы *r* и *c* содержат соответственно номера строк и столбцов левых верхних углов блоков изображения, помещенных в массив *vals*. Если при сегментации не получено ни одного блока размера m, то всем возвращаемым параметрам присваиваются значения пустых матриц [2].

Рассмотрим пример квадратичного разложения изображения.

Пример:

```
I = imread('cameraman.tif');
S = qtdecomp(I,0.5);
blocks = repmat(uint8(0),size(S));

for dim = [512 256 128 64 32 16 8 4 2 1];
  numblocks = length(find(S==dim));
  if (numblocks > 0)
    values = repmat(uint8(1),[dim dim numblocks]);
    values(2:dim,2:dim,:) = 0;
    blocks = qtsetblk(blocks,S,dim,values);
  end
end

blocks(end,1:end) = 1;
blocks(1:end,end) = 1;

imshow(I), figure, imshow(blocks,[])
```

Исходное изображение представлено на рисунке 1.5, разложенное изображение – на рисунке 1.6.

Рисунок 1.5. Исходное изображение

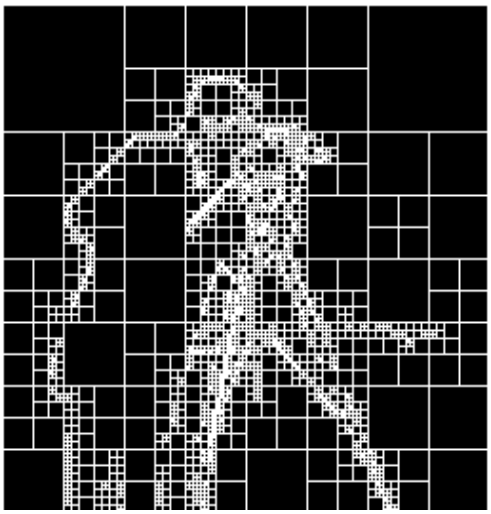


Рисунок 1.6. – Разложенное по квадродереву изображение

В пакете IPT полная реализация метода разделения-слияния отсутствует. Для демонстрации данного метода была использована функция *splitmerge*, представленная в [1]. Для удобства в данной лабораторной работе она была несколько изменена и имеет следующий синтаксис:

g = splitmerge(f, mindim, @predicate, matOj, minMean, maxMean)

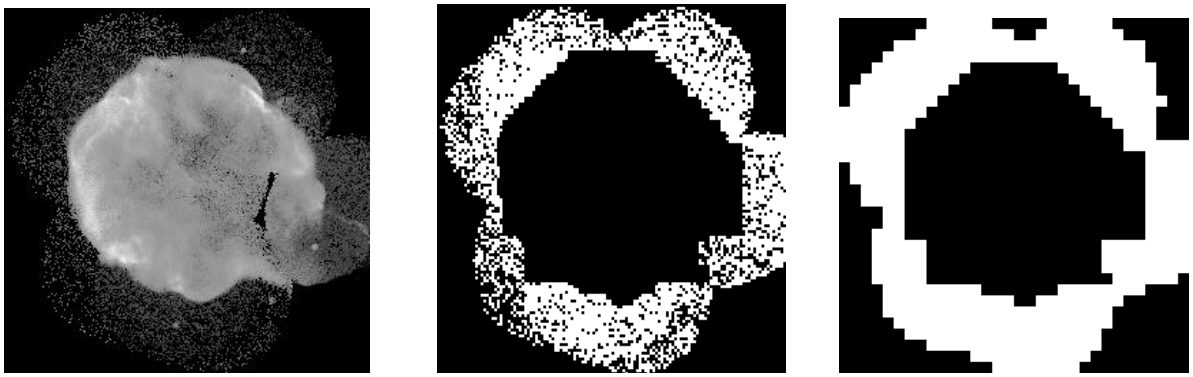
где *g* – сегментированное изображение, *f* – исходное изображение; *mindim* – минимальный размер блока декомпозиции изображения, принимающий значение 2^n *(n = 1, 2, 3…)* (при этом размер mindim не может превышать размера исходного изображения); *predicate* – функция, определяющая возможность дальнейшего разбиения блока; *matOj, minMean, maxMean* – параметры функции fun, а именно – среднеквадратичное отклонение матрицы яркости пикселов блока, минимальное и максимальные средние значения яркости пикселов блока.

Рассмотрим пример использования данной функции. Пусть на рисунке 1.7(а) необходимо выделить менее яркую область, окружающую центральную более яркую область [1]. Интересующая область имеет несколько характеристик, которые могут помочь при сегментации. Так, яркость пикселов имеет случайный характер, т.е. стандартное отклонение их яркости превышает стандартное отклонение фона (равного 0) и стандартного отклонения яркости яркой центральной области. Аналогично, среднее значения яркости выделяемой области больше среднего значения фона и меньше среднего значения центральной области". Поэтому параметры при сегментации примут следующие значения*: matOj = 10, minMean = 0, maxMean = 125*

Пример:

```
I = imread('F:/M/fig.tif');
a = splitmerge(double(f), 2, @predicate, 10, 0, 125);
figure, imshow(a)
a = splitmerge(double(f), 8, @predicate, 10, 0, 125);
```

figure, imshow(a)



а) б) в)

Рисунок 1.7

а – исходное изображение,
б – фрагментированное изображение с блоками минимального размера 2x2,
в – фрагментированное изображение с блоками минимального размера 8x8)

Как видно из рисунка 1.7, для корректного сегментирования лучшим выбором был размер минимальных блоков 8x8.

1.2 Выделение границ объектов изображения с помощью градиентных фильтров

Для нахождения границ объектов изображения часто используют градиентные фильтры. В упрощенной форме градиентный фильтр представляет маску размером 3x3, которая скользящим окном построчно проходит по всем пикселям изображения. Данный процесс отображен на рисунке 1.8 [3]. В процессе такого прохода матрица пикселов размером 3x3 (центральный пиксел и 8 прилегающих к нему) поэлементно умножается на маску фильтра. Если матрица пикселы выходит за пределы изображения, то ее выходящие за пределы изображения пикселы считаются равными нулю.

Полученная на каждом шагу матрица суммируется по представленной ниже формуле [3]:

$$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots + w(0,0)f(x,y) + \cdots + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1)$$

где R – соответствующий Gx или GY при использовании матрицы градиентного фильтра, предназначенного для оси OX или OY. Конечный пиксель обработанного изображения получается по следующей формуле:

$$f = \sqrt{G_x^2 + G_y^2}$$

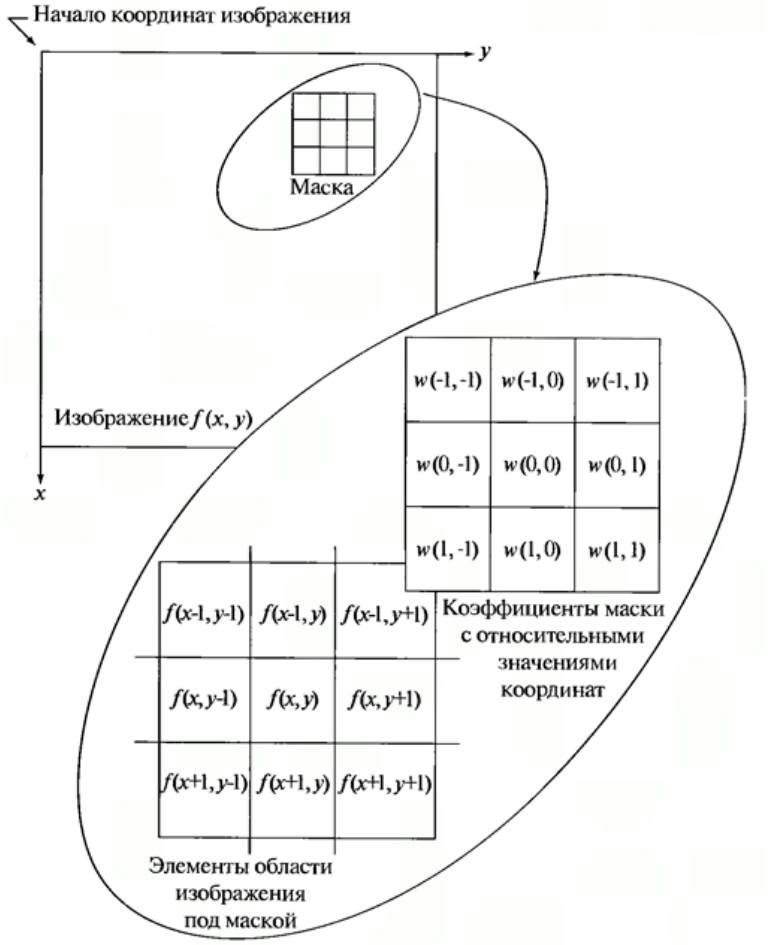


Рисунок 1.8. – Схема работы градиентного фильтра.

В данной лабораторной работе рассматриваются фильтры Робертса, Собеля и Превитта, маски которых имеют следующий вид [4]:

а) Робертса:

по оси OX

0	0	0
0	-1	0
0	0	1

по оси OY

0	0	0
0	0	-1
0	1	0

б) Собеля:

по оси OY

по оси OX

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

в) Превитта:

По оси OX

-1	-1	-1
0	0	0
1	1	1

по оси OY

-1	0	1
-1	0	1
-1	0	1

В пакете IPT градиентные фильтры реализуются с помощью функции *edge*, записываемой в следующем виде

BW=edge(l, method, thresh)

где *BW* – бинаризованное сегментированное изображение, *I* – исходное изображение, *method* – выбранный градиентный фильтр. Например, *method* может принимать значения *'sobel'*, *'prewitt'*, *'roberts'*.; *Thresh* – порог бинаризации сегментированного изображения.

Пример [4]:

```
% Пример демонстрирует выделение границ методом Собеля.
% Чтение исходного изображения и вывод его на экран.
I=imread('lena.bmp');
imshow(l);
% Выделение границ методом Собеля и вывод результата на экран.
BW=edge(l, 'sobel', 0.09);
figure, imshow(BW);
```

Результаты представлены на рисунке 1.9.

а)

б)

Рисунок 1.9. – Выделение границ объектов изображения фильтром Собеля (а – исходное изображение, б – сегментированное)

1.3. Задание

Задание № 1

1. Используя полученное изображение лица человека, при помощи приведенной в разделе 2.1.1 функции осуществите его сегментацию методом центроидного связывания. Необходимо выделить лицо человека, добившись выделения максимально возможной области.
2. По результатам выполнения п.1 выведите на экран соответствующий фрагмент полутонового изображения, используя результат сегментации как маску.

Полученные изображения и программный код (за исключением программного кода реализации функции *regiongrow*) поместите в отчет.

Задание № 2

Используя представленный выше программный код, осуществите квадратичное разложение полученного для данного упражнения изображения. Постарайтесь выбрать порог, обеспечивающий наиболее точное выделение основной части изображения.

Программный код и полученные результаты поместите в отчет.

Задание № 3

Используя функцию *splitmerge* примените метод разделения-слияния на полученном для данного задания изображении. Постарайтесь выбрать параметры *matOj, minMean, maxMean* таким образом, чтобы добиться максимально возможного выделения основного объекта изображения.

Программный код и полученные изображения поместите в отчет.

Задание № 4

Не используя функцию *edge*, создайте три функции *Roberts.m, Sobel.m, Previtt.m,* реализующих работу каждого из рассмотренных выше фильтров с помощью масок по алгоритму, представленному на рисунке 8.

Выполнение функций проверьте на заранее полученных для этого задания изображениях разного типа.

Листинг программы и полученные изображения поместите в отчет. В отчете сделайте выводы о фильтрах Собеля и Превитта, сравнив их эффективность работы по полученным результатам.

1.3.1. Содержание отчета

- Титульный лист
- Тексты заданий
- Исходный текст программ с комментариями
- Полученные результаты (рисунки) с подписями
- Выводы по проделанной работе

Контрольные вопросы

Что представляет собой сегментация изображения?

В чем заключается сегментация по порогу?

В чем отличие процедуры сегментации по диапазону от сегментации по порогу?

В чем отличие метода центроидного связывания от метода разделение-слияния?

Поясните процесс выделения контуров изображения на основе градиентных фильтров.

Укажите основные виды градиентных фильтров.

Литература

1. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде Matlab, Москва: Техносфера, 2006. – 616с.

2. http://matlab.exponenta.ru/imageprocess/book3/12/qtgetblk.php

3. http://habrahabr.ru/post/114452/

4. http://matlab.exponenta.ru/imageprocess/book3/12/edge.php

2. Лабораторная работа № 2
Сжатие изображений в системе Matlab

Цели лабораторной работы:

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

2.1. Сжатие изображений – основные понятия

Сжатие изображений основывается на их изначальной избыточности, что позволяет при ее устранении сократить объем данных, необходимых для представления цифрового изображения. Методы сжатия подразделяется на две категории [1]:

- сжатие без потерь – восстановленное изображение полностью идентично первоначальному;
- сжатие с потерями – восстановленное изображение не является идентичным первоначальному.

Ко второй категории относится формат сжатия *JPEG*, который будет рассмотрен в данной лабораторной работе. Данный формат сжатия основан на удалении визуальной избыточности, заключающейся в наличии в изображении информации, не воспринимаемой органами зрения человека [1]. Степень сжатия определяется по представленной ниже формуле:

$$K_{сж} = \frac{Im_1}{Im_2} \quad ,$$

где Im_1 - объем исходного изображения в байтах, Im_2 - объем сжатого изображения в байтах, $k_{сж}$ – безразмерный коэффициент сжатия.

Для получения информации о размере изображения (в байтах), а также другие сведения о графическом файле используется функция *imfinfo*, синтаксис которой приведен ниже:

$$info = imfinfo(path) \quad ,$$

где *path* – строковая переменная, содержащая путь к файлу изображения, *info* – структура, хранящая в себе сведения об изображении.

Пример:

```
info = imfinfo('F:/M/fishboat.png')

info =

        Filename: 'F:/M/fishboat.png'
    FileModDate: '09-may-2018 13:20:19'
        FileSize: 177762
          Format: 'png'
   FormatVersion: []
           Width: 512
          Height: 512
        BitDepth: 8
       ColorType: 'grayscale'
  FormatSignature: [137 80 78 71 13 10 26 10]
```

Colormap: []
Histogram: []
InterlaceType: 'none'
Transparency: 'none'
...
XResolution: 47244
YResolution: 47244
ResolutionUnit: 'meter'

Как видно, структура info содержит ряд сведений об изображении (формат, разрешение, число битов на пиксель и т. д.). Поскольку список сведений весьма обширен, некоторые свойства не были отображены в данном примере.

Для получения конкретного свойства из данной структуры достаточно написать название данной структуры, поставить после нее точку, а после нее указать название необходимого свойства.

Например:

size = info.FileSize
size =
177762

Как видно из примера, файл *'F:/M/boat.png'* занимает объем *177762* байтов.

2.2. Сжатие по стандарту JPEG

Формат сжатия JPEG – самый популярный на сегодняшний день формат сжатия в сети Интернет, использование которого позволяет достичь высоких коэффициентов сжатия при относительно хорошем качестве изображения.

Упрощенная схема алгоритма сжатия JPEG представлена на рисунке 2.1.

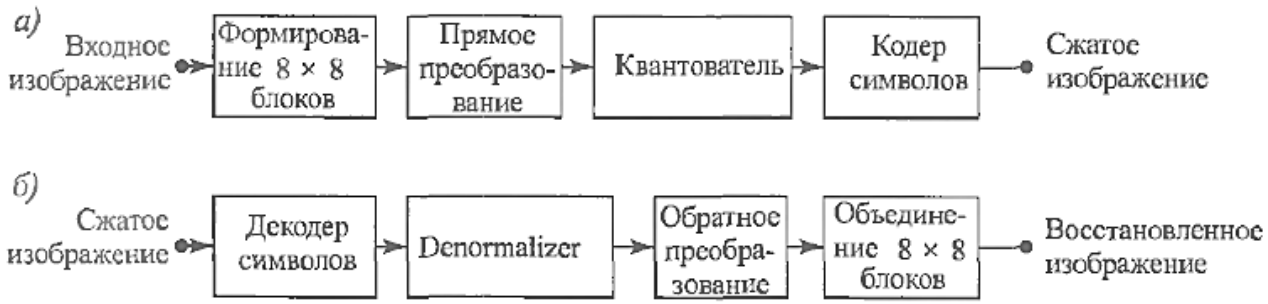


Рисунок 2.1. – а) Сжатие JPEG. б) Восстановление изображения после сжатия JPEG

В пакете IPT отсутствует функция, реализующая полностью процесс JPEG-сжатия. Однако в Matlab есть ряд функций, которые позволят в упрощенной форме рассмотреть основные этапы данного алгоритма, а именно: формирование блоков изображения, прямое преобразование и квантование.

Сжатие изображения. Первым этапом алгоритма JPEG является разделение изображения на блоки размером 8×8 пикселов. Строго размер блока

не оговаривается, но все блоки должны быть одного размера (например, возможен и размер 16×16).

На втором этапе каждый блок подвергается двумерному дискретному косинусному преобразованию (ДКП). Использование ДКП увеличивает вычислительную сложность алгоритма.

$$ДКП(i,j) = C(i) * C(j) * \sum_{x=1}^{N}\sum_{y=1}^{N} f(x,y) * cos\left[\frac{(2x+1)i\pi}{2N}\right] * cos\left[\frac{(2y+1)j\pi}{2N}\right],$$

$$C(i),C(j) = \begin{cases} \sqrt{\frac{1}{N}} \text{ при } i,j = 1 \\ \sqrt{\frac{2}{n}} \text{ при } i,j = 2,\dots,N \end{cases}$$

где ,*f(x, y)* – пиксел изображения, *N* – размер квадратного блока (в данном случае равен 8).

В результате формируется матрица частотных коэффициентов преобразования (рисунок 2.2). Коэффициент в верхнем левом углу данной матрицы называется DC-коэффициентом, остальные – AC-коэффициенты. ДКП позволяет эффективно перераспределить энергию изображения, основная часть которой содержится в DC-коэффициенте. Полученные коэффициенты преобразования также подразделяются на 3 категории:

- высокочастотные – содержат незначительную часть энергии изображения;
- среднечастотные – содержат энергию, связанную именно со спецификой сжимаемого изображения;
- низкочастотные – содержат наибольшую часть энергии изображения.

Энергия коэффициентов убывает в порядке Z сканирования коэффициентов разложения фрагмента, начиная с DC-коэффициента (рисунок 2.2).

В системе Matlab есть два способа реализации ДКП. Первый вариант заключается в использовании функции *DCT2*, которая имеет следующий синтаксис:

coefficients = dct2(image),

где *image* – матрица элементов (в данном контексте матрица яркостей блока пикселов изображения, имеющая размеры 8×8), *coefficients* – матрица коэффициентов преобразования.

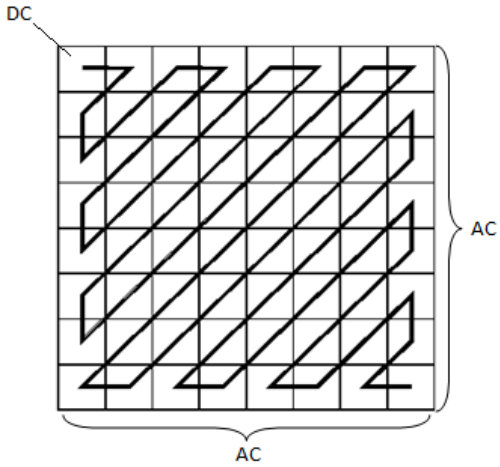


Рисунок 2.2. – Местоположение DC и AC-коэффициентов ДКП.

Пример:

f = [73 255 63 49 68 50 204 180;
71 170 97 65 154 49 10 24;
51 38 33 137 73 51 44 70;
66 111 219 9 76 185 143 200;
70 11 82 23 96 98 31 195;
200 162 83 36 33 202 69 140;
60 16 82 49 45 90 52 39;
56 40 96 41 29 42 153 128];
transformCoeff1= dct2(f)

transformCoeff1 =

704.6250	*-49.6535*	*103.6456*	*-33.4176*	*-53.1250*	*-73.7958*	*45.4189*	*5.9710*
64.1862	*47.5690*	*-23.5854*	*-32.7407*	*-5.9890*	*-85.0273*	*-125.2335*	*-7.0272*
-37.9819	*40.5304*	*17.3512*	*-19.4245*	*-29.7999*	*29.4565*	*-163.6131*	*11.0492*
79.9618	*26.4890*	*103.4291*	*52.1656*	*-20.6781*	*-65.8899*	*-3.3497*	*-44.0490*
80.8750	*-129.1831*	*100.1149*	*-88.0779*	*-52.8750*	*-10.0577*	*-4.2617*	*62.1714*
43.6461	*-41.5915*	*25.0799*	*-16.3366*	*-77.8999*	*40.6438*	*3.2204*	*-10.8930*
51.0457	*-17.2876*	*83.8869*	*11.9570*	*10.0434*	*42.8170*	*-40.1012*	*-71.4507*
-140.7993	*-45.9739*	*-56.1612*	*-20.5906*	*113.5466*	*43.8998*	*-32.7630*	*-6.3783*

В данном примере DC-коэффициент равен 704.6250. Как видно из результатов, DC-коэффициент действительно содержит б**о**льшую часть энергии данного изображения.

Существует и альтернативный вариант реализации ДКП строчно-столбцовым методом. При таком подходе прямое двумерное ДКП будет осуществляться по формуле :

$$B=C*A*C^T,$$

где - A – исходное изображение или его блок, C – матрица преобразования ДКП, C^T – транспонированная матрица преобразования ДКП.

В Matlab матрица преобразования ДКП может быть получена при помощи команды *dctmtx*, имеющей следующий синтаксис:

$$C = dctmtx(dimension),$$

где *dimension* – размер квадратной матрицы преобразования, C – полученная матрица преобразования ДКП.

Пример:

```
%В примере используем ту же матрицу f
T = dctmtx(8)
transformCoeff2= C*f*T'
```

C =

0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536
0.4904	0.4157	0.2778	0.0975	-0.0975	-0.2778	-0.4157	-0.4904
0.4619	0.1913	-0.1913	-0.4619	-0.4619	-0.1913	0.1913	0.4619
0.4157	-0.0975	-0.4904	-0.2778	0.2778	0.4904	0.0975	-0.4157
0.3536	-0.3536	-0.3536	0.3536	0.3536	-0.3536	-0.3536	0.3536
0.2778	-0.4904	0.0975	0.4157	-0.4157	-0.0975	0.4904	-0.2778
0.1913	-0.4619	0.4619	-0.1913	-0.1913	0.4619	-0.4619	0.1913
0.0975	-0.2778	0.4157	-0.4904	0.4904	-0.4157	0.2778	-0.0975

transformCoeff2 =

704.6250	-49.6535	103.6456	-33.4176	-53.1250	-73.7958	45.4189	5.9710
64.1862	47.5690	-23.5854	-32.7407	-5.9890	-85.0273	-125.2335	-7.0272
-37.9819	40.5304	17.3512	-19.4245	-29.7999	29.4565	-163.6131	11.0492
79.9618	26.4890	103.4291	52.1656	-20.6781	-65.8899	-3.3497	-44.0490
80.8750	-129.1831	100.1149	-88.0779	-52.8750	-10.0577	-4.2617	62.1714
43.6461	-41.5915	25.0799	-16.3366	-77.8999	40.6438	3.2204	-10.8930
51.0457	-17.2876	83.8869	11.9570	10.0434	42.8170	-40.1012	-71.4507
-140.7993	-45.9739	-56.1612	-20.5906	113.5466	43.8998	-32.7630	-6.3783

Если сравнить матрицы *transformCoeff1* и *transformCoeff2*, то очевидно, что результат полностью идентичный.

После получения матрицы коэффициентов преобразования осуществляется их квантование для уменьшения их разрядности по формуле при помощи соответствующей матрицы квантования (рисунок 2.3) [2] :

$$f'(u,v)=round\left(\frac{f(u,v)}{T(u,v)}\right),$$

где *f(u,v)* – элемент матрицы частотных коэффициентов, полученных в результате применения ДКП; *T(u,v)* – элемент матрицы квантования, *f'(u,v)* – квантованный частотный коэффициент. Заметим, что при размере блока 8×8 осуществляется переход к шести битам для представления каждого коэффициента. Это позволяет своим собственным кодом представить любой из 64-х коэффициентов.

На данном этапе происходит наибольшая потеря информации. Каждый частотный коэффициент делится на соответствующий элемент матрицы квантования. В результате происходит уменьшение количества информации, вплоть до обнуления некоторых частотных коэффициентов.

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Рисунок 2.3. – Матрица квантования

Пример:

%В качестве матрицы коэффициентов возьмем матрицу из предыдущего примера.

```
quantMatrix = [3 5 7 9 11 13 15 17;
5 7 9 11 13 15 17 19;
7 9 11 13 15 17 19 21;
9 11 13 15 17 19 21 23;
11 13 15 17 19 21 23 25;
13 15 17 19 21 23 25 27;
15 17 19 21 23 25 27 29;
17 19 21 23 25 27 29 31];
quantedCoeff = round(transformCoeff2 ./ quantMatrix)
quantedCoeff =
  235   -10    15    -4    -5    -6     3     0
   13     7    -3    -3     0    -6    -7     0
   -5     5     2    -1    -2     2    -9     1
    9     2     8     3    -1    -3     0    -2
    7   -10     7    -5    -3     0     0     2
    3    -3     1    -1    -4     2     0     0
    3    -1     4     1     0     2    -1    -2
   -8    -2    -3    -1     5     2    -1     0
```

В результате квантования и округления величина коэффициентов достаточно сильно изменилась. Некоторые из коэффициентов обнулились.

После того, как каждый блок частотных коэффициентов был подвергнут квантованию с помощью Z сканирования (рисунок 2.2), коэффициенты каждого блока выстраиваются в виде вектора-строки. Для дополнительного уменьшения квантования применяется сжатие без потерь (в данном случае - кодирование Хаффмана). Однако данный этап выходит за рамки данной лабораторной работы.

2.2.1 Восстановление изображения

Восстановление изображения осуществляется при помощи тех же операций, но совершаемых в обратной последовательности.

Операция обратная квантованию совершается по формуле:

$$f'(u,v)=f(u,v)*T(u,v),$$

где $f(u,v)$ – элемент матрицы частотных коэффициентов, полученных в результате применения ДКП; $T(u,v)$ – элемент матрицы квантования; $f'(u,v)$ – квантованный частотный коэффициент.

В дальнейшем для получения значения яркости пикселов применяется обратное двумерное ДКП, определяемое по формуле:

$$f(x,y)=\sum_{i=1}^{N}\sum_{j=1}^{N} C(i)*C(j)*\text{ДКП}(x,y)*\cos\left[\frac{(2x+1)i\pi}{2N}\right]*\cos\left[\frac{(2y+1)j\pi}{2N}\right],$$

$$C(i),C(j)=\begin{cases}\sqrt{\frac{1}{N}} \text{ при } i,j=1\\ \sqrt{\frac{2}{n}} \text{ при } i,j=2,\dots,N\end{cases}$$

Обратное ДКП обратное ДКП выполняется строчно-столбцовым методом по формуле:

$$B=C^T*A*C$$

После данного этапа все блоки объединяются в единое изображение.

Пример:

%В качестве матрицы квантованных коэффициентов возьмем матрицу quantedCoeff
из предыдущего примера.
dequantedCoeff = quantedCoeff . quantMatrix;*
restoredImage = idct2(dequantedCoeff)
dequantedCoeff =

```
   705   -50   105   -36   -55   -78    45     0
    65    49   -27   -33     0   -90  -119     0
   -35    45    22   -13   -30    34  -171    21
    81    22   104    45   -17   -57     0   -46
    77  -130   105   -85   -57     0     0    50
    39   -45    17   -19   -84    46     0     0
    45   -17    76    21     0    50   -27   -58
  -136   -38   -63   -23   125    54   -29     0
```

restoredImage =

```
  76.4742  245.9975   68.5861   48.6337   67.1943   48.4929  207.8460  175.9226
  72.7499  184.2138   89.6165   64.6029  159.1941   46.6957    3.6504   36.3835
  51.3591   32.2805   43.0143  132.0959   84.7295   51.0817   38.4603   62.2768
  53.8345  116.8188  213.3658    6.0500   68.0776  191.0178  140.6799  202.3203
  73.7003   10.4359   79.6316   36.7039   86.3254  101.0582   23.1711  203.5607
 192.5861  159.7988   85.6039   30.8471   29.5488  207.2119   70.0499  140.5297
  57.2327   21.3216   81.4513   55.8806   48.7473   80.1200   57.4293   41.2360
  60.6383   46.4910   89.8288   36.4148   21.8683   47.1630  157.4156  122.2810
```

Данный пример в несколько этапов выполнялся для одного блока размером 8×8. Чтобы повторить данную операцию на всех блоках потребуется функция *blcproc*. Данная функция обеспечивает выполнение какой-либо операции на всех блоках изображения заранее заданного размера. Функция blcproc имеет следующий синтаксис:

F=blkproc(S, [m n], fun, P1, P2, ...) ,

где *S* – исходное изображение; *m* и *n* – размерность блоков изображения, которые будут подвергаться обработке; *fun* – функция, использование которой обеспечивает обработку каждого блока; *P* – параметры функции. В представленном ниже примере вместо квантования для упрощения было осуществлено обнуление некоторых частотных коэффициентов каждого блока изображения путем наложения маски.

Пример:

```
f = imread('C:/ lena.png');
f = im2double(f);
B = blkproc(f,[8 8],'dct2(x)');                    %Получаем набор частотных коэффициентов для каждого блока
mask = [1 1 1 1 0 0 0 0;                           %Создаем маску для обнуления частотных коэффициентов
        1 1 1 0 0 0 0 0;
        1 1 0 0 0 0 0 0;
        1 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0];
B2 = blkproc(B,[8 8],'P1.*x',mask);                %Обнуляем ряд частотных коэффициентов, накладывая маску
f2 = blkproc(B2,[8 8],'idct2(x)');                 %Восстанавливаем изображение
imshow(f)
figure, imshow(f2)
```

Подробнее рассмотрим следующую строку программного кода:

*B2 = blkproc(B,[8 8],'P1.*x',mask);*

В данной строке *x* задает блок изображения. Данный символ является постоянным обозначением блока. Используя этот символ, блок может передаваться какой-либо функции в качестве параметра. *P1* – параметр, которым является переменная *mask*. Используемая функция, а также местоположение параметров в выражении всегда заключается в кавычки. Например, *'P1.*x'*, что означает, что на каждый блок коэффициентов будет наложена маска.

а) б)

Рисунок 2.4. – а) Оригинальное изображение; б) Сжатое изображение

Как видно из рисунка 2.4(б), после обнуления ряда коэффициентов качество изображения ухудшилось.

2.3 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.

2.3.1 Содержание отчета

- Титульный лист
- Тексты заданий
- Исходный код программы с комментариями
- Полученные результаты (рисунки) с подписями
- Выводы по проделанной работе

Контрольные вопросы

Что представляет собой сжатие изображений?
Какие методы сжатия изображений вы знаете?
Из каких основных этапов состоит JPEG-сжатие?
На каком этапе JPEG-сжатия происходит основная потеря информации?
Назовите недостатки JPEG-преобразования.
Что определяет RD-критерий?
В чем отличие сжатия цветных изображений от процедуры сжатия полутоновых изображений по стандарту JPEG?
Обнуление какого коэффициента ДКП вызовет самую сильную деградацию изображения?

Литература

1. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде Matlab, Москва: Техносфера, 2006. – 616с.
2. А.Ю. Тропченко, А.А. Тропченко Методы сжатия изображений, аудиосигналов и видео, Санкт-Петербург, 2009. – с.

3.Лабораторная работа № 3.
Восстановление изображений в системе Matlab

Цели лабораторной работы:

- Получение навыков формирования аддитивных шумов в системе Matlab
- Обучение восстановлению изображения после шумового воздействия

3.1. Зашумление изображений

В процессе фотосъемки некоторые изображения подвергаются шумовому воздействию. Поэтому обработка графических файлов неотделима от операции восстановления их приблизительного первоначального вида путем устранения шумов различного типа. Искажение изображения в общем виде представлено в виде формулы [1]:

$$g(x,y) = H[f(x,y)] + n(x,y) \quad (1)$$

где $g(x,y)$ – искаженное изображение, H – искажающий оператор, $n(x,y)$ – аддитивный шум.

Общая схема искажения/восстановления изображения представлена на рисунке 3.1 [1].

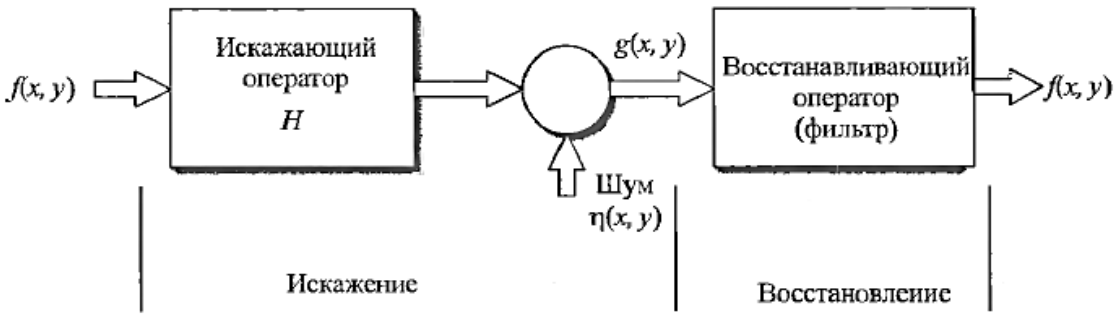


Рисунок 3.1. – Моделирование процесса искажения/восстановления изображения

Как видно из рисунка 3.1, искажение изображения осуществляется за счет искажающего оператора и аддитивного шума, восстановление изображения – за счет процедуры фильтрации.

Аддитивный шум подразделяется на два типа [1]:

- пространственный – характеризуется функцией плотности вероятности (PDF, Probability Density Function) или функцией кумулятивного распределения (CDF, Cumulative Distribution Function);
- частотный – описываемый различными частотными характеристиками преобразования Фурье.

В данной лабораторной работе искажения вносятся только пространственным шумом. В качестве графического файла выбрано изображение полутонового типа.

3.1.1 Формирование аддитивного пространственного шума

В пакете IPT системы Matlab добавление шума в изображение осуществляется с помощm. функции *imnoise,* имеющей вид:

$$g = imnoise(image, noiseType, parameters)$$

где *image* – исходное изображение, *noiseType* – тип шума, parameters – параметры шума, g – зашумленное изображение.

Функция *imnoise* поддерживает следующие типы шумов:

- Гауссов шум – задается параметром *‘gaussian’*
- Локальный гауссов шум - *‘localvar’*
- Шум Пуассона - ‘*poisson’*
- Шум типа соли и перца – *‘salt & pepper’*
- Спекл-шум (мультипликативный шум) – *‘speckle’*

Пример:

В данном примере исходное изображение подвергнем зашумлению шумом Гаусса с нулевым средним и дисперсией 0.1. Исходное изображение представлено на рисунке 3.2(а), зашумленное – на рисунке 3.2(б).

```
image = imread('cameraman.tif');
image = im2double(image);          %Все значения яркости изображения заключаются в диапазон [0; 1]
                                   %В данной работе подобная операция обязательна
noisedImage = imnoise(image, 'gaussian', 0, 0.1);
imshow(image), figure, imshow(noisedImage)
```

а б

Рисунок 3.2
а) Исходное изображение,
б) Зашумленное изображение

Функция *imnoise* удобна для зашумления изображения. Однако если нам нужна матрица самого шума, то его необходимо реализовывать самостоятельно.

Пространственный шум основан на использовании генератора случайных чисел. В Matlab встроены две функции, отвечающие за генерацию случайных чисел [1]:

- *rand* – генерирует матрицу равномерно распределенных случайных величин в интервале (0, 1).
- *randn* – генерирует матрицу с независимыми гауссовыми (нормальными) случайными величинами с нулевым средним и единичной дисперсией.

Функция *rand* имеет следующий синтаксис:

numbers = rand(M, N)

где *numbers* – матрица случайных чисел, *M* и *N* – размер матрицы. Если не указаны параметры *M* и *N*, то генерируется скалярная случайная величина. Если указана только одна сторона матрицы, то она считается квадратной.

Синтаксис функции *randn* аналогичен синтаксису функции *rand*:

numbers = randn(M, N)

Создание генераторов различных пространственных шумов основано на использовании данных функций.

В таблице 3.1 представлены виды пространственных шумов и генераторы, их реализующие.

Пример:

Рассмотрим реализацию гауссова шума.

```
noise = randn(256, 256);
imshow(noise)
```

В данном примере был создан гауссов шум в виде матрицы размером 256x256 элементов. Графическое изображение шума представлено на рисунке 3.3.

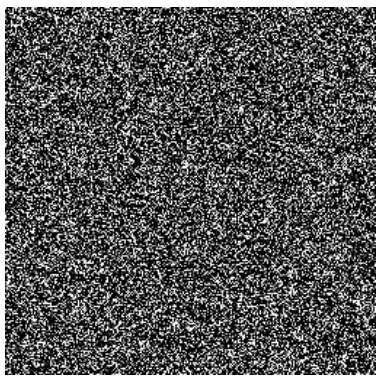


Рисунок 3.3. – Гауссов шум размером 256x256

3.2. Восстановление изображений

Восстановление изображений осуществляется при помощи фильтрации. В лабораторной работе № 2 вы уже использовали фильтрацию для выделения границ объектов изображения. В данной работе фильтрация используется для устранения шума. Напомним принцип работы фильтров [1]:

Фильтр представляет маску размером 3×3, которая в режиме построчного сканирования скользящим окном проходит по всем пикселям изображения. Данный процесс отображен на рисунке 3.4 [2]. Если матрица пикселов выходит за пределы изображения, то ее выходящие за пределы изображения пикселы считаются равными нулю (так называемый граничный эффект).

Полученная на каждом шагу матрица суммируется по представленной ниже формуле [2]:

$$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots + w(0,0)f(x,y) + \cdots + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1)$$

где R – область изображения, подвергнутая фильтрации.

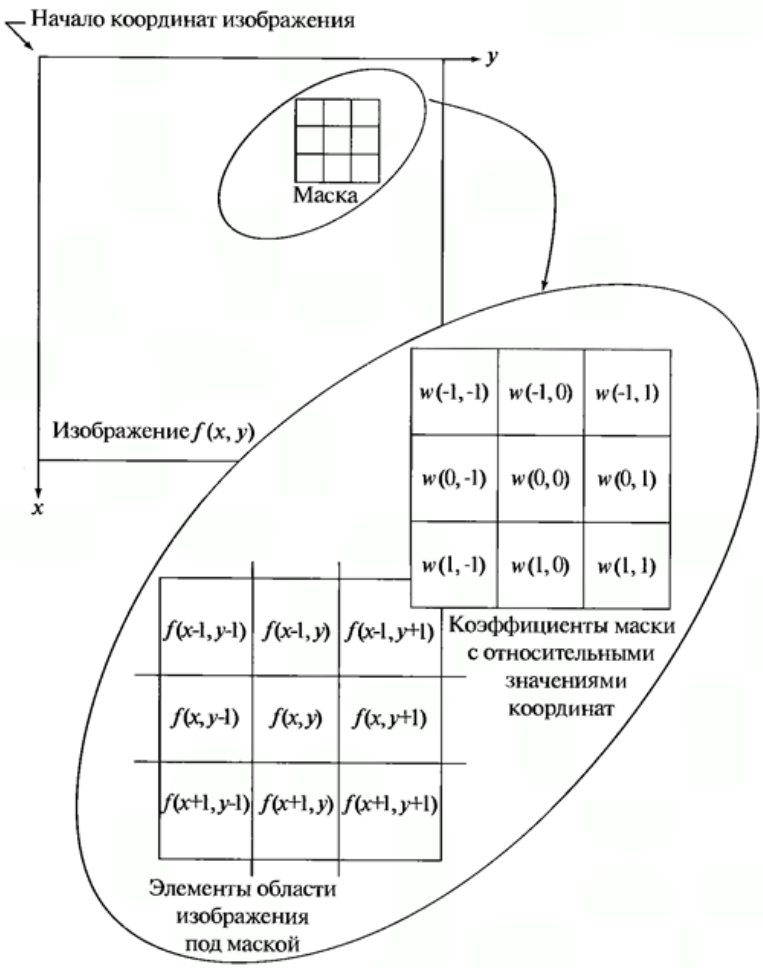


Рисунок 3.4. – Принцип работы фильтра изображения.

В таблице 3.2 представлены различные типы фильтров, предназначенных для фильтрации изображения. Каждый фильтр обладает различной мерой эффективности против конкретного вида шумового воздействия.

В системе Matlab встроены функции реализации фильтров, но только некоторых.

3.3. Задание

Задание № 1:

Используя данные таблицы 1, создайте функции, реализующие каждый из представленных видов шумов. Программный код функций с комментариями поместите в отчет.

Таблица 3.1.

Тип шума	Генератор шума (z)	Параметры для данной работы	
		a	b

Равномерный	функция rand	-	-
Гауссов	функция a + b*randn	0	0.15
Логарифмически нормальный	$z = a * e^{b*N(0,1)}$	1	0.25
Реллея	$z = \sqrt{b * ln * [1 - U(0,1)]}$	0	1
Экспоненциальный	$z = -\frac{1}{a} * \ln[1 - U(0,1)]$	1	-
Эрланга	z = $E_1 + E_2 + \ldots + E_b$, где $E_n = -\frac{1}{a} * \ln\left[1 - U(0,1)\right]$	2	5

N(0, 1) – нормальная Гауссова величина со средним 0 и дисперсией 1 (аналогична функции *randn*). *U(0, 1)* – равномерная случайная величина из интервала (0, 1) (аналогична функции *rand*).

Построим гистограмму шума Гаусса из предыдущего примера. Для создания гистограммы в системе Matlab предусмотрена функция *hist*, которая имеет следующий синтаксис [1]:

histogram = hist(r, bins)

где *histogram* – полученная гистограммы, *r* – исходный массив данных, *bins* – количество корзин гистограммы.

Таким образом, гистограмму данного шума можно построить следующим образом (рисунок 3.5)

Пример:

noise = randn(256, 256);

hist(noise(☺, 50)

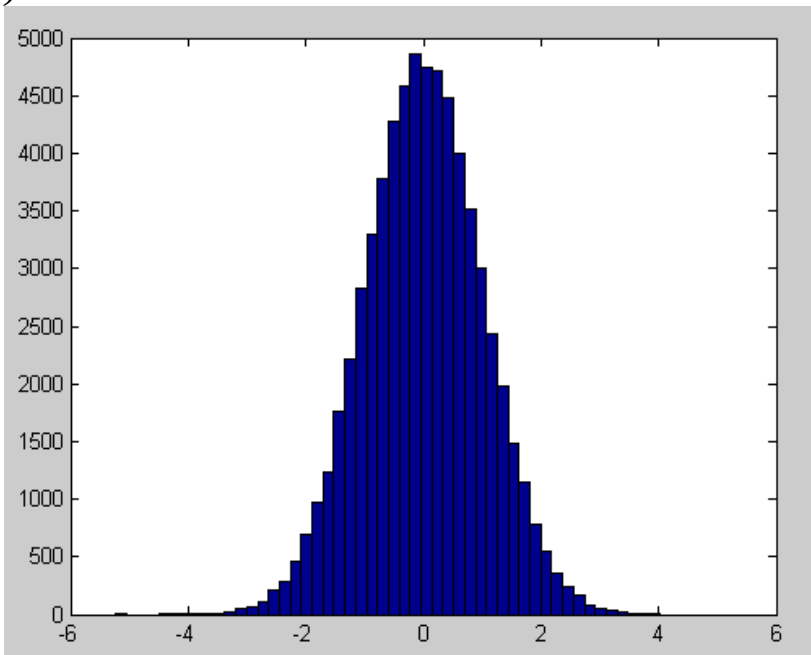


Рисунок 3.5. – Гистограмма Гауссова шума.

Задание № 2:

Осуществите генерацию каждого типа шума размером 256×256 при помощи созданных функций. Для всех шумов, кроме шума Соль и перец, создайте гистограмму и поместите их в отчет. Количество корзин гистограммы должно иметь значение, равное 50.

Используя матрицу шума, можно легко осуществить зашумление изображения. Зашумление осуществляется путем сложения матриц пикселов изображения и матрицы шумовых компонент:

$$noisedImg = image + noise$$

где *noisedImg* – зашумленное изображение, *image* – исходное изображение, *noise* – шум.

Пример:

```
image = imread('cameraman.tif');
image = im2double(image);
noise = randn(256);
noisedImg = image + noise;
imshow(noisedImg)
```

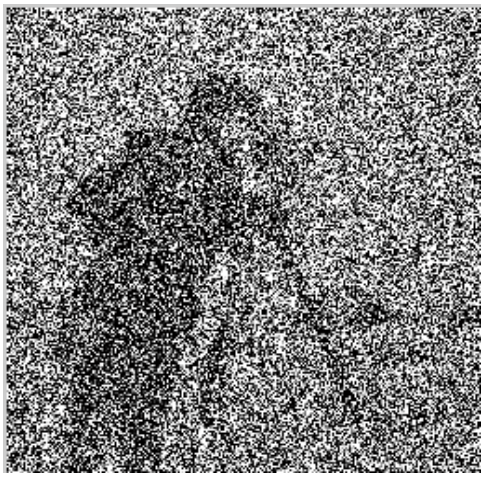


Рисунок 3.6. – Изображение, подвергнутое зашумлению шумом Гаусса

Задание № 3:

Используя созданные функции и полученное для данной лабораторной работы изображение, осуществите зашумление полученного изображения при помощи всех видов шумов, представленных в таблице 1. Каждый полученный файл сохраните в виде отдельного изображения. Полученные изображения для каждого вида шума поместите в отчет, указав при этом вид шума, которому изображение соответствует.

Задание № 4:

Создайте функции, реализующие фильтрацию изображения каждым фильтром, формула которого представлена в таблице 2. При реализации конкретного фильтра используйте только его формулу. Код программы с комментариями поместите в отчет.

В таблице 3.2 *n и m* – разрешение окна фильтрации. В качестве окна фильтрации используйте окно размером 3x3. *G* – исходное зашумленное изображение. *S* и *t* – координаты середины окна просмотра.

Таблица 3.3.

Имя фильтра	Уравнение
Арифметическое среднее	$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$
Геометрическое среднее	$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$
Гармоническое среднее	$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$
Контрагармонические средне	$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$
Медиана	$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\text{median}} \{g(s,t)\}$
Максимум	$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$
Минимум	$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$
Срединная точка	$\hat{f}(x,y) = \frac{1}{2} \left[\max_{(s,t)\in S_{xy}} \{g(s,t)\} + \min_{(s,t)\in S_{xy}} \{g(s,t)\} \right]$
α-усеченное среднее	$\hat{f}(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$

Задание № 5:

- Каждое изображение, полученное в задании 3, подвергните фильтрации всеми фильтрами, используя созданные в задании № 4 функции. Используя функцию корреляции Пирсона corr2, определите качество каждого восстановленного изображения, сравнив его с оригиналом, выданным для выполнения данной работы. Код программы с комментариями поместите в отчет.

- Выполняя данное задание, заполните таблицу 3.4 и поместите ее в отчет. В каждую ячейку, находящуюся на пересечении строки с типом шума и столбца с типом фильтра необходимо вписать значение корреляции между оригиналом и восстановленным изображением. **Жирным** выделить значение наилучшего показателя корреляции Пирсона.
- После заполнения таблицы 3.4 представьте десять вариантов восстановленных изображений и укажите, какой тип фильтра был при этом использован.

Таблица 3.4

	Равномерный	Гауссов	Логарифмически нормальный	Реллея	Экспоненциальный	Эрланга
Арифметическое среднее						
Геометрическое среднее						
Гармоническое среднее						
Контргармоническое среднее						
Медиана						
Максимумы						
Минимумы						
Срединная точка						
α-усеченное среднее						

Сделайте вывод об эффективности каждого фильтра для конкретного вида шума.

3.3.1. Содержание отчета:

- Титульный лист
- Тексты заданий
- Исходный текст программ с комментариями
- Полученные результаты (рисунки) с подписями
- Выводы по проделанной работе

Контрольные вопросы

В чем заключается процесс искажения/восстановления изображения?
На какие типы делятся шумы, вносимые в изображения?
На чем основаны пространственные фильтры?
Приведите примеры линейных пространственных фильтров.
Приведите примеры нелинейных пространственных фильтров.
Какие помехи и шумы устраняет медианный фильтр?
Поясните отличие взвешенной медианной фильтрации от просто медианной.
Каким образом осуществляется устранение зашумления изображения?

Литература

1. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде Matlab, Москва: Техносфера, 2006. – 616с.
2. http://habrahabr.ru/post/114452/

4. Лабораторная работа №4
Сокрытие данных в изображении-стегоконтейнере

Цели работы:

- получение навыков сокрытия данных в цифровых полутоновых изображениях методами стеганографии;
- получение навыков самостоятельного программирования алгоритмов обработки изображений в системе Matlab.

4.1. Основы стеганографии

Стеганография (в переводе с др. греческого - скрытая запись или тайнопись) – метод передачи данных, при котором от третьих лиц скрывается не только содержание передаваемой информации, но и сам факт передачи.

4.1.1.Метод сокрытия данных в младшем разрядном срезе стегоконтейнера

При таком способе передачи данных подлежащие скрытой передаче данные (текст, изображения, графика и т.д.) помещаются в некоторый блок открыто передаваемых данных. Такой блок носит название стегоконтейнера. При этом необходимо, чтобы объем данных в стегоконтейнере существенно превосходил количество данных в скрываемом сообщении.

На первом этапе (при передаче) подлежащие скрытой передаче данные помещают в контейнер, замещая часть имеющихся в нем данных данными скрываемого сообщения по определенному алгоритму. Смысл данных в самом контейнере после помещения в него скрытой информации не должен измениться по сравнению с исходным содержанием. Поэтому для третьих лиц сам факт передачи скрытого сообщения является неустановленным.

На втором этапе (при приеме) получатель изымает скрытую информацию из стегоконтейнера. Для этого необходимо знать алгоритм помещения данных в контейнер.

Рассмотрим один из методов такого типа.

В качестве стегоконтейнера используется изображение формата *.bmp, элементы которого (или пикселы) имеют разрядность 8 бит. Такое изображение содержит пикселы, яркость которых лежит в диапазоне от 0 до 255. Биты одного и того же двоичного разряда всех пикселов изображения носят название разрядного среза (очевидно, что их всего 8). Чувствительность человеческого глаза к восприятию пикселов небольшой яркости весьма мала и если уменьшить разрядность

пикселов изображения до семи или даже шести бит, то такое изображение будет восприниматься визуально как изображение с меньшей яркостью по сравнению с исходным. Если же затем увеличить вновь разрядность пикселов до 8 (при этом все младшие разряды всех пикселов станут нулевыми), при этом такое изображение визуально будет практически неотличимо от исходного. Тем самым на место разрядного среза исходного изображения можно поместить разрядный срез, содержащий скрываемую информацию. Такая информация может иметь практически любой характер - это может быть произвольный разрядный срез другого (скрываемого) изображения, блок численных данных, страницы текста и т.д.

Емкость такого стегоконтейнера (для скрываемой информации) составляет 1/8 от объема исходного изображения. Например, для изображения-контейнера размером 1024×1024 пикселов (1 Мбайт) можно разместить 128 Кбайт скрываемой информации (более 40 страниц печатного текста стандартного формата (40 строк по 60 символов) или один разрядный срез скрываемого изображения того же размера, что и контейнер).

Основные этапы подобной процедуры при передаче сводятся к следующему.

Для реализации метода используется система Matlab - пакет программ моделирования алгоритмов обработки изображений.

1. **Подготовка изображения-стегоконтейнера.**

Для этого требуется освободить младший разрядный срез какого-либо выбранного изображения невинного содержания.

Далее необходимо поделить яркость пикселов на 2, а затем вновь умножить на 2 . Затем подготовленный контейнер перезаписывается в новый буфер и сохраняется там.

2. **Подготовка помещаемой в контейнер скрытой информации.**

В качестве скрытой информации предлагается использовать разрядный срез (или некоторое бинарное) изображение. При этом размер такого изображения должен быть меньше, чем у стегоконтейнера (по каждой из координат).

Для этого необходимо произвести бинаризацию по порогу содержимого выбранного в качестве скрываемых данных изображения, задав порог бинаризации, далее полученное бинарное изображение разделить на 128 и оставить в новом буфере .

3. **Запись скрываемой информации**

Требуется сложить содержимое обоих буферов, содержащих подготовленный стегоконтейнер и скрываемый бинарный срез по опции пототочечной обработки, результат загрузить в буфер, где находился подготовленный стегоконтейнер (или сохранить под новым именем.

При приеме необходимо выделить из стегоконтейнера скрытую информацию, т.е. содержимое младшего разрядного среза контейнера.

4. **Восстановление изображения**

В буфер загрузить переданное изображение, а затем переписать в другой буфер. Разделить яркость пикселов изображения в новом буфере на 2, далее умножить на 2. Затем поменять местами содержимое буферов и из содержимого буфера со стегокнтенером вычесть содержимое другого буфера, далее умножить содержимое этого буфера А на 128 – и в нем оказывается видимым скрытое бинарное изображение.

Самостоятельно провести подобные манипуляции с несколькими различными изображениями. Привести вид исходного стегоконтейнера и гистограмму яркости пикселов подлежащего сокрытию бинарного изображения, стегоконтейнера после помещения в него скрываемой информации (с гистограммой) и вид восстановленного бинарного изображения (для каждого из изображений). Проанализировать недостатки и метода и вариант модификации для помещения скрываемой текстовой (символьной) информации.

4.1.2. Сокрытие данных в псевдо-белых и псевдо-черных пикселах изображения

В данном методе учитываются особенности восприятия яркости очень светлых и очень темных пикселов изображения зрительной системой человека.

Очень темные пикселы с яркостью в диапазоне от 0 до 15 воспринимаются как черные с уровнем 0, а очень яркие пикселы из диапазона от 240 до 255 воспринимаются как белые с яркостью 255. Таким пикселам для многоуровневых полутоновых изображений будут соответствовать двоичные коды:

- *0000xxxx* – для псевдо-черых пикселов
- *1111xxxx* – для псевдо-белых пикселов.

Здесь символ «*x*» означает произаольное значение каждого бита младшей тетрады.

Тем самым имеется возможность замены младшей тетрады на новое значение, соответствующее скрываемым данным – например, двоичному коду буквы скрываемого сообщения.

4.2. Задание

Задание №1.

1. Используя соответствующие функции пакета Matlab, напишите программу, реализующую алгоритм сокрытия данных в младшем разрядном срезе изображения.
2. Проведите эксперименты по сокрытию данных и их извлечению для различных изображений и типов скрываемых данных (бинарных изображений, данных типа .txt, .dat).
3. Определите количественные характеристики, определяющие соответствие между пустым и заполненным стегоконтйнером.

Задание №2

1. Самостоятельно реализуйте алгоритм сокрытия данных (текста) в псевдо-белых и псевдо-черных пикселах изображения и с использованием средств пакета Matlab напишите программу, реализующую данный метод.
2. Выполните п.п. 2 и 3 задания №1.
3. Сравните емкость стегоконтейнеров для обоих методов.
4. Самостоятельно провести подобные манипуляции с несколькими различными изображениями. Привести вид исходного стегоконтейнера и гистограмму яркости пикселов подлежащего сокрытию бинарного изображения, стегоконтейнера после помещения в него скраваемой информации (с гистограммой) и вид восстановленного бинарного изображения (для каждого из изображений). Проанализировать недостатки и достоинства методов и вариант модификации для помещения скрываемой текстовой (символьной) информации.

4.2.1. Содержание отчета

- Титульный лист
- Тексты заданий
- Блок схема- алгоритма сокрытия данных в псевдо-черных и псевдо-белых пикселах
- Исходный текст программ с комментариями
- Полученные результаты (рисунки) с подписями
- Графики – коэффициента корреляции для пустого и заполненного стегоконтейнеров
- Выводы по проделанной работе

Контрольные вопросы

В чем отличие стеганографии от криптографии?
Что понимается под разрядным срезом изображения?
Что понимается под стегоконтейнером?
В чем отличие пустого стегоконтейнера от заполненного?
Укажите основную задачу стеганографии.
Каким образом осуществляется сокрытие данных в псевдо-белых и псевдо-черных пикселях изображения?

Литература

1. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде Matlab, Москва: Техносфера, 2006. – 616с.
2. http://habrahabr.ru/post/114452/

Заключение

В данном учебно-методическом пособии были рассмотрены основные приёмы решения различных прикладных задач обработки цифровых изображений с использованием пакета программ MATLAB. Лабораторные работы представляют собой реальные практические задания для эффективной работы с информацией и данными.

Успешное выполнение лабораторных и качественная работа при ответе на вопросы для самостоятельной подготовки позволит закрепить умение теоретически и практически решать в дальнейшем задачи по обработке, распознаванию сигналов и изображений с использованием пакета MATLAB в ходе самостоятельных исследований.

В тоже время только постоянное использование полученных умений и навыков позволит стать действительно высококлассным специалистом в области информационных технологий и вычислительной техники.

УНИВЕРСИТЕТ ИТМО

Миссия университета – открывать возможности для гармоничного развития конкурентоспособной личности и вдохновлять на решение глобальных задач

Тропченко Александр Ювенальевич
Тропченко Андрей Александрович

Обработка изображений в системе MATLAB: лабораторные работы
учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н. Ф. Гусарова
Подписано к печати
Заказ №
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверский пр., 49