Лабораторная работа № 5

ОБРАБОТКА ИЗОБРАЖЕНИЙ В GNU OCTAVE (MATLAB)

Цель работы: ознакомление с средствами и методическими приемами анализа дистанционных изображений в средах научного программирования.

Задача: провести анализ изображений в соответствии с вариантом. Выполнить детальные задания в конце файла. Подготовить содержательный отчет, сделать выводы.

Примечание. Для выполнения настоящей работы используется бесплатная программа GNU Octave (http://www.octave.org), имеющая существенную совместимость с Matlab. Для работы программы обработки изображений в Octave, требуется скачать модуль *image* (по ссылке), а затем установить его с помощью консольной команды Octave:

```
pkg install C:\my_folder\image-2.4.1.tar.gz
```

Здесь `C:\my_folder\` – папка, в которой находится скачанный архив с модулем image, `image-2.4.1.tar.gz` – имя файла архива.

После установки необходимо перезапустить программу. Загрузка модулей не автоматизирована и требует ввода консольной команды:

```
pkg load all
```

Внимание! Некоторые коды примеров содержат небольшие неточности, препятствующие их корректному выполнению. Это означает, что необходимо найти и исправить «неточность», что требует понимания работы кода программы. Разумеется, предоставляемый по итогам работы отчет должен содержать корректный код программы, без ошибок, с которыми программа работать не сможет.

5.1. Представление об изображениях

В информатике изображения рассматриваются, как прямоугольные матрицы, каждый элемент которых соответствует яркости пикселя изображения (“pixel” от англ. “picture element”). Открытие изображений в средах научного программирования позволяют применять к ним средства, доступные для анализа матриц. Представление матрицы в виде изображения показано на рисунке (рис. 1).

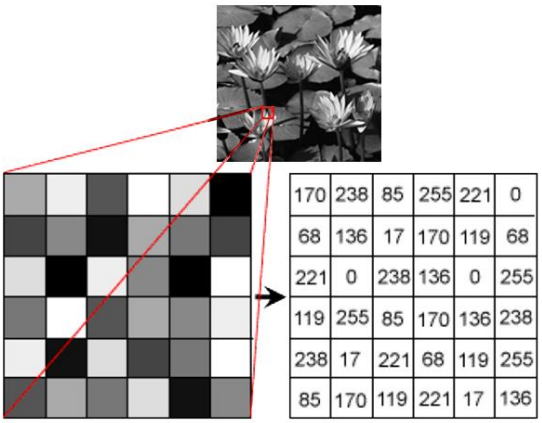

Рис. 5.1. *Представление фрагмента серого изображения в виде матрицы*

При этом следует помнить, что «серые», одноканальные изображения

представляют собой одну матрицу, тогда как изображения «цветные» представляют собой три канала R,G,B – красный (λ=625—740), зеленый (λ=500—565), голубой (λ=485—500), т.е. представляются тремя матрицами. Это нужно учитывать при открытии цветного изображения.

5.2. Фильтры изображений в Octave (Matlab)

Для ознакомления с методами обработки изображений в Octave рассмотрим следующий пример. Для занятия необходимо использовать изображения в папке files архива лабораторной работы.

Внимание! Для корректной работы Octave имя файла и путь к нему не должны содержать кириллических символов.

Пример 1. *Откроем изображение lily.jpg из архива.*

```
%открытие файла изображения в матрицу A
My_lily= imread('lily.jpg');

%посмотрим картинку
imshow(My_lily);
```

Обратим внимание, что изображение содержит три канала, а значит открытие создаст трехмерную матрицу. Для изучения переменных и объектов, находящихся в памяти, можно использовать окно «Область переменных» Octave или «Workspace» Matlab.

Наряду с этим, свойства переменной можно узнать с помощью команды whos VAR_NAME. Это выдаст нам свойства переменной VAR_NAME. Узнаем свойства матрицы my_lily.

```
%узнаем свойства переменной My_lily
whos My_lily

Variables in the current scope:

   Attr Name        Size                     Bytes    Class
   ==== ====        ====                     =====    =====
        My_lily     600x800x3                1440000  uint8
```

Что мы узнали? *Переменная My_lily представляет собой трехмерный массив размерностью 600x800 (*`600x800x3`*), занимающий в памяти 1440000 байт, элементы массива представляют собой данные класса* `uint8` *(8-ми битное целое). В элемент класса uint8 можно записать данные от 0 до* 2^8*=255, т.е. всего 256 значений. Ясно, что этот тип данных наиболее приемлем для хранения в них матриц изображений.*

Octave (Matlab) доступны любые методы, применимые для преобразования изображений и, зачастую, используемые в графических редакторах, такие как обрезка, масштабирование, фильтры и т.д. Однако, неоспоримым достоинством Octave является воспроизводимость и контроль действий.

Пример 2. *Масштабирование изображения lily.jpg и сохранение файла с меньшим разрешением. Кадрирование.*

Для масштабирование используется функция new_image_mat=imresize(image_matrix, scale). Функция imresize использует два параметра. В качестве первого, image_matrix, используется исходное изображение, в качестве второго scale, используется масштабный коэффициент. Если он меньше 1, изображение уменьшается, если больше, увеличивается.

Масштабированную матрицу изображения new_image_mat можно пересохранить, используя функцию, imwrite(new_image_mat, filepath). Здесь new_image_mat – матрица, которую требуется сохранить в изображение, filepath – полный путь к файлу, включая имя[1].

```
%открытие файла изображения в матрицу My_lily
My_lily= imread('lily.jpg');

%масштабирование на 50%
My_lily_small= imrsize(My_lily, 0,5);

%посмотрим картинку после масштабирования
imshow(My_lily_small);
title('После уменьшения на 50%');

%путь к файлу и его имя в переменной
filepath=('C:\my_folder\lily_sm.jpg');

%сохраняем уменьшенное на 50% изображение в файл
imwrite(My_lily_small, filepath);
```

В результате выполнения кода, в папке `C:\my_folder` *будет сохранено новое изображение lily_sm.jpg, ширина и высота которого уменьшены на 50% по сравнению с исходным изображением.*

Изображение иногда приходится кадрировать, т.е. выбрать фрагмент исходной матрицы, удалив все лишнее. Код ниже показывает, как это может быть сделано:

```
%открытие файла изображения в матрицу My_lily
My_lily= imread('lily.jpg');

%исходная картинка
figure;
imshow(My_lily);
title('Исходное изображение');

%кадрирование одного из цветков
My_lily_crop= My_lily(140:285,433:617,:);

%посмотрим кадрированный фрагмент
figure;
imshow(My_lily_cro);
title('Кадрированный фрагмент');

%путь к файлу и его имя в переменной
filepath=('C:\my_folder\lily_crop.jpg');

%сохраняем уменьшенное на 50% изображение в файл
imwrite(My_lily_crop, filepath);
```

Самостоятельно: *почему кадрирование в коде выше выполняется как*

```
My_lily_crop= My_lily(140:285,433:617,:);
```

а не как:

```
My_lily_crop= My_lily(140:285,433:617);
```

Проверить, отразить в отчете и обосновать свой ответ.

Для проведения анализа изображение часто должно пройти этапы

[1] О способах применения функции imwrite ( ), дополнительных параметрах и способах прменения следует читать официальную справку к функции

обработки, направленные на его размытие или повышение резкости, распознавание границ, а также преобразование изображения (например, из цветного в оттенки серого или в двухцветное, черно-белое). Это может быть достигнуто в Octave (Matlab) с помощью пользовательских или предустановленных функций. Достижение целого ряда графических эффектов возможно с помощью применения так называемой *свертки изображения* – применение матрицы коэффициентов (матрицы свертки), которая «умножается» на значение пикселей изображения для получения требуемого результата.

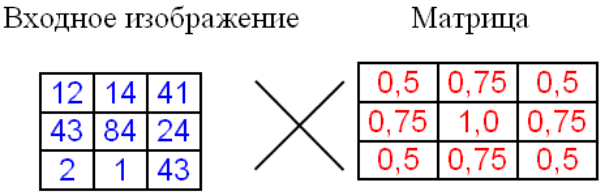

Рис. 5.2. *Свертка изображения по матрице коэффициентов*

Например, для создания фильтра размытия (blur) требуется выполнить свертку изображение по «матрице Гаусса» – матрице, в которой элементы распределены по нормальному закону (распределение Гаусса). Чтобы создать такую матрицу, может быть использована функция fspecial().

```
>> B=fspecial('Gaussian',5,5)
B =
    0.0369    0.0392    0.0400    0.0392    0.0369
    0.0392    0.0416    0.0424    0.0416    0.0392
    0.0400    0.0424    0.0433    0.0424    0.0400
    0.0392    0.0416    0.0424    0.0416    0.0392
    0.0369    0.0392    0.0400    0.0392    0.0369
```

Код выше показывает создание «гауссовой матрицы». Пространственная сетка для этой матрицы (создается вызовом функции *mesh()*) представлена (рис. 5.3).

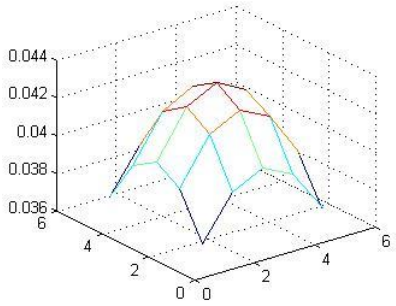

Рис. 5.3. Пространственная сетка «гауссовой матрицы»

Необходимо знать, что свертка матриц (конволюция) может быть выполнена только для одноканальной матрицы – серого изображения. Что же делать, если матрица 3-х канальная, как в случае нашей цветной картинки? Логично предположить, что надлежит сначала разложить матрицу по отдельным каналам, применить к ней фильтр, а затем снова совместить каналы в цветное изображение.

Пример 3. *Разложение цветного изображения по отдельным каналам и вывод его в совмещенном графическом окне (subplot).*

```
A=imread('lily.jpg');

%выбор в отдельные матрицы каналы изображения
%красный
A_R=A(:,:,3);
%зеленый
A_G=A(:,:,2);
%голубой
A_B=A(:,:,1);

figure;
%создание новой картинки
subplot(2,2,1); %значит, что создается изображение 2x2, текущий
сегмент 1
imshow(A);
title('RGB');
subplot(2,2,2); %значит, что создается изображение 2x2, текущий
сегмент 2
imshow(A_B);
title('Blue channel');
subplot(2,2,3); %значит, что создается изображение 2x2, текущий
сегмент 3
imshow(A_G);
title('Green channel');
subplot(2,2,4); %значит, что создается изображение 2x2, текущий
сегмент 4
imshow(A_R);
title('Red channel');

saveas(gcf,'MyFigure.png'); %сохраним картинку в файл png

figure; %create new figure
%combine channels into older picture - nothing been changed
comb_lily(:,:,1)=A_R; comb_lily(:,:,2)=A_G; comb_lily(:,:,3)=A_B;
comb_lily=uint8(comb_lily);
imshow(comb_lily);
title('lily combined from separated channels');
```

Самостоятельно: *детально разобрать код выше и понять как он работает.* *Результат вывода каналов изображения показан на рис. 5.4.*

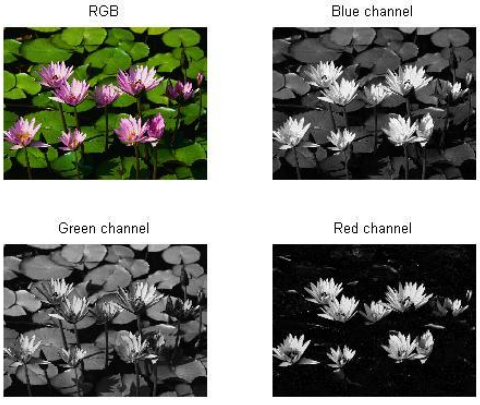

Рис. 5.4. Совмещенная графика *subplot*

Пример 4. *Графические фильтры изображения: размытие. Рассмотрим применение фильтра размытия по гауссовой матрице для серого изображения.*

```
%открытие файла изображения в матрицу A
My_lily= imread('lily.jpg');

%посмотрим картинку
imshow(My_lily);

%преобразуем картинку в оттенки серого
My_gray_lily=rgb2gray(My_lily);

%отобразим серую картинку в новом окне
figure;
imshow(My_gray_lily);
title('Изображение в оттенках серого';

%преобразуем серое изображение в формат с плавающей запятой
%из uint8
My_gray_lily_d=im2double(My_gray_lily);

%создаем «гауссову матрицу»
B=fspecial('Gaussian',5,5);

%свертка по матрице B
My_gray_lily_blur=conv2(My_gray_lily_d,B);

%вывод размытого изображения
figure;
imshow(My_gray_lily_blur);
title('Применен фильтр blur');
```

Пример 5. *Графические фильтры изображения: усиление резкости (sharpen).*

```
%открытие файла изображения в матрицу A
My_lily= imread('lily.jpg');

%посмотрим картинку
imshow(My_lily);

%выберем только красный канал изображения
My_lily_red= My_lily(:,:,3);

%отобразим серую картинку в новом окне
figure;
imshow(My_lily_red);
title('Красный канал исследуемого фото');

%преобразуем серое изображение красного канала в формат с
%плавающей запятой
%из uint8
My_lily_rd=im2double(My_lily_red);

%создаем матрицу повышения резкости
S=[-1 -1 -1; -1 9 -1; -1 -1 -1];

%свертка по матрице B
Lily_rd_blur=conv2(My_lily_rd,S);

%вывод размытого изображения
figure;
imshow(Lily_rd_blur);
title('Применен фильтр sharpen');
```

5.3. Подстройка изображений в Octave (Matlab)

Иногда необходимо выполнить коррекцию изображения, сделать его светлее или темнее. Для того, чтобы это сделать в Octave/Matlab необходимо помнить, что матрица изображения содержит значения его спектральной яркости. Это значит, что коррекция изображения может означать поэлементное умножение матрицы на некоторый коэффициент n. Для того, чтобы сделать изображение темнее, надо умножить на коэффициент n<1, для того, чтобы осветлить изображение, коэффициент должен быть n>1.

```
%открытие файла изображения в матрицу A
My_lily= imread('lily.jpg');

%преобразуем в оттенки серого
My_lily=rgb2gray(My_lily);
%коэффициенты осветления и затемнения
n_d=0.5; n_l=1.8;

%затемним изображение
Ml_dark=My_lily*n_d;

%осветлим изображение
Ml_lighter=My_lily*n_l;

%вывод изображений и их гистрограмм
subplot(2,3,1); imshow(Ml_dark);
title('If darker');
subplot(2,3,2);
imshow(My_lily);
title('Initial image');
subplot(2,3,3);
imshow(Ml_lighter);
title('If lighter');

subplot(2,3,4); imhist(Ml_dark);
title('If darker');
subplot(2,3,5);
imhist(My_lily);
title('Initial image');
subplot(2,3,6);
imhist(Ml_lighter);
title('If lighter');
```

Рекомендации. В Matlab/Octave для вывода строковых сообщений используются следующие функции. Преобразование числа в строку, выполняемое в Scilab функцией *string(Число),* в Matlab/Octave требует применения функции *num2str(Число)*. Объединение строк, осуществляемое в Scilab оператором сложения, в Matlab/Octave требует strcat(Строка1, Строка2, …Строка n).

Существуют и другие несовместимости. При возникновении технических затруднений, необходимо читать справку по функциям, применяемым в Matlab/GNU Octave.

Прервать выполнение программы можно, нажав в командном окне комбинацию клавиш «Ctrl+C».

Внимание! Начало координат изображения в Matlab/Scilab/Octave – левый верхний угол. Столбцы и строки графического изображения развернуты на 90 относительно графиков и диаграмм.

Задание для самостоятельной работы

- Исправить коды примеров Matlab, найдя ошибки, препятствующие их нормальной работе. Указать в отчете, что и где было исправлено;
- Составить глоссарий применяемых в уроке функций Matlab, с кратким описанием;
- Прорешать приводимые примеры, с внимательным изучением комментариев в коде, выполнить самостоятельные задачи;
- Основываясь на полученном опыте и литературном материале, выполнить самостоятельные задания;
- В таблице вариантов найти координаты кадрирования исходного изображения. Кадрированное изображение, фрагмент файла *lily.jpg* будет использоваться для обработки в рамках своего варианта;
- Для кадрированного ЦВЕТНОГО изображения выполнить фильтры *blur* и *sharpen*;
- На совмещенной графике *subplot* размером 4x4 клетки показать (1) исходное изображение фрагмента, (2) изображение в оттенках серого, (3) цветное изображение с фильтром blur, (4) цветное изображение с фильтром *sharpen*. Сохранить совмещенную графику в файл в виде изображения и заем вставить его в отчет;
- Для анализируемого фрагмента сделать два преобразования – затемнить и осветлить, для чего составить программу. По результатам выполнения этой программы вывести затемненное и осветленное серые изображения фрагмента и их гистограммы.

Требования к отчету

Отчет о выполнении лабораторной работы должен содержать:

- тему и цель лабораторной работы на титульной странице;
- автособираемое оглавление, с номерам страниц;
- необходимые теоретические сведения по теме решаемых задач, вставка рисунков из методички должна быть обоснованной;
- исходные данные и последовательность их обработки (преобразования);
- поэтапный расчет в виде вставок кода Scilab/Matlab в объектах «Подпись»;
- результаты выполнения кода Octave/Matlab в виде вывода в командном окне;
- графики, выполненные в Octave/Matlab, оформленные как рисунки, после ссылок на них в тексте отчета;
- интерпретацию результатов работы моделей;
- отметку преподавателя о выполнении лабораторной работы.

Рекомендуемая литература

1. Поршнев С. В. Компьютерное моделирование физических процессов в пакете MATLAB / М.:, 2003. 593 с.
2. Алексеев Е.Р. , Чеснокова О.В. Введение в Octave для инженеров и математиков: / Е.Р. Алексеев, О.В.Чеснокова М.: ALT Linux, 2012. 368 с.
3. Материалы по продуктам MATLAB & Toolboxes // [Электронный ресурс]: Математический сайт Exponenta.ru. Веб-сайт. URL: http://matlab.exponenta.ru/index.php (Дата обращения: 05.11.2015)

Варианты исходных данных для решения самостоятельной задачи

Указаны координаты фрагмента для кадрирования, верхняя Y1 и нижняя Y2, левая X1 и правая X2

Вариант	Файл изображения	Координаты фрагмента для кадрирования			
		Y1	Y2	X1	X2
1	lily.jpg	134	256	125	285
2	lily.jpg	213	314	182	348
3	lily.jpg	220	324	350	467
4	lily.jpg	146	288	450	618
5	lily.jpg	206	307	738	764
6	lily.jpg	171	246	740	800
7	lily.jpg	116	190	463	587
8	lily.jpg	185	213	179	242
9	lily.jpg	342	502	12	148
10	lily.jpg	390	506	209	354
11	lily.jpg	337	485	430	592
12	lily.jpg	405	442	479	549
13	lily.jpg	209	257	396	502
14	lily.jpg	182	236	170	258
15	campus.jpg	191	272	196	290
16	campus.jpg	251	391	352	441
17	campus.jpg	468	557	229	424
18	campus.jpg	407	474	330	457
19	campus.jpg	285	464	169	318
20	campus.jpg	468	560	318	470
21	campus.jpg	47	153	83	219
22	campus.jpg	225	374	12	130
23	campus.jpg	178	317	326	456
24	campus.jpg	285	390	133	287
25	arduino.jpg	253	390	288	445
26	arduino.jpg	292	406	50	210
27	arduino.jpg	161	333	531	640
28	arduino.jpg	166	282	424	546
29	arduino.jpg	336	450	215	358
30	arduino.jpg	71	178	328	447
31	arduino.jpg	159	253	50	165
32	arduino.jpg	345	476	525	642
33	arduino.jpg	33	120	535	646