МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«Витебский государственный технологический университет»

Основы систем технического зрения

Методические указания
по выполнению лабораторных работ
для студентов специальности
1-55 01 03 «Компьютерная мехатроника»

Витебск
2024

УДК 681.5

Составитель А. Г. Кириллов

Одобрено кафедрой «Автоматизация производственных процессов» УО «ВГТУ», протокол № 7 от 25.01.2024.

Рекомендовано к изданию редакционно-издательским советом УО «ВГТУ», протокол № 7 от 03.04.2024.

Основы систем технического зрения: методические указания по выполнению лабораторных работ / сост. А. Г. Кириллов. – Витебск : УО «ВГТУ», 2024. –47с.

Методические указания являются руководством к выполнению лабораторных работ по дисциплине «Основы систем технического зрения» для студентов специальности 1-55 01 03 «Компьютерная мехатроника», содержат перечень лабораторных работ, освещают теоретические вопросы подготовки по их выполнению, приводят примеры разработки программ для обработки изображений в системах технического зрения.

УДК 681.5

Содержание

Введение

Основная цель лабораторных работ по дисциплине «Основы систем технического зрения» – это развитие инженерных навыков по разработке систем технического зрения на основе современных технических средств и программного обеспечения.

Системы технического зрения – это методы и технологии, которые позволяют компьютеру анализировать изображения с целью автоматической обработки данных. Лабораторные работы по основам систем технического зрения представляют собой набор практических задач, направленных на ознакомление студентов с основными методами обработки изображений и распознавания образов.

В данном методическом пособии используется среда MATLAB в качестве инструмента для анализа и обработки данных, включая изображения. MATLAB предоставляет широкий спектр функций, которые упрощают работу с изображениями и обеспечивают быструю реализацию различных алгоритмов компьютерного зрения.

Лабораторные работы включают в себя такие темы, как геометрические преобразования, фильтрация изображений, сегментация объектов. Каждая лабораторная работа содержит пошаговые инструкции по выполнению заданий, а также примеры кода на MATLAB для реализации необходимых операций.

Лабораторная работа 1
Введение в MATLAB

Изначально MATLAB разрабатывался как диалоговая среда для матричных вычислений (MATrix LABoratory). Со временем пакет был оснащен хорошей графической системой, дополнен средствами компьютерной алгебры от Maple и усилен библиотеками команд (или Toolboxes), предназначенными для эффективной работы со специальными классами задач.

В состав MATLAB входят интерпретатор команд, графическая оболочка, редактор-отладчик, библиотеки команд, компилятор, символьное ядро пакета Maple для проведения аналитических вычислений, математические библиотеки MATLAB на C/C++, генератор отчетов и инструментарий (Toolboxes).

Интерфейс MATLAB многооконный и имеет ряд средств прямого доступа к различным компонентам системы.

Главное меню – посредством этого меню осуществляются наиболее общие действия системы MATLAB. Меню имеет стандартный вид и организацию, присущую многим программным продуктам (рис. 1).

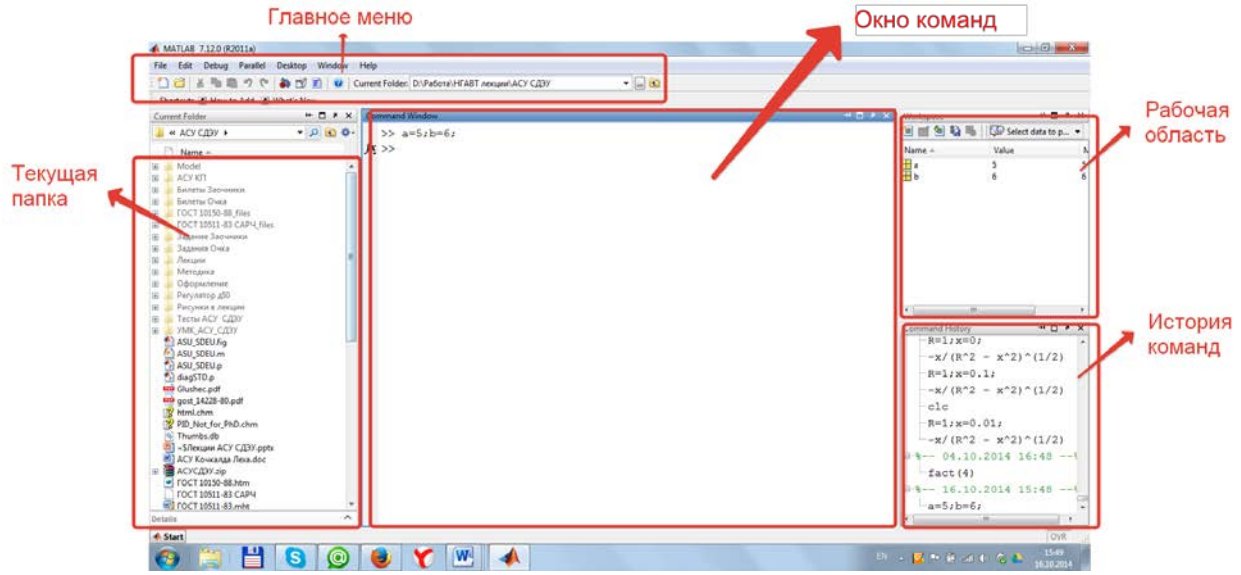

Рисунок 1 – Интерфейс системы MATLAB

Окно команд (Command Window) – наиболее важное окно для пользователя. Посредством этого окна вводятся математические выражения, получают результаты вычислений, а также выдаются сообщения, посылаемые системой. Данное окно становится доступным пользователю сразу же после запуска программы. Математические выражения пишутся в командной строке после знака приглашения >>.

Например:

```
>>X = 2 + 3
```

Для получения результата необходимо нажать клавишу «Enter».

Если необходимо изменить одну цифру, то это не получится сделать, установив курсор в уже введенное выражение и провести редактирование. Невозможность редактирования ранее введенной команды простой установкой курсора в нужную строку является одной из особенностей программы MATLAB. Для редактирования ранее введенной команды, необходимо установить курсор в строку ввода и воспользоваться клавишами «↑» и «↓». Эти клавиши позволяют пролистать историю введенных ранее команд и оставить в строке ту команду, которая необходима. Команду можно выполнить сразу (нажав клавишу «Enter») или после редактирования.

Текущая папка (Current Folder) – является аналогом известной программы Проводник, но имеет для MATLAB свое предназначение. Дело в том, что, кроме работы с математическими выражениями из командного окна, пользователь также может работать с файлами. Математические функции, используемые в процессе работы, физически представляют собой файлы, названные по именам функций. В этих файлах записаны программы, реализующие функции. Таким образом, в работе постоянно используются файлы. Например, указывая встроенную функцию, фактически указывается имя файла без расширения, в котором хранится текст программы. Этот файл программа будет искать в текущей папке или в путях, прописываемых отдельно.

Рабочая область (Workspace) – содержит список всех переменных, хранящихся в рабочем пространстве. В процессе работы используются переменные различных типов. Созданные переменные хранятся в специально отведенной области памяти компьютера. Они не исчезают сами по себе, а только при выходе из программы или путем использования специальных команд. При этом значения переменных можно использовать в любом вводимом математическом выражении. С помощью окна «Рабочая область» можно выбрать любую переменную, посмотреть ее содержимое или выполнить другие действия.

История команд (Command History) – содержит список команд, вводимых в командной строке «Окна команд». При необходимости повторить ранее выполненную команду, нужно отыскать в списке «История команд» эту команду и, двойным щелчком левой кнопки мыши, можно выполнить команду. Содержимое этого окна не теряется после выхода из системы и выключения компьютера. Удалить список команд можно только с помощью меню.

Все расчеты в MATLAB выполняются с двойной точностью, а для представления чисел на экране имеются разные форматы. Нужный формат может быть определен в меню (**File/Preferences**) либо при помощи команды **format**. Существуют следующие способы представления чисел (табл. 1).

Переменные в MATLAB не нужно предварительно описывать, указывая их тип. Все данные хранятся в виде массивов: числовые переменные (внутренний тип *numeric*), текстовые строки (*char*), ячейки (*cell*) и структуры (*struct*). Двумерный массив – это матрица, одномерный – вектор, а скаляр – матрица размера 1x1. Имя переменной должно начинаться с буквы, за ней могут идти

буквы, цифры и символ подчеркивания. Допустимы имена любой длины, но MATLAB идентифицирует их по первым 31 символам и различает большие и малые буквы.

Таблица 1 – Форматы вывода на экран

Формат	Представление
short	Число отображается с 4 цифрами после десятичной точки или в формате short e
short e	Число в экспоненциальной форме с мантиссой из 5 цифр и показателем из 3 цифр
rat	Представление в виде рационального дробного числа
long	Число с 16 десятичными цифрами
long e	Число в экспоненциальной форме с мантиссой из 16 цифр и показателем из 3 цифр
hex	Число в шестнадцатеричной форме

В MATLAB имеется ряд встроенных констант (табл. 2) и специальных символов (табл. 3).

Таблица 2 – Зарезервированные имена констант

Имя	Описание
ans	Результат последней операции
pi	Число π
eps	Машинная точность
realmax	Максимальное вещественное число
realmin	Минимальное вещественное число
inf	Бесконечность
NaN	Нечисловая переменная (зарезервировано для результата операций 0/0, 0*inf, inf-inf и т. п.)
end	Наибольшее значение индекса размерности массива

В командном окне в режиме диалога проводятся вычисления. Пользователь вводит команды или запускает на выполнение файлы с текстами на языке MATLAB. Интерпретатор обрабатывает введенное значение и выдает результаты: числовые и строковые данные, предупреждения и сообщения об ошибках.

Таблица 3 – Специальные символы

Символ	Назначение
1	2
[]	Квадратные скобки используются при задании матриц и векторов
	Пробел служит для разделения элементов матриц
,	Запятая применяется для разделения элементов матриц и оператора в строке ввода

Окончание таблицы 3

1	2
;	Точка с запятой отделяет строки матриц, а точка с запятой в конце оператора (команды) отменяет вывод результата на экран
:	Двоеточие используется для указания диапазона (интервала изменения величины) и в качестве знака групповой операции над элементами матриц
()	Круглые скобки применяются для задания порядка выполнения математических операций, а также для указания аргументов функций и индексов матриц
.	Точка отделяет дробную часть числа от целой его части, а также применяется в составе комбинированных знаков (.*, .^, ./, .\)
…	Три точки и более в конце строки отмечают продолжение выражения на следующей строчке
%	Знак процента означает начало комментария
’	Апостроф указывает на символьные строки, а для включения самого апострофа в символьную строку нужно поставить два апострофа подряд

Имена переменных должны начинаться с буквы. Знак «=» соответствует операции присваивания. Нажатие клавиши *Enter* заставляет систему вычислить выражение и показать результат. Если запись оператора не заканчивается символом «;», то результат выводится в командное окно, в противном случае – не выводится. Если оператор не содержит знака присваивания «=», то значение результата присваивается системной переменной *ans*.

Для просмотра значения любой переменной из текущего рабочего пространства системы достаточно набрать ее имя и нажать клавишу *Enter*.

После окончания сеанса работы с системой MATLAB все ранее вычисленные переменные теряются. Чтобы сохранить в файле на диске компьютера содержимое рабочего пространства системы MATLAB, нужно выполнить команду меню **File –> Save Workspace As…** По умолчанию расширение имени файла **mat**, поэтому такие файлы принято называть MAT-файлами.

В арифметических выражениях применяются следующие знаки операций:

+, - – сложение, вычитание;

* – умножение;

/ – деление слева направо;

\ – деление справа налево;

^ – возведение в степень.

Система MATLAB позволяет вычислять различные математические функции. Следующие элементарные алгебраические функции имеют в качестве аргумента одно или два действительных (x, y) числа (табл. 4).

Таблица 4 – Элементарные алгебраические функции

Функция	Описание
sqrt(*x*)	Вычисление квадратного корня
round(*x*)	Округление до целого.
fix(*x*)	Округление до ближайшего целого в сторону нуля.
rem(*x*, *y*)	Вычисление остатка от деления x на y.
exp(*x*)	Вычисление e в степени x.
log(*x*)	Вычисление натурального логарифма числа x.
log10(*x*)	Вычисление десятичного логарифма числа x.

Система MATLAB предоставляет возможности для вычисления следующих тригонометрических и обратных тригонометрических функций переменной *x* (табл. 5).

Таблица 5 – Тригонометрические функции

Функция	Описание
sin(*x*)	Вычисление синуса
cos(*x*)	Вычисление косинуса
tan(*x*)	Вычисление тангенса
asin(*x*)	Вычисление арксинуса
acos(*x*)	Вычисление арккосинуса
atan(*x*)	Вычисление арктангенса
atan2(*y*, *x*)	Вычисление арктангенса по координатам точки

Лабораторная работа 2
Типы изображений и их представление в MATLAB

Изображения бывают *векторными* и *растровыми*.

Векторным называется изображение, описанное в виде набора графических примитивов.

Растровые же изображения представляют собой двумерный массив, элементы которого (пиксели) содержат информацию об их яркости и цвете. Обычно элементом изображения называется минимальная деталь изображения, внутри которой яркость и цвет считаются постоянными, т. е. внутри элемента неравномерность яркости и цвета уже не будут различаться глазом. В этом случае информация об изображении представляется в виде значений характеристик каждой точки изображения – пикселя (pixel – ***pic****ture* ***el****ement*) – наименьшей структурной единицы изображения. В качестве таких характеристик выступают яркость и цвет пикселя, а его порядковый номер в кадре дает информацию о его координате на экране.

Пиксель – наименьший логический элемент двумерного цифрового изображения в растровой графике. Пиксель представляет собой неделимый

объект прямоугольной, обычно квадратной, или круглой формы, обладающий определённым цветом и яркостью. Растровое компьютерное изображение состоит из пикселей, расположенных по строкам и столбцам. В цифровой обработке используются обычно растровые изображения. Они в свою очередь делятся на типы: бинарные (черно-белые), полутоновые (в виде оттенков серого, grayscale), палитровые (индексированные), полноцветные.

Элементы ***бинарного изображения*** могут принимать только два значения – 0 или 1. Природа происхождения таких изображений может быть самой разнообразной. Но в большинстве случаев, они получаются в результате обработки полутоновых, палитровых или полноцветных изображений методами бинаризации с фиксированным или адаптивным порогом. Бинарные изображения имеют то преимущество, что они очень удобны при передаче данных.

Полутоновое изображение состоит из элементов, которые могут принимать одно из значений интенсивности какого-либо одного цвета. Обычно рассматривается смесь трех основных цветов и в этом случае изображение принимает вид оттенков серого (*grayscale*). Информация о яркости элементов изображения более приоритетна, поэтому такое представление изображения наиболее распространено, и оно чаще применяется в различных исследованиях. В большинстве случаев информация о яркости пикселя кодируется 8 битами, что определяет 256 уровней яркости.

В ***палитровых изображениях*** значение пикселей является ссылкой на ячейку карты цветов (палитры). Палитра представляет собой двумерный массив, в трех столбцах которого расположены интенсивности цветовых составляющих для определенного индекса.

Элементы ***полноцветных изображений*** непосредственно хранят информацию о яркостях цветовых составляющих (RGB). Для хранения полноцветных изображений требуется трехмерный массив, третий индекс которого определяет номер цвета для пикселя, определяемого первыми двумя индексами.

Выбор типа изображения зависит от решаемой задачи, от того, насколько полно и без потерь нужная информация может быть представлена данным типом изображения. Использование полноцветных изображений требует больших вычислительных затрат.

По умолчанию MATLAB работает с вещественными числами двойной точности (восемь байт для хранения числа типа *double*), а для работы с изображениями и сокращения требуемой памяти реализовано хранение данных также в виде однобайтных целых без знака (класс *uint*8). В зависимости от типа изображения они по-разному представляются в разных форматах. Информация об этом представлена в таблице 6.

Таблица 6 – Формат изображений

Тип изображения	*double*	*uint8*
Бинарное	0 и 1	0 и 1
Полутоновое	[0, 1]	[0, 255]
Палитровое	[1, размер палитры], где 1 – первая строка палитры	[0, 255], где 0 – индекс первой строки палитры
Полноцветное	[0, 1]	[0, 255]

Тип *uint8* изначально не предназначался для арифметических операций и не все функции и команды можно с ним использовать. Поэтому если с изображением необходимо пронести какие-нибудь численные действия, то вначале нужно преобразовать массив при помощи команды *A* = double(*A*) + l.

В MATLAB изображение представляется матрицей чисел, где значение каждого элемента отвечает определенному уровню квантования его энергетической характеристики (яркости). Это так называемая пиксельная система координат. Она применяется в большинстве функций пакета IPT. Существует также пространственная система координат, где изображение представляется непрерывным числовым полем квадратов с единичной величиной. Количество квадратов совпадает с числом пикселей. Значение интенсивности элемента в центре квадрата совпадает со значением соответствующего пикселя в пиксельной системе координат. При решении практических задач, связанных с измерениями реальных геометрических размеров объектов на изображении, удобно использовать пространственную систему координат, так как она позволяет учитывать разрешение (количество пикселей на метр).

Цветовые системы и их преобразования

Существует несколько цветовых систем используемых для представления полноцветной графической информации. Прежде всего, это наиболее распространенная система RGB (red, green, blue). Другая часто используемая система HSV (hue – цветовой тон, saturation – насыщенность, value – яркость). Система HSV соответствует особенностям человеческого глаза лучше, чем система RGB. Поэтому система HSV отвечает особенностям подбора цветов, которые используются художниками. Близкой к системе HSV является система HLS (hue, lightness, saturation).

Существуют и другие цветовые системы: CMYK, YIQ, YcbCr и др.

Базовые возможности среды MATLAB по работе с изображениями

Базовый набор функций системы MATLAB имеет развитые средства для работы с растровыми объектами (изображениями), включая подготовку растровых графических изображений, запись их в файл, считывание из файла изображений, созданных другими программами. Эти команды представлены в таблице 7.

Таблица 7 – Команды для операций чтения и сохранения изображений

Команда	Назначение
image	Вывод графического образа
imfinfo	Информация о графическом файле
imread	Чтение изображения из графического файла
imwrite	Запись изображения в графический файл

Для вывода на экран графического объекта в базовом наборе средств MATLAB существует функция *image*. Команда *image(C)* выводит двумерный или трехмерный массив C как графический образ. Пусть размер массива есть M×N или M×N×3, тогда число M определяет количество прямоугольников по горизонтали, а N – по вертикали. Если C – двумерный массив, то каждый элемент C рассматривается как значение индекса для массива, определяющего текущую палитру (команда colormap), и соответствующий этому элементу C прямоугольник окрашивается в этот цвет. Этот способ задания изображения называется Indexed image (индексированное изображение). В случае трехмерного массива C цвет точки (m,n) определяют элементы C(m,n,1:3), дающие соответственно доли красного, зеленого и синего цветов. При таком способе построения объекта image получаются изображения с числом цветов до 16 миллионов (truecolor image). В этом случае таблица цветов не используется.

Обращение image (X, Y, C), где X и Y – векторы, определяет размещение пикселя C(1,1) в точке с координатами (X(1),Y(l)) и пикселя C(M,N) соответственно в точке (X(end),Y(end)). По умолчанию MATLAB масштабирует выводимое изображение, поэтому пиксель обычно представляется в виде прямоугольника. Чтобы отменить масштабирование, нужно явно указать размеры.

Для записи растрового изображения (массив A) в файл *FILE* в графическом формате *TYPE* применяется команда *imwrite (A, FILE, TYPE)*. Команда считывания изображения из файла *FILE* в случае индексированного изображения имеет следующий вид: *[A, M] = imread(FILE, TYPE)* или *A = imread(FILE, TYPE)*. В массив A заносятся данные об изображении (цвета пикселей построчно), а массив M будет содержать таблицу цветов, если массив A двумерный (indexed image). Для считывания изображения truecolor image достаточно одного выходного параметра. При чтении и записи в качестве *TYPE* выступают следующие графические форматы: jpg (jpeg), tif (tiff), bmp, png, pcx, hdf, pcx, xmd. Чтобы узнать тип изображения в файле, можно использовать команду *imfinfo(FILE)*.

При помощи команды печати или меню графического окна можно распечатать изображение, записать рисунок в собственном формате (fig) и сохранить его в нескольких графических форматах.

Команда печати обычно имеет дополнительные параметры, указывающие графический формат (см. табл. ниже) и некоторые особенности. Так, для добавления рисунка к существующему файлу достаточно набрать: *print-append* file.

В MATLAB используются многие графические форматы, полную информацию о которых можно получить, обратившись к справке help print. Кроме того, сохранить изображение в каком-либо формате можно при помощи пункта Export... из меню File.

Основные форматы хранения растровых изображений.

BMP (Bitmap Picture) – формат хранения растровых изображений. Изначально формат мог хранить только аппаратно-зависимые растры (англ. Device Dependent Bitmap, DDB), но с развитием технологий отображения графических данных формат BMP стал преимущественно хранить аппаратно-независимые растры (англ. Device Independent Bitmap, DIB). С форматом BMP работает большое количество программ, так как его поддержка интегрирована в операционные системы Windows и OS/2. Файлы формата BMP могут иметь расширения .bmp, .dib и .rle. Кроме того, данные этого формата включаются в двоичные файлы ресурсов RES и в PE-файлы. Глубина цвета в данном формате может быть от 1 до 48 бит на пиксель, максимальные размеры изображения 65535×65535 пикселей. В формате BMP есть поддержка сжатия по алгоритму RLE, однако существуют форматы с более сильным сжатием, и из-за большого объёма BMP редко используется в Интернете, где для сжатия без потерь используются PNG и более старый GIF.

Данные изображения bmp – это последовательность пикселей, записанных в том или ином виде. Пиксели хранятся построчно, снизу вверх. Каждая строка изображения дополняется нулями до длины, кратной четырём байтам. В bmp файлах с глубиной цвета 24 бита байты цвета каждого пикселя хранятся в порядке BGR (Blue, Green, Red). В bmp файлах с глубиной цвета 32 бита байты цвета каждого пикселя хранятся в порядке BGRA (Blue, Green, Red, Alpha). При количестве 1, 2, 4 или 8 бит на каждый пиксель может использоваться специальный режим индексированных цветов. В этом случае число, соответствующее каждому пикселю, указывает не на цвет, а на номер цвета в палитре. Благодаря использованию палитры имеется возможность адаптировать изображение к цветам, присутствующим на изображении. В таком случае изображение ограниченно не заданными цветами, а максимальным количеством одновременно используемых цветов.

JPEG (Joint Photographic Experts Group – название разработчика) – один из популярных графических форматов, применяемый для хранения фотоизображений и подобных им изображений. Файлы, содержащие данные JPEG, обычно имеют расширения .jpeg, .jfif, .jpg, .JPG, или .JPE. MIME-типом является image/jpeg. Алгоритм JPEG является алгоритмом сжатия данных с потерями. Файл JPEG содержит последовательность маркеров, каждый из которых начинается с байта 0xFF, свидетельствующего о начале маркера, и байта-идентификатора.

Алгоритм JPEG в наибольшей степени пригоден для сжатия фотографий и картин, содержащих реалистичные сцены с плавными переходами яркости и цвета. Наибольшее распространение JPEG получил в цифровой фотографии и для хранения и передачи изображений с использованием сети Интернет. С

другой стороны, JPEG малопригоден для сжатия чертежей, текстовой и знаковой графики, где резкий контраст между соседними пикселями приводит к появлению заметных артефактов. Такие изображения целесообразно сохранять в форматах без потерь, таких как TIFF, GIF, PNG или RAW.

JPEG (как и другие методы искажающего сжатия) не подходит для сжатия изображений при многоступенчатой обработке, так как искажения в изображения будут вноситься каждый раз при сохранении промежуточных результатов обработки. JPEG не должен использоваться и в тех случаях, когда недопустимы даже минимальные потери, например, при сжатии астрономических или медицинских изображений. В таких случаях может быть рекомендован предусмотренный стандартом JPEG режим сжатия Lossless JPEG или стандарт сжатия JPEG-LS.

Вывод изображения на экран средствами Image Processing Toolbox

Для вывода изображения на экран существует специальная функция *imshow* из состава пакета Image Processing Toolbox. Функция *imshow(I, n)* выводит на экран полутоновое изображение I, используя при выводе n уровней серого. Если при вызове функции опустить параметр n, то когда MATLAB запущен в графическом режиме TrueColor, для вывода полутонового изображения используется 256 градаций серого или 64 градации серого, когда MATLAB запущен в графическом режиме с меньшим количеством цветов.

Функция *imshow(I, [low high])* выводит на экран полутоновое изображение I, дополнительно контрастируя выводимое изображение. Пиксели изображения I, яркость которых меньше либо равна low, отображаются черным цветом. Пиксели, яркость которых больше либо равна high, отображаются белым цветом. Пиксели, яркость которых имеет значение между low и high, отображаются серым цветом. Все уровни серого равномерно распределены от low до high. Если вызвать функцию *imshow(I, [ ])*, указав вторым аргументом пустой массив, то low будет присвоено минимальное значение в *I(low = min(I(:)))*, a high будет присвоено максимальное значение в I(max = max(I(:))).

Функция *imshow(BW*) выводит на экран бинарное изображение BW. Пиксели, значение которых равно 0, отображаются черным цветом. Пиксели, значение которых равно 1, отображаются белым цветом.

Функция *imshow(X, map)* выводит на экран палитровое изображение X с палитрой map.

Функция *imshow(RGB)* выводит па экран полноцветное изображение RGB.

Задания

Создать изображение 200×200 пикселей в виде трех квадратов основных цветов (размера 100×100), смещенных друг относительно друга (рис. 2).
Отобразить полученную матрицу как изображение.

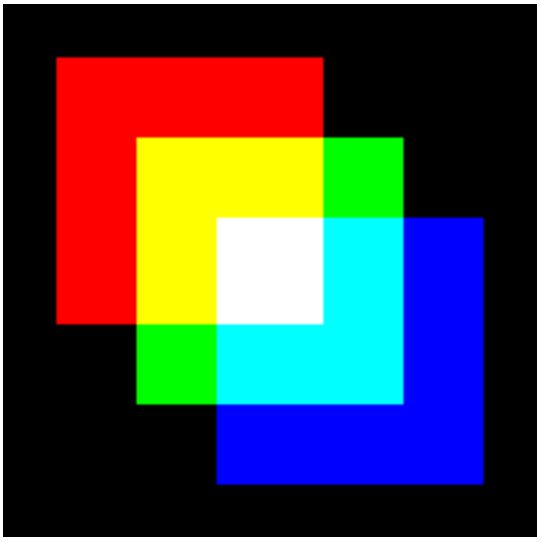

Рисунок 2 – Изображение в виде трех квадратов

Ввести цветное изображение из файла с изображением, предложенного преподавателем или выбранного самостоятельно, в виде информации о трех основных цветах (RGB-формат) в среду MATLAB. Отобразить введенную информацию средствами среды MATLAB на экране как изображение, а также отобразить распределение информации о каждом из трех основных цветов. Ввести из файла с изображением также информацию об изображении и отобразить ее в командном окне MATLAB. Затем преобразовать это изображение в изображение в виде информации об оттенках серого, а также в черно-белое изображение. Отобразить полученные изображения на экране рядом с цветным изображением.

Создать изображение (200×200) в виде трех смещенных по диагонали друг относительно друга кругов разных цветов (рис. 3). Диаметр одного круга должен составлять 100 пикселей, расстояние между краем изображения и кругом должно быть не менее 20 пикселей. Сохранить полученное изображение. Выполнить кадрирование полученного изображения, обрезав его по краям (сверху, снизу, слева и справа) на 10 % (т. е. убрать поля). Отобразить исходное и кадрированное изображение рядом. Уменьшить объем памяти, выделяемый под кадрированное изображение, путем прореживания (децимации) элементов матрицы, хранящей изображение, и отобразить уменьшенное изображение рядом с исходным. Увеличить размер децимированного изображения в 2 раза и отобразить рядом с исходным. Повернуть при помощи операций с матрицами кадрированное изображение на 90º, а также выполнить поворот его на 45º. Оба полученных изображения нужно отобразить рядом с исходным.

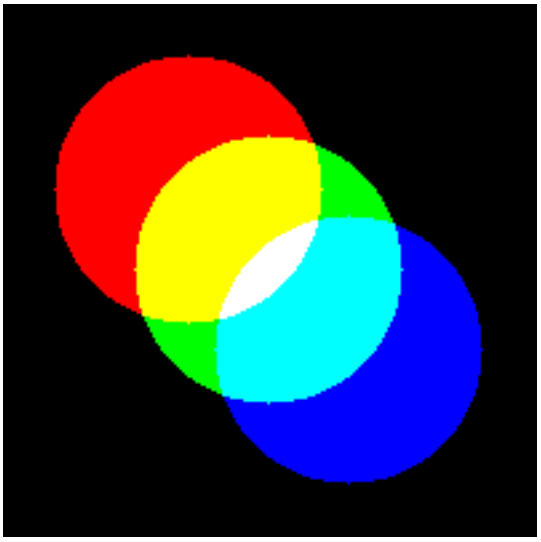

Рисунок 3 – Изображение в виде трех кругов

Пояснения к работе

Для выполнения всех действий, указанных в задании, необходимо создать скрипт с названием, соответствующим номеру работы, например 'Lab1_1.m', и разместить его в соответствующей папке.

Для обеспечения возможности многократного исполнения скрипта в течение одного сеанса работы необходимо в первой исполняемой строке дать команду очистить рабочую память Workspace и закрыть все открытые графические окна:

```
clear; close all
```

Перед выполнением задания необходимо предварительно определить используемые в дальнейшем параметры и задать значения индексирующей переменной i:

```
s = 4;
A = 256;
N = 200;
i = 1:N;
```

Для отображения изображений рекомендуется предварительно создавать графическое окно, в котором будут реализовываться операции по визуализации изображений. Это действие не является обязательным, поскольку любая функция MATLAB, создающая графический образ, автоматически предварительно создает графическое окно в качестве контейнера для создаваемого графического образа, задавая при этом его параметры значениями по умолчанию. Явно вызванная функция создания графического окна *figure(…)* позволяет задать ему те параметры, которые более предпочтительны для контекста конкретных действий. Ниже приведен пример ее вызова с заданием имени создаваемого графического окна 'Gaussian image' и указанием, что окно должно быть пристыкованным (docked) к рабочему столу MATLAB:

```
figure( 'Name', 'Gaussian image', 'WindowStyle', 'docked' );
```

Для реализации возможности сопоставления рядом нескольких вариантов отображения матрицы M как изображения необходимо специальной функцией создать подобласть вывода графического образа внутри графического окна:

```
subplot(2,3,1);
```

Первые два параметра функции subplot определяют количество подобластей по вертикали и горизонтали, а третий определяет номер текущей подобласти, в которую будут осуществляться текущие графические операции.

Изображение в MATLAB может быть представлено различными способами. Для отображения матрицы M как индексированного изображение, в котором значение каждого пикселя определяет номер цвета из некоторой заранее предопределенной палитры цветов, достаточно вызвать базовую функцию image:

```
image(M);
```

В таком варианте вызова функции image матрица M будет рассматриваться как индексированное изображение, использующее палитру по умолчанию ('Jet'). Для того чтобы пропорции создаваемого изображения определялись не пропорциями места, выделенного под него, а пропорциями самого изображения (а оно должно быть квадратным, поскольку матрица M является квадратной), необходимо применить команду

```
axis square;
```

Для того чтобы отличать созданное изображение от других необходимо ему дать заголовок, например, 'image(M)':

```
title('image(M)');
```

Чтобы не повлиять на цветовую палитру сформированного первого отображения матрицы M, следующее ее отображение как изображения в виде оттенков серого (grayscale) в том же графическом окне следует выполнять при помощи функций subimage из состава Image Processing Toolbox. Ее предназначение изолировать используемую при текущем формировании изображения палитру от ранее использовавшейся. Если текущее формирование индексированного изображения в графическом окне, в котором уже имеются ранее сформированные изображения, сделать при помощи функций image или imshow, используя при этом какую-то новую палитру, то это приведет к смене палитры и в ранее сформированных изображениях. Функция subimage позволяет формировать свое собственное изображение без конфликта с цветовыми схемами ранее сформированных изображений. Использование функции subimage, как и функции imshow, только с одним параметром в виде имени отображаемой матрицы без ссылки на требуемую палитру приведет к отображению матрицы в виде оттенков серого, когда значение каждого элемента матрицы будет интерпретироваться как информация об яркости соответствующего пикселя. Причем, если тип матрицы будет double, то минимальная яркость соответствует нулевому значению ее элемента, а максимальная – единичному. А если тип матрицы будет uint8 или uint16, то яркость будет определяться целочисленными значениями в диапазоне от 0 до 255. Поэтому для корректного отображения рассматриваемой матрицы M,

следует преобразовать ее к типу uint8 при помощи функции uint8. Таким образом, для отображения матрицы M в виде оттенков серого можно использовать следующий фрагмент

```
    subplot(2,3,2); subimage(uint8(M)); axis square;
title('subimage(M)');
```

Чтобы не выполнять преобразование типа матрицы M при ее отображении в виде оттенков серого можно воспользоваться функцией imshow из состава Image Processing Toolbox. Если ее вызвать в виде imshow(M, [low high]), то она будет отображать в виде оттенков серого значения матрицы из диапазона [low high]. Если вместо второго параметра использовать пустую матрицу [], то это будет эквивалентно вызову данной функции в виде imshow(M, [min(M (:)) max(M (:))]), который обеспечивает автоматическое масштабирование отображаемых значений матрицы M. Таким образом, для отображения матрицы M в виде оттенков серого при помощи функции imshow можно использовать следующий фрагмент

```
    subplot(2,3,3); imshow( M, [], 'InitialMagnification',
'fit' );
    axis square; title('imshow(M)');
```

Здесь при вызове функции imshow использовалось явное задание параметра отображения InitialMagnification значением fit, что позволяет согласовать размер изображения с размером выделенной под него области.

Для преобразования матрицы M как индексированного изображения в изображение в виде оттенков серого следует использовать функцию ind2gray, первым параметром которой должна быть преобразуемая матрица (предварительно преобразованная в тип uint8, чтобы значения ее элементов интерпретировались как соответствующие индексы), а вторым – используемая палитра:

```
    imshow( ind2gray(uint8(M),hsv), 'InitialMagnification',
'fit' );
```

Для преобразования матрицы M как индексированного изображения в черно-белое изображение следует использовать функцию im2bw, первым параметром которой должна быть матрица, преобразованная в тип uint8, вторым – используемая палитра, третьим – значение уровня в диапазоне от 0 до 1, разделяющего пиксели по яркости на черный или белый:

```
    imshow( im2bw(uint8(M), hot, 0.5), 'InitialMagnification',
'fit' ); title('imshow(BW)');
```

Цветное RGB изображение в MATLAB должно быть представлено трехмерным массивом данных, у которого первые два индекса используются для адресации пикселя, а третий индекс определяет номер одного из трех цветов: 1 – красный, 2 – зеленый и 3 – синий. При наличии трех матриц MR, MG и MB, определяющих интенсивность цветовых составляющих формируемого изображения, трехмерный массив изображения можно получить при помощи функции cat, реализуя ее вызов в виде cat(3,MR,MG,MB), что означает объединение трех матриц по третьей размерности.

Выполнение пункта 0 задания необходимо начать с формирования массива Squares, который будет хранить изображения трех квадратов разного цвета. Поскольку цвет фона формируемого изображения должен быть черным, то создаваемый массив необходимо инициализировать нулями. Сделать это можно при помощи функции zeros:

```
Squares = uint8(zeros(N,N,3));
```

Для экономии памяти сформированный массив Squares рекомендуется преобразовать к типу uint8.

Размер квадрата каждого цвета составляет половину (50 %) от размера формируемого изображения, поэтому диапазон индексов, отводимых на каждый квадрат можно задать индексированной переменной j = 0:N/2. Смещение относительно левого и верхнего краев изображения, реализующее смещение квадратов по диагонали, можно выбрать следующим: 10 % для красного квадрата, 25 % для зеленого квадрата и 40 % для синего квадрата. Задать его можно вектором N0 = [0.1 0.25 0.4]*N. Используя эти переменные, квадрат каждого цвета можно определить при помощи присваивания соответствующим пикселям максимального значения интенсивности определенного цвета, например, для красного квадрата при помощи выражения `Squares(N0(1)+j,N0(1)+j,1) = uint8(255).` Все три квадрата можно задать в цикле:

```
for k = 1:3, Squares(N0(k)+j,N0(k)+j,k) = uint8(255); end
```

Отображение сформированного массива Squares в виде изображения можно реализовать следующими командами:

```
figure( 'Name', 'Square images', 'WindowStyle', 'docked' );
imshow( Squares, 'InitialMagnification', 'fit' );
```

Для считывания изображения из файла в MATLAB, как того требует пункт 0 задания, существует команда imread, а считывания информации об изображении – команда imfinfo. Их вызов можно реализовать в виде:

```
I = imread('peppers.png');
Info = imfinfo('peppers.png');
```

Вывод содержимого структуры Info на экран для просмотра информации об изображении реализуется командой disp(Info), которая может быть использована для вывода на дисплей значения любого объекта MATLAB. Однако структура Info может содержать множество пустых полей, не содержащих информации, что приведет к выводу на дисплей большого числа пустых строк, затрудняющих восприятие информации об изображении в целом. Для очистки этой структуры от пустых строк (при помощи команды rmfield для удаления поля и функции isempty для проверки пустоты поля) и, затем, вывода ее содержимого на экран можно воспользоваться следующими командами

```
fInfo = fieldnames(Info);
Info = rmfield(Info, fInfo(structfun(@isempty, Info)));
disp(Info);
```

Информация о цветном изображении хранится в трехмерном массиве I, информацию о котором (размер, количество байт и т. п.) можно просмотреть в окне Workspace рабочего стола MATLAB. Если в этом окне для этого массива выполнить команду Plot all columns / More Plots… / image (для этого

использовать панель команд или контекстное меню), то массив отобразиться как изображение в текущем графическом окне. Информация о красной составляющей исходного цветного изображения хранится в двумерной матрице I(:,:,1), зеленой – в I(:,:,2), синей – в I(:,:,3).

Трехмерный массив, хранящий информацию о цветном RGB изображении, может быть преобразован в двумерную матрицу, хранящую информацию о яркости пикселей в виде оттенков серого, либо в матрицу черно-белого изображения. Преобразование цветного RGB изображения в изображение в виде оттенков серого выполняется функцией rgb2gray, а в черно-белое изображение – функцией im2bw(I,level), у которой второй параметр level определяет порог бинаризации (выше level – 1, ниже level – 0):

```
Gray = rgb2gray(I);
BW = im2bw(Gray, 0.5);
```

Окончательно, отображение всех полученных вариантов исходного изображения может быть реализовано следующим образом:

```
figure( 'Name', 'Image from file', 'WindowStyle', 'docked' );
subplot(2,3,1); imshow( I, 'InitialMagnification', 'fit' );
title('Color image');
subplot(2,3,2); imshow( Gray, 'InitialMagnification', 'fit' );
title('Grayscale image');
subplot(2,3,3); imshow( BW, 'InitialMagnification', 'fit' );
title('Black-white image image');
subplot(2,3,4); imshow( I(:,:,1), 'InitialMagnification',
'fit' ); title('Information about red');
subplot(2,3,5); imshow( I(:,:,2), 'InitialMagnification',
'fit' ); title('Information about green');
subplot(2,3,6); imshow( I(:,:,3), 'InitialMagnification',
'fit' ); title('Information about blue');
```

Для того, чтобы на базе исходного цветного изображения построить три новых, у каждого из которых интенсивность одной из трех составляющих цвета больше истинного в 1,5 раза, как того требует пункт **Ошибка! Источник ссылки не найден.** задания, рекомендуется воспользоваться функцией cat(dim, A, B), объединяющей массивы A и B вдоль размерности dim:

```
Ir = cat(3,1.5*I(:,:,1), I(:,:,2), I(:,:,3));
Ig = cat(3, I(:,:,1),1.5*I(:,:,2), I(:,:,3));
Ib = cat(3, I(:,:,1), I(:,:,2),1.5*I(:,:,3));
```

Отображение всех полученных вариантов вместе с исходным изображением может быть реализовано следующим образом:

```
figure( 'Name', 'Color manipulations', 'WindowStyle',
'docked' );
subplot(2,2,1); imshow( I, 'InitialMagnification', 'fit' );
title('Start image');
subplot(2,2,2); imshow( Ir, 'InitialMagnification', 'fit' );
title('Red image');
subplot(2,2,3); imshow( Ig, 'InitialMagnification', 'fit' );
title('Green image');
subplot(2,2,4); imshow( Ib, 'InitialMagnification', 'fit' );
title('Blue image');
```

Выполнение пункта 0 задания, как и третьего пункта, необходимо начать с формирования трехмерного массива Circles, который будет хранить исходное изображение трех кругов разного цвета. Поскольку цвет фона формируемого изображения должен быть черным, то создаваемый массив необходимо инициализировать нулями. Сделать это можно при помощи функции zeros, преобразовывая при этом сформированный массив к типу uint8 для экономии памяти:

```
Circles = uint8(zeros(N,N,3));
```

Для формирования круговой области в массиве Circles, заполненной одним числом, необходимо воспользоваться логической операцией проверки принадлежности точки кругу, выполняемой над операндами в виде матриц, которые хранят координаты рассматриваемой области. Результатом такой операции будет матрица, заполненная нулями или единицами, в зависимости от результата операции («ложь» или «истина») для данного матричного элемента. Радиус круга определяется значением N/4, центр круга – значением N/4 + N0(k), где k – номер цвета. Сетку с координатами рассматриваемой области с началом отсчета, расположенным в центре круга, можно задать при помощи специально предназначенной для этого функции meshgrid и введенной ранее индексированной переменной i, хранящей диапазон значений координат: [x, y] = meshgrid(i - N/4 - N0(k)). Для введенных таким образом координат условие принадлежности круговой области примет вид: x.^2 + y.^2 < = (N/4)^2. Исходя из этого, формирование круговых областей разного цвета в исходной матрице Circles можно реализовать в виде цикла:

```
for k = 1:3
    [x, y] = meshgrid(i - N/4-N0(k));
    Circles(:,:,k) = uint8(255*(x.^2+y.^2<=(N/4)^2));
end
```

Поскольку предполагается рядом с полученным изображением сопоставить его модификации, графическое окно, в котором будет отображаться изображение, следует поделить на 6 подобластей и массив Circles отобразить в первой подобласти:

```
figure( 'Name', 'Manipulation with image', 'WindowStyle',
'docked' );
subplot(2,3,1); imshow( Circles, 'InitialMagnification',
'fit' ); title('Image of circles');
```

Чтобы сохранить сформированное таким образом изображение в одном из стандартных графических форматов, необходимо воспользоваться функцией imwrite:

```
imwrite(Circles, 'Circles.tiff', 'tif');
```

Кадрирование изображения в Image Processing Toolbox выполняется функцией imcrop(I, rect), где I – обрабатываемое изображение, rect = [xmin ymin width height] – вектор, определяющий местоположение и размер кадрирующего прямоугольника. Поэтому, для того чтобы выполнить кадрирование полученного изображения, обрезав его по краям (сверху, снизу, слева и справа) на 10 %, необходимо реализовать вызов этой функции в виде:

```
Frame = imcrop( Circles, [0.1 0.1 0.8 0.8]*N );
```

Децимация (decimatio – десять) – это уменьшение частоты дискретизации дискретного во времени сигнала путем удаления его отсчетов. Использование децимации позволяет уменьшить объем памяти для хранения изображения (естественно за счет огрубления дискретного образа изображения). В MATLAB децимация реализуется очень просто при помощи перечисления только нечетных значений первых двух индексов массива Circles (конструкция n:m:k задает диапазон значений от n до k с шагом m, вместо максимального значения индекса можно указать зарезервированное имя end.):

```
Fdec = Frame(1:2:end,1:2:end,:);
```

Увеличение размера изображения предполагает решение задачи интерполяции при вставке новых точек между уже существующими. В Image Processing Toolbox для этого есть специальная функция imresize(A, scale, method), которая создает новое изображение, изменяя размеры исходного изображения A в scale раз. Для изменения размеров используется один из предопределенных методов интерполяции, который задается во входном параметре method в виде одной из следующих строк: 'nearest' (интерполяция значениями ближайших соседей), 'bilinear' (интерполяция с помощью двух линейных функций), 'bicubic' (интерполяция при помощи двух кубических функций). Таким образом, увеличение размера децимированного изображения в 2 раза можно сделать при помощи следующей команды:

```
Fbig = imresize( Fdec, 2, 'bilinear' );
```

Функция imrotate(A,angle,method) из Image Processing Toolbox осуществляет поворот изображения, заданного в массиве A на заданный угол angle против часовой стрелки, используя один из перечисленных выше методов интерполяции method. Исходя из этого, поворот кадрированного изображения на 45º можно реализовать так:

```
Frot = imrotate( Frame, 45, 'bilinear' );
```

При повороте при помощи матричных операций необходимо помнить, что массив, хранящий цветное изображение, по сути, состоит из трех матриц, и указанный поворот необходимо осуществить для каждой из них. Поворот на 90º можно осуществить при помощи функции rot90(A) из базового набора MATLAB. Поскольку предполагается осуществлять поворот для каждой матрицы цветов по отдельности, то для хранения результатов поворота необходимо создать массив, который будет хранить результат поворота:

```
Flip = uint8(zeros(size(Frame)));
```

Затем в цикле можно осуществить сам поворот:

```
for k = 1:3, Flip(:,:,k) = rot90(Frame(:,:,k)); end
```

Полученные новые изображения можно отобразить рядом с исходным при помощи следующих команд:

```
subplot(2,3,2); imshow( Fdec, 'InitialMagnification', 'fit' );
title('Image decimation');
subplot(2,3,3); imshow( Flip, 'InitialMagnification', 'fit' );
title('Image flip');
subplot(2,3,4); imshow( Frame, 'InitialMagnification', 'fit'
); title('Image cropping');
```

```
    subplot(2,3,5); imshow( Fbig, 'InitialMagnification', 'fit' );
title('Image increasing');
    subplot(2,3,6); imshow( Frot, 'InitialMagnification', 'fit' );
title('Image rotation');
```

Лабораторная работа 3
Гистограммы, профили и проекции

Цель работы: освоение основных яркостных и геометрических характеристик изображений и их использование для анализа изображений.

Методические рекомендации. До начала работы студенты должны ознакомиться с основными функциями среды MATLAB по работе с гистограммами, профилями и проекциями изображений.

Теоретические сведения.

Пиксель цифрового изображения характеризуется тремя параметрами: (x, y, I), где пара целочисленных значений (x, y) описывает геометрическое положение пикселя в плоскости изображения, а значение I характеризует его яркость (интенсивность) в точке плоскости. Таким образом, в изображении можно выделить яркостную и геометрическую составляющие. В общем случае данные составляющие не связаны между собой (например, изменение в освещенности сцены не изменит геометрических параметров объектов на сцене). Из-за этого проще исследовать отдельно яркостные свойства изображения и отдельно – геометрические. Такой подход понижает порядок исследуемого изображения в случае геометрических свойств с n = 3 (x, y, I) до n = 2 (x, y), а в случае яркостных свойств – до n = 1 (I). Яркостная составляющая изображения характеризуется одномерным массивом гистограммы, из которого можно вычислить контраст.

Гистограмма – это распределение частоты встречаемости пикселей одинаковой яркости на изображении.

Яркость – это среднее значение интенсивности сигнала.

Контраст – это интервал значений между минимальной и максимальной яркостями изображения.

Для сведения геометрических составляющих изображения к одномерному массиву данных n = 1 используются такие характеристики, как «профили» и «проекции» изображения.

Профиль вдоль линии – это функция интенсивности изображения, распределенного вдоль данной линии (прорезки).

Проекция на ось – это сумма интенсивностей пикселей изображения взятая в направлении перпендикулярном данной оси.

Гистограмма изображения. Для 8-битного полутонового изображения гистограмма яркости представляет собой одномерный целочисленный массив Hist из 256 элементов [0 . . . 255]. Элементом гистограммы Hist[i] является

сумма пикселей изображения с яркостью i. По визуальному отображению гистограммы можно оценить необходимость изменения яркости и контрастности изображения, оценить площадь, занимаемую светлыми и темными элементами, определить местоположение на плоскости изображения отдельных объектов, соответствующих некоторым диапазонам яркости. Для цветного RGB-изображения необходимо построить три гистограммы по каждому цвету.

Листинг 3.1. Построение гистограмм изображения.

```
[ numRows numCols Layers ] = size ( I );
imhist ( I (: ,: ,1)); % red
imhist ( I (: ,: ,2)); % green
imhist ( I (: ,: ,3)); % blue
```

Если гистограмма неравномерна, то для улучшения изображения можно ее выровнять, причем выравнивание гистограммы в зависимости от решаемой задачи можно выполнять различным образом.

Арифметические операции. Простейшими способами выравнивания гистограммы являются арифметические операции с изображениями. Например, в случае, если большинство значений гистограммы находятся слева, то изображение является темным. Для увеличения детализации темных областей можно сдвинуть гистограмму правее, в более светлую область, например, на 50 градаций для каждого цвета. В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = I(i,j) + 50 / 255;
```

Гистограмма исходного изображения сдвигается в среднюю часть диапазона, которая является более приемлемой с точки зрения визуального восприятия. Данный подход обладает следующим недостатком: повышение интенсивностей темных областей приводит к сдвигу светлых к максимуму, что может привести к потере информации в светлых областях. Растяжение динамического диапазона. Если интенсивности пикселей областей интереса находятся в узком динамическом диапазоне, то можно растянуть этот диапазон. Подобные преобразования выполняются согласно следующему выражению:

$$I_{new} = \left(\frac{I - I_{\min}}{I_{\max} - I_{\min}} \right)^{\alpha}, \qquad (3.1)$$

где I и I_{new} – массивы значений интенсивностей исходного и нового изображений соответственно; I_{min} и I_{max} – минимальное и максимальное значения интенсивностей исходного изображения соответственно; α – коэффициент нелинейности. Данное выражение является нелинейным из-за коэффициента α. В случае, если $\alpha = 1$, применение формулы (3.1) к исходному изображению не даст желаемого эффекта, поскольку гистограммы цветовых компонент изображения занимают весь возможный диапазон. Нелинейные преобразования проводятся для каждой цветовой составляющей.

Равномерное преобразование

Осуществляется по следующей формуле:

$$I_{new} = (I_{max} - I_{min}) \cdot P(I) + I_{min}, \quad (3.2)$$

где I_{min}, I_{max} – минимальное и максимальное значения интенсивностей исходного изображения I; $P(I)$ – функция распределения вероятностей исходного изображения, которая аппроксимируется кумулятивной гистограммой:

$$P(I) \approx \sum_{m=0}^{i} Hist(m) \quad (3.3)$$

Кумулятивная гистограмма исходного изображения в среде MATLAB может быть вычислена с помощью функции cumsum():

```
CH = cumsum(H)./ (numRows * numCols);
```

где H – гистограмма исходного изображения, numRows и numCols – число строк и столбцов исходного изображения соответственно.

Согласно формуле (3.2) можно вычислить значения интенсивностей пикселей результирующего изображения. В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = CH(ceil(255 * I(i,j) + eps));
```

Параметр eps необходим для защиты программы от присваивания индексам кумулятивной гистограммы нулевых значений.

Экспоненциальное преобразование.

Осуществляется по следующей формуле:

$$I_{new} = I_{\min} - \frac{1}{\alpha} \cdot \ln\left(1 - P(I)\right), \quad (3.4)$$

где α – постоянная, характеризующая крутизну преобразования.

Согласно формуле (3.4) можно вычислить значения интенсивностей пикселей результирующего изображения. В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = Imin -(1 / alfa) * ...
log10(1 - CH(ceil(255 * I(i,j) + eps)));
```

Преобразование по закону Рэлея.

Осуществляется по следующей формуле:

$$I_{new} = I_{\min} + \left(2\alpha^2 \frac{1}{1 - P(I)}\right)^{1/2}, \quad (3.5)$$

где α – постоянная, характеризующая гистограмму распределения интенсивностей элементов результирующего изображения. В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = Imin + sqrt(2 * alfa^2 * ...
```

```
log10(1 / (1 - CH(ceil(255 * I(i,j) + eps))));
```

Преобразование по закону степени 2/3

Осуществляется по следующей формуле:

$$I_{new} = \left(P(I)\right)^{2/3}. \quad (3.6)$$

В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = (CH(ceil(255 * I(i,j) + eps)))^(2/3);
```

Рассмотренные методы преобразования гистограмм могут применяться для устранения искажений при передаче уровней квантования, которым были подвергнуты изображения на этапе формирования, передачи или обработки данных. Кроме того, данные методы могут применяться не только ко всему изображению, но использоваться локально в скользящем окне, что позволит повысить детализированность отдельных областей.

В среде MATLAB реализовано несколько функций, автоматически выравнивающих гистограммы полутонового изображения:

1. *imadjust()* – повышает контрастность изображения, изменяя диапазон интенсивностей исходного изображения.

2. *histeq()* – эквализирует (выравнивает) гистограмму методом распределения значений интенсивностей элементов исходного изображения.

3. *adapthisteq()* – выполняет контрастно-ограниченное адаптивное выравнивание гистограммы методом анализа и эквализации гистограмм локальных окрестностей изображения.

Для обработки цветного изображения данные функции можно применять поочередно к каждому цвету.

Профиль изображения

Профилем изображения вдоль некоторой линии называется функция интенсивности изображения, распределенного вдоль данной линии (прорезки). Простейшим случаем профиля изображения является профиль строки:

$$\text{Profile } i(x) = I(x,i), \quad (3.7)$$

где i – номер строки изображения I.

Профиль столбца изображения:

$$\text{Profile } j(y) = I(j,y), \quad (3.8)$$

где j – номер столбца изображения I.

В общем случае профиль можно рассматривать вдоль любой прямой, ломаной или кривой линии, пересекающей изображение.

После формирования массива профиля изображения проводится его анализ стандартными средствами. Анализ позволяет автоматически выделять особые точки функции профиля, соответствующие контурам изображения, пересекаемым данной линией. Например, на рисунке 4 представлен профиль

изображения штрих-кода, взятого вдоль оси Ox. Данный профиль содержит всю необходимую информацию для считывания штрих-кода, поскольку позволяет определить последовательность чередования «толстых» и «тонких» штрихов и пробелов различной ширины. В среде MATLAB профиль изображения можно найти при помощи фунции improfile().

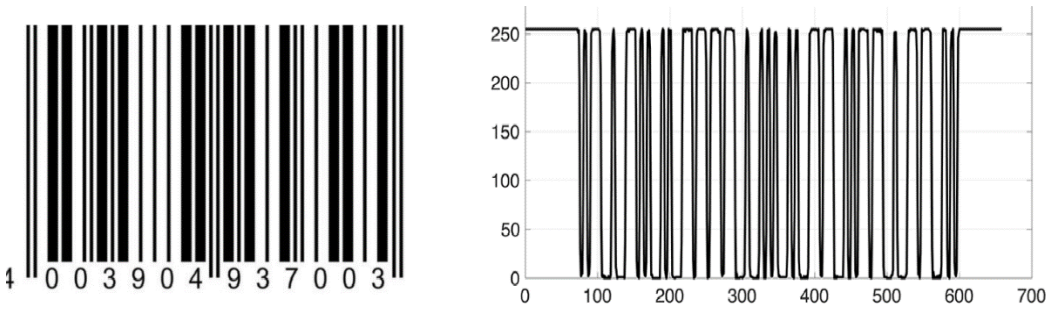

Рисунок 4 – Слева – штрих-код, справа – его профиль вдоль оси Ox

Листинг 3.2. Поиск профиля штрих-кода вдоль оси Ox

```
I = imread (‘code . jpg’);
[ numRows , numCols , Layers ] = size ( I );
x = [1 numCols ];
y = [ ceil ( numRows /2) ceil ( numRows /2)];
figure
improfile (I ,x , y ) , grid on ;
```

Для интерактивного указания линии (ломаной) вдоль которой следует построить профиль необходимо использовать функцию improfile без параметров.

Листинг 3.3. Интерактивное задание линии профиля

```
I = imread (‘code . jpg’);
imshow ( I );
improfile
```

Проекция изображения.

Проекцией изображения на некоторую ось называется сумма интенсивностей пикселей изображения в направлении, перпендикулярном данной оси. Простейшим случаем проекции двумерного изображения являются вертикальная проекция на ось Ox, представляющая собой сумму интенсивностей пикселей по столбцам изображения:

$$\mathrm{Pr}ojX\left(y\right)=\sum_{y=0}^{\dim Y-1} I\left(x,y\right), \tag{3.9}$$

и горизонтальная проекция на ось Oy, представляющая собой сумму интенсивностей пикселей по строкам изображения:

$$\mathrm{Pr}\,ojY(x) = \sum_{x=0}^{\dim X - 1} I(x, y). \qquad (3.10)$$

Анализ массива проекции позволяет выделять характерные точки функции проекции, которые соответствуют контурам объектов на изображении. Например, если на изображении имеются контрастные объекты, то в проекции будут видны перепады или экстремумы функции, соответствующие положению каждого из объектов.

На рисунке 5 показан пример проекции на ось Oy машиночитаемого документа. Видно, что две машиночитаемые текстовые строки порождают два существенных экстремума функции проекции.

Подобные проекции могут быть использованы в алгоритмах обнаружения и сегментации текстовых строк в системах распознавания текста.

Листинг 3.4. Определение положения текста

```
for i =1:1: numRows
Proj (i ,1) = ( round ( sum ( I (i ,: ,1))) + ...
round ( sum ( I(i ,: ,2))) + ...
round ( sum ( I(i ,: ,3)))) / (256 * 3);
end
```

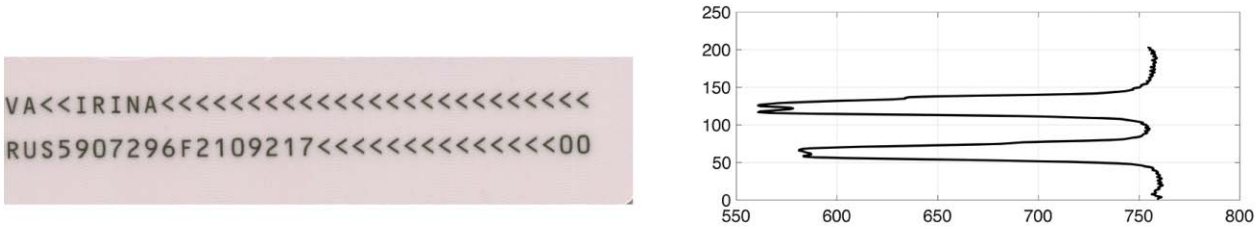

Рисунок 5 – Слева – фрагмент заграничного паспорта,
справа – его проекция на ось Oy

Порядок выполнения работы

1. Гистограммы. Выбрать произвольное слабоконтрастное изображение. Выполнить выравнивание гистограммы и растяжение контраста, использовать рассмотренные преобразования и встроенные функции пакета MATLAB. Сравнить полученные результаты.

2. Профили. Выбрать произвольное изображение, содержащее монотонные области и выделяющиеся объекты. Произвести построение проекций изображения на вертикальную и горизонтальную оси. Определить границы областей объектов.

3. Проекции. Выбрать произвольное изображение, содержащие штрих-код. Выполнить построение профиля изображения вдоль штрих-кода.

Содержание отчета

1. Цель работы.

2. Теоретическое обоснование применяемых методов и функций геометрических преобразований.

3. Ход выполнения работы:
(a) исходные изображения;
(b) листинги программных реализаций;
(c) комментарии;
(d) результирующие изображения.
4. Выводы о проделанной работе.
Вопросы к защите лабораторной работы:
1. Что такое контрастность изображения и как её можно изменить?
2. Чем эффективно использование профилей и проекций изображения?
3. Каким образом можно найти объект на равномерном фоне?

Лабораторная работа 4
Геометрические преобразования изображений

Цель работы. Освоение основных видов отображений и использование геометрических преобразований для решения задач пространственной коррекции изображений.

Методические рекомендации. До начала работы студенты должны ознакомиться с основными функциями среды MATLAB по работе с геометрическими преобразованиями изображений.

Теоретические сведения. Геометрические преобразования изображений подразумевают пространственное изменение местоположения множества пикселей с целочисленными координатами (x,y) в другое множество с координатами (x',y'), причем интенсивность пикселей сохраняется. В двумерных плоских геометрических преобразованиях, как правило, используется евклидово пространство P^2 с ортонормированной декартовой системой координат. В этом случае пикселю изображения соответствует пара декартовых координат, которые интерпретируются в виде двумерного вектора, представленного отрезком из точки (0,0) до точки $X_i = (x_i,y_i)$. Двумерные преобразования на плоскости можно представить в виде движения точек, соответствующих множеству пикселей.

Для общности с дальнейшими преобразованиями будем использовать однородные координаты, обладающие тем свойством, что определяемый ими объект не меняется при умножении всех координат на одно и то же ненулевое число. Из-за этого свойства необходимое количество координат для представления точек всегда на одну больше, чем размерность пространства P^n, в котором эти координаты используются. Например, для представления точки $X = (x,y)$ на плоскости в двумерном пространстве P^2 необходимо три координаты $X = (x,y,w)$. Проиллюстрируем это следующим примером:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = w \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} x' & y' & w \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} w, \quad (4.1)$$

где w – скалярный произвольный множитель, $x = x' \cdot w$, $y = y' \cdot w$.

При помощи троек однородных координат и матриц третьего порядка можно описать любое линейное преобразование плоскости.

Таким образом, геометрические преобразования являются матричными преобразованиями, и множества координат пикселей преобразованного и исходного изображений связаны следующим матричным соотношением либо в строчном виде $X' = XT$, либо в столбцовом $X' = T^{\mathrm{T}}X$.

Точка с декартовым координатами (x,y) в однородных координатах запишется как (x',y'):

$$\begin{cases} x' = Ax + By + C \\ y' = Dx + Ey + F \end{cases} \quad (4.2)$$

Линейные преобразования

Конформное отображение – это отображение, при котором сохраняется форма бесконечно малых фигур и углы между кривыми в точках их пересечения. Основными линейными конформными преобразованиями являются евклидовы преобразования. К ним относятся сдвиг, отражение, однородное масштабирование и поворот.

Конформные преобразования являются подмножеством аффинных преобразований.

Сдвиг изображения.

Система уравнений и матрица преобразования координат

T в случае сдвига изображения $A = E = 1$, $B = D = 0$ примут вид:

$$\begin{cases} x' = x + C \\ y' = y + D \end{cases}, \quad (4.3)$$

$$\Leftrightarrow \begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} T \Rightarrow T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ C & F & 1 \end{bmatrix} \quad (4.4)$$

где C и F – сдвиг по осям Ox и Oy соответственно.

Листинг 4.1. Сдвиг изображения на 50 и 100 пикселей по осям Ox и Oy соответственно. Размер исходного рисунка roadSign.jpg 300 × 300 пикселей

```
I = imread ( ‘ roadSign . jpg ’);
T = [1 0 0; 0 1 0; 50 100 1];
tform = affine2d ( T );
I_shift = imwarp (I , tform , ...
```

```
 'Interp' ,'nearest', ...
 'OutputView' , imref2d ( size ( I) , ...
 [1 size (I ,2)] , [1 size (I ,1)]));
```

Функция affine2d() создает матрицу преобразования, которая применяется к изображению I при помощи функции imwarp(). Дополнительные параметры задают одинаковые координаты для исходного и преобразованного изображений, а также метод интерполяции для неопределенных пикселей.

Отражение изображения.

Система уравнений (2.6) и матрица преобразования координат T в случае отражения изображения вдоль оси Ox при $A = 1$, $E = -1$, $B = C = D = F = 0$ примут вид:

$$\begin{cases} x' = x \\ y' = y \end{cases} \quad T \Rightarrow T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.5}$$

Листинг 4.2. Отражение относительно оси Ox

```
T = [1 0 0; 0 -1 0; 0 0 1];
tform = affine2d (T);
I_reflect = imwarp (I, tform );
```

Однородное масштабирование изображения.

Система уравнений (4.5) и матрица преобразования координат T в случае масштабирования изображения $A = \alpha$, $E = \beta$, $B = C = D = F = 0$ примут вид:

$$\begin{cases} x' = \alpha x \\ y' = \beta y \end{cases} \quad T \Rightarrow T = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.6}$$

если $\alpha < 1$ и $\beta < 1$, то изображение уменьшается, если $\alpha > 1$ и $\beta > 1$ – увеличивается. Если $\alpha \neq \beta$, то пропорции будут не одинаковыми по ширине и высоте. В общем случае данное отображение будет являться аффинным, а не конформным.

Листинг 4.3. Увеличение исходного изображения в два раза.

```
T = [2 0 0; 0 2 0; 0 0 1];
tform = affine2d ( T );
I_scale = imwarp (I , tform );
```

Поворот изображения.

Система уравнений (2.6) и матрица преобразования координат T в случае поворота изображения по часовой стрелке $A = \cos\phi$, $B = -\sin\phi$, $D = \sin\phi$, $E = \cos\phi$, $C = F = 0$ примут вид:

$$\begin{cases} x' = x\cos\varphi - y\sin\varphi \\ y' = x\sin\varphi + y\cos\varphi \end{cases} \Rightarrow T = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -s\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.7)$$

Листинг 4.4. Поворот изображения на $\phi = 10^{\circ}$

```
phi = 10* pi /180;
T = [ cos ( phi ) sin ( phi ) 0;…
sin ( phi ) cos ( phi ) 0;…
0 0 1];
tform = affine2d ( T );
I_rotate = imwarp (I , tform );
```

Конформные преобразования показаны на рисунке 6.

Рисунок 6 – Конформные преобразования; верхний ряд слева направо: исходное изображение, сдвиг, отражение, вращение; нижний ряд: однородное масштабирование.

Аффинное отображение – это отображение, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся в пересекающиеся, скрещивающиеся в скрещивающиеся; сохраняются отношения длин отрезков, лежащих на одной прямой (или на параллельных прямых), и отношения площадей фигур.

Базовыми преобразованиями являются конформные преобразования, скос и неоднородное масштабирование. Произвольное аффинное преобразование можно получить при помощи последовательного произведения матриц базовых преобразований. В непрерывной геометрии любое аффинное преобразование имеет обратное аффинное преобразование, а произведение прямого и обратного дает единичное преобразование, которое оставляет все точки на месте. Аффинные преобразования являются подмножеством проекционных преобразований.

Порядок выполнения работы

1. Простейшие геометрические преобразования. Выбрать произвольное изображение. Выполнить над ним линейные и нелинейные преобразования (конформные, аффинные и проективные отображения).

2. Коррекция дисторсии. Выбрать произвольное изображение либо с подушкообразной, либо с бочкообразной дисторсией. Выполнить коррекцию изображения.

3. «Склейка» изображений. Выбрать два изображения (снимки с фотокамеры, фрагменты сканированного изображения и пр.) на которых имеется область пересечения. Выполнить коррекцию второго изображения для его перевода в систему координат первого; затем выполнить автоматическую «склейку» из двух изображений в одно.

Содержание отчета

1. Цель работы.

2. Теоретическое обоснование применяемых методов и функций геометрических преобразований.

3. Ход выполнения работы:

(a) исходные изображения;

(b) листинги программных реализаций;

(c) комментарии;

(d) результирующие изображения.

4. Выводы о проделанной работе.

Вопросы к защите лабораторной работы:

1. Каким образом можно выполнить поворот изображения, не используя матрицу поворота?

2. Какое минимальное количество соответствующих пар точек необходимо задать на исходном и искаженном изображениях, если порядок преобразования $n = 4$?

3. После геометрического преобразования изображения могут появиться пиксели с неопределенными значениями интенсивности. С чем это связано и как решается данная проблема?

Лабораторная работа 5
Фильтрация и выделение контуров

Цель работы. Освоение основных способов фильтрации изображений от шумов и выделения контуров.

Методические рекомендации. До начала работы студенты должны ознакомиться с основными функциями среды MATLAB фильтрации изображений. Иметь представление о низкочастной и высокочастотной фильтрации.

Теоретические сведения

Типы шумов. Цифровые изображения, полученные различными оптикоэлектронными приборами (рис. 7), могут содержать в себе разнообразные искажения, обусловленные разного рода помехами, которые принято называть шумом. Шум на изображении затрудняет его обработку

автоматическими средствами и, поскольку шум может иметь различную природу, для его успешного подавления необходимо определить адекватную математическую модель. Рассмотрим наиболее распространенные модели шумов. В среде MATLAB шум может быть наложен на изображение с помощью функции *imnoise()*.

Импульсный шум

При импульсном шуме сигнал искажается выбросами с очень большими отрицательными или положительными значениями малой длительностью и может возникать, например, из-за ошибок декодирования. Такой шум приводит к появлению на изображении белых («соль») или черных («перец») точек, поэтому зачастую называется точечным шумом.

Рисунок 7 – Исходное полутоновое изображение

Для его описания следует принять во внимание тот факт, что появление шумового выброса в каждом пикселе $I(x,y)$ не зависит ни от качества исходного изображения, ни от наличия шума в других точках и имеет вероятность появления p, причем значение интенсивности пикселя $I(x,y)$ будет изменено на значение $d \in [0,255]$:

$$I(x,y) = \begin{cases} d, \text{с вероятностью } p \\ s_{x,y}, \text{с вероятностью } 1-p \end{cases}, \quad (5.1)$$

где $s_{x,y}$ – интенсивность пикселя исходного изображения, I – зашумленное изображение, если $d = 0$ – шум типа «перец», если $d = 255$ – шум типа «соль». В среде MATLAB задается параметром ‘salt & pepper’ функции *imnoise()*: `imnoise(I,'salt &pepper')`.

Аддитивный шум

Аддитивный шум описывается следующим выражением:

$$I_{new}(x,y) = I(x,y) + \eta(x,y), \tag{5.2}$$

где I_{new} – зашумленное изображение, I – исходное изображение, η – не зависящий от сигнала аддитивный шум с гауссовым или любым другим распределением функции плотности вероятности.

Мультипликативный шум

Мультипликативный шум описывается следующим выражением:

$$I_{new}(x,y) = I(x,y) \cdot \eta(x,y), \tag{5.3}$$

где I_{new} – зашумленное изображение, I – исходное изображение, η – не зависящий от сигнала мультипликативный шум, умножающий зарегистрированный сигнал. В качестве примера можно привести зернистость фотопленки, ультразвуковые изображения и т. д. Частным случаем мультипликативного шума является спекл-шум, который появляется на изображениях, полученных устройствами с когерентным формированием изображений, например, медицинскими сканерами или радарами. На таких изображениях можно отчетливо наблюдать светлые пятна, крапинки (спеклы), которые разделены темными участками изображения. В среде MATLAB спекл-шум накладывается на изображение I функцией *imnoise(I,'speckle')*.

Гауссов (нормальный) шум

Гауссов шум на изображении может возникать в следствие недостатка освещенности сцены, высокой температуры и т. д. Модель шума широко распространена в задачах низкочастотной фильтрации изображений. Функция распределения плотности вероятности $p(z)$ случайной величины z описывается следующим выражением (рис. 8):

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-\mu)^2}{2\sigma^2}}, \tag{5.4}$$

где z – интенсивность изображения (например, для полутонового изображения $z \in [0,255]$), μ – среднее (математическое ожидание) случайной величины z, σ – среднеквадратичное отклонение, дисперсия σ^2 определяет мощность вносимого шума. Примерно 67 % значенией случайной величины z сосредоточено в диапазоне $[(\mu - \sigma),(\mu + \sigma)]$ и около 96 % в диапазоне $[(\mu - 2\sigma),(\mu + 2\sigma)]$.

В среде MATLAB шум может быть задан с помощью функции *imnoise(I,'gaussian') или imnoise(I,'localvar')* в случае нулевого математического ожидания.

Шум квантования

Зависит от выбранного шага квантования и самого сигнала. Шум квантования может приводить, например, к появлению ложных контуров вокруг объектов или убирать слабо контрастные детали на изображении. Такой шум не устраняется. Приближенно шум квантования можно описать распределением Пуассона и задать функцией *imnoise(I,'poisson')*.

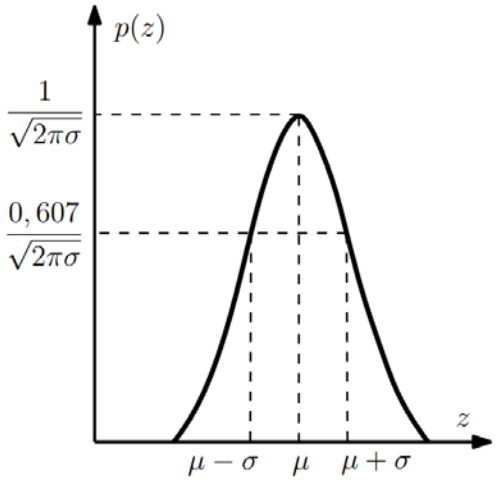

Рисунок 8 – Функция распределения плотности вероятности $p(z)$

Фильтрация изображений

Рассмотрим основные методы фильтрации изображений. Если для вычисления значения интенсивности каждого пикселя учитываются значения соседних пикселей в некоторой окрестности, то такое преобразование называется локальным, а сама окрестность – окном, представляющим собой некоторую матрицу, называемую маской, фильтром, ядром фильтра, а сами значения элементов матрицы называются коэффициентами. Центр маски совмещается с анализируемым пикселем, а коэффициенты маски умножаются на значения интенсивностей пикселей, накрытых маской. Как правило, маска имеет квадратную форму размера 3 × 3, 5 × 5 и т. п.

Фильтрация изображения I, имеющего размеры $M \times N$, с помощью маски размера $m \times n$ описывается формулой:

$$I_{new}(x,y) = \sum_s \sum_t w(s,t) I(x+s, y+t), \qquad (5.5)$$

где s и t – координаты элементов маски относительно ее центра (в центре $s = t = 0$). Такого рода преобразования называются линейными. После вычисления нового значения интенсивности пикселя $I_{new}(x,y)$ окно w, в котором описана маска фильтра, сдвигается и вычисляется интенсивность следующего пикселя, поэтому подобное преобразование называется фильтрацией в скользящем окне. На рисунке 9 показаны результаты наложения шума на исходное изображение.

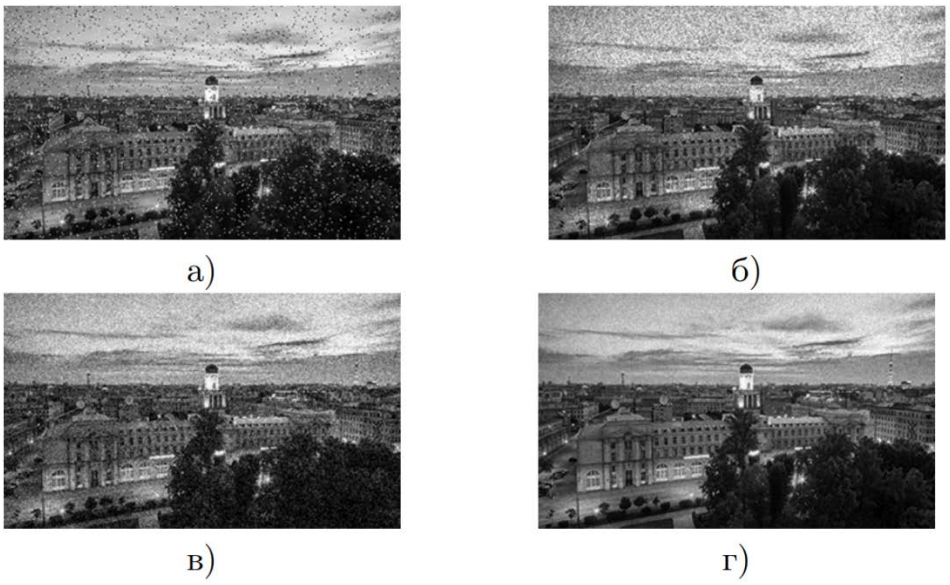

Рисунок 9 – Результат наложения:
а) – шума типа «соль» и «перец», б) – спекл-шума, в) – нормального шума,
г) – шума Пуассона

В среде MATLAB фильтрация изображения может быть осуществлена при помощи функции *filter2(mask,I)*, где матрица mask задает маску фильтра. Маска может задаваться либо вручную, либо с помощью функции *fspecial()*.

Низкочастотная фильтрация

Низкочастотные пространственные фильтры ослабляют высокочастотные компоненты (области с сильным изменением интенсивностей) и оставляют низкочастотные компоненты изображения без изменений. Используются для снижения уровня шума и удаления высокочастотных компонентов, что позволяет повысить точность исследования содержания низкочастотных компонентов. В результате применения низкочастотных фильтров получим сглаженное или размытое изображение. Главными отличительными особенностями являются:

1) неотрицательные коэффициенты маски;

2) сумма всех коэффициентов равна единице.

Рассмотрим основные виды низкочастнотных сглаживающих фильтров.

Арифметический усредняющий фильтр

Данный фильтр усредняет значение интенсивности пикселя по окрестности с использованием маски с одинаковыми коэффициентами, например, для маски размером 3 × 3 коэффициенты равны 1/9, при 5 × 5 – 1/25. Благодаря такому нормированию значение результата фильтрации будет приведено к диапазону интенсивностей исходного изображения. Графически двумерная функция, описывающая маску фильтра, похожа на параллелепипед, поэтому в англоязычной литературе используется название box-фильтр.

Арифметическое усреднение достигается при использовании следующей формулы:

$$I_{new}(x,y) = \frac{1}{m \cdot n} \sum_{i=0}^{m} \sum_{j=0}^{n} I(i,j), \quad (5.6)$$

где $I_{new}(x,y)$ – значение интенсивности пикселя отфильтрованного изображения, $I(i,j)$ – значение интенсивностей пикселей исходного изображения в маске, m и n – ширина и высота маски фильтра соответственно. Данный алгоритм эффективен для слабо зашумленных изображений. В среде MATLAB фильтрация изображения I с маской размера 3 × 3 может быть осуществлена при помощи функции *filter2(fspecial('average',3),I)*.

Фильтр Гаусса

Пиксели в скользящем окне, расположенные ближе к анализируемому пикселю, должны оказывать большее влияние на результат фильтрации, чем крайние. Поэтому коэффициенты весов маски можно описать колоколообразной функцией Гаусса. При фильтрации изображений используется двумерный фильтр Гаусса:

$$G_\sigma I = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \quad (5.7)$$

Чем больше параметр σ, тем сильнее размывается изображение.

Как правило, радиус фильтра $r = 3\sigma$. В таком случае размер маски $2r + 1 \times 2r + 1$ и размер матрицы $6\sigma + 1 \times 6\sigma + 1$. За пределами данной окрестности значения функции Гаусса будут пренебрежимо малы.

В среде MATLAB фильтрация изображения фильтром Гаусса может быть осуществлена при помощи функции *imgaussfilt(I)*.

Нелинейная фильтрация

Низкочастотные фильтры линейны и оптимальны в случае, когда имеет место нормальное распределение помех на цифровом изображении. Линейные фильтры локально усредняют импульсные помехи, сглаживая изображения. Для устранения импульсных помех лучше использовать нелинейные, например, медианные фильтры.

Медианная фильтрация

В классическом медианном фильтре используется маска с единичными коэффициентами. Произвольная форма окна может задаваться при помощи нулевых коэффициентов. Значения интенсивностей пикселей в окне представляются в виде вектора-столбца и сортируются по возрастанию. Отфильтрованному пикселю присваивается медианное (среднее) в ряду значение интенсивности. Номер медианного элемента после сортировки может быть вычислен по формуле $n = (N + 1)/2$, где N – число пикселей, участвующих в сортировке. В MATLAB может быть реализован с использованием функции *medfilt2(I)*.

Высокочастотная фильтрация

Высокочастотные пространственные фильтры усиливают высокочастотные компоненты (области с сильным изменением интенсивностей) и ослабляют низкочастотные составляющие изображения. Используются для выделения перепадов интенсивностей и определения границ (контуров) на изображениях. Для этого в MATLAB может быть использована функция *edge()*. В результате применения высокочастотных фильтров повышается резкость изображения.

Высокочастотные фильтры аппроксимируют вычисление производных по направлению, при этом приращение аргумента Δx принимается равным 1 или 2. Главными отличительными особенностями являются:

1) коэффициенты маски фильтра могут принимать отрицательные значения;

2) сумма всех коэффициентов равна нулю.

Фильтр Робертса. Фильтр Робертса работает с минимально допустимой для вычисления производной маской размерности 2×2, поэтому является быстрым и довольно эффективным.

В MATLAB с использованием дифференциального оператора Робертса границы можно выделить при помощи функции *edge(I, 'Roberts')*.

Фильтр Превитта. В данном подходе используются две ортогональные маски размером 3×3, позволяющие более точно вычислить производные по осям Ox и Oy. В MATLAB границы фильтром Превитта можно выделить при помощи функции *edge(I, 'Prewitt')*.

Фильтр Собела. Данный подход аналогичен фильтру Робертса, однако используются разные веса в масках. Типичный пример фильтра Робертса:

В MATLAB границы фильтром Собела можно выделить при помощи функции *edge(I, 'Sobel')*.

Фильтр Лапласа. Фильтр Лапласа использует аппроксимацию вторых производных по осям Ox и Oy в отличие от предыдущих подходов, использующих первую производную.

Алгоритм Кэнни. Одним из самых распространенных и эффективных алгоритмов выделения контуров на изображении является алгоритм Кэнни.

Данный алгоритм позволяет не только определять краевые пиксели, но и связные граничные линии. Алгоритм состоит из следующих шагов:

1. Устранение мелких деталей путем сглаживания исходного изображения с помощью фильтра Гаусса.

2. Использование дифференциального оператора Собела для определения значений модуля градиента всех пикселей изображения, причем результат вычисления округляется с шагом 45º.

3. Анализ значений модулей градиента пикселей, расположенных ортогонально исследуемому. Если значение модуля градиента исследуемого пикселя больше, чем у ортогональных соседних пикселей, то он является краевым, в противном случае – немаксимумом.

4. Выполнение двойной пороговой фильтрации краевых пикселей, отобранных на предыдущем шаге:

(a) если значение модуля градиента выше порога $t2$, то наличие края в пикселе является достоверным;

(b) если значение модуля градиента ниже порога $t1$, то пиксель однозначно не является краевым;

(c) если значение модуля градиента лежит в интервале $[t_1, t_2]$, то такой пиксель считается неоднозначным.

5. Подавление всех неоднозначных пикселей, не связанных с достоверными пикселями по 8-связности.

В MATLAB алгоритмом Кэнни можно выделить границы при помощи функции *edge(I, 'Canny')*.

Порядок выполнения работы

1. Типы шумов. Выбрать произвольное изображение. Получить искаженные различными шумами изображения с помощью функции imnoise() с отличными от значений по умолчанию параметрами.

2. Низкочастотная фильтрация. Обработать полученные в предыдущем пункте искаженные изображения фильтром Гаусса и контргармоническим усредняющим фильтром с различными значениями параметра Q.

3. Нелинейная фильтрация. Обработать полученные в первом пункте искаженные изображения медианной, взвешенной медианной, ранговой и Винеровской фильтрациями при различных размерах маски и ее коэффициентов. Реализовать адаптивную медианную фильтрацию.

4. Высокочастотная фильтрация. Выбрать исходное изображение. Выделить границы фильтрами Робертса, Превитта, Собела, Лапласа, алгоритмом Кэнни.

Содержание отчета.

1. Цель работы.

2. Теоретическое обоснование применяемых методов и функций фильтрации изображений.

3. Ход выполнения работы:

(a) исходные изображения;

(b) листинги программных реализаций;

(c) комментарии;

(d) результирующие изображения.

4. Выводы о проделанной работе.

Вопросы к защите лабораторной работы:

1. В чем заключаются основные недостатки адаптивных методов фильтрации изображений?

2. При каких значениях параметра Q контргармонический фильтр будет работать как арифметический, а при каких – как гармонический?

3. Какими операторами можно выделить границы на изображении?

4. Для чего на первом шаге выделения контуров, как правило, выполняется низкочастотная фильтрация?

Лабораторная работа 6
Сегментация изображений

Цель работы. Освоение основных способов сегментации изображений на семантические области.

Методические рекомендации. До начала работы студенты должны ознакомиться с основными функциями среды MATLAB по преобразованию цветовых пространств изображений и способами определения порогов.

Теоретические сведения

Бинаризация изображений

Простейшим способом сегментации изображения на два класса (фоновые пиксели и пиксели объекта) является бинаризация. Бинаризацию можно выполнить по порогу или по двойному порогу. В первом случае:

$$I_{new}(x,y)=\begin{cases}0, I(x,y)\le t\\ 1, I(x,y)>t\end{cases}, \quad (6.1)$$

где I – исходное изображение, I_{new} – бинаризованное изображение, t – порог бинаризации. Бинаризация данным методом в среде MATLAB может быть выполнена с использованием функций *im2bw()* (устаревшая) или *imbinarize()*.

Листинг 6.1. Бинаризация.

```
I = imread ( ‘ pic . jpg ’);
L = 255;
t = 127 / L ; % norm to 0...1
Inew = im2bw (I , t );
```

Бинаризация по двойному порогу (диапазонная бинаризация):

$$I_{new}(x,y)=\begin{cases}0, I(x,y)\le t_1\\ 1, t_1< I(x,y)\le t_2\\ 0, I(x,y)>t_2\end{cases}, \quad (6.2)$$

где I – исходное изображение, I_{new} – бинаризованное изображение, t_1 и t_2 – верхний и нижний пороги бинаризации. Бинаризация данным методом в среде MATLAB может быть выполнена с использованием функции r*oicolor()*. Для преобразования полноцветного изображения в полутоновое можно предварительно воспользоваться функцией *rgb2gray()*.

Листинг 6.2. Бинаризация по двойному порогу.

```
I = imread ( ‘ pic . jpg ’);
t1 = 110;
t2 = 200;
Igray = rgb2gray ( I );
Inew = roicolor ( Igray , t1 , t2 );
```

Пороги бинаризации t, t_1 и t_2 могут быть либо заданы вручную, либо вычислены с помощью специальных алгоритмов. В случае автоматического вычисления порога можно воспользоваться следующими алгоритмами.

1. Поиск максимального I_{max} и минимального I_{min} значений интенсивности исходного полутонового изображения и нахождение их среднего арифметического. Среднее арифметическое будет являться глобальным порогом бинаризации t:

$$t = (I_{max} - I_{min})/2. \quad (6.3)$$

2. Поиск оптимального порога t на основе модуля градиента яркости каждого пикселя. Для этого сначала вычисляется модуль градиента в каждой точке (x,y):

$$G(x,y) = \max\{|I(x+1, y) - I(x-1, y)|, |I(x,y+1) - I(x,y-1)|\}, \quad (6.4)$$

затем вычисляется оптимальный порог t.

3. Вычисление оптимального порога t статистическим методом Отсу (Оцу, англ. Otsu), разделяющим все пиксели на два класса 1 и 2, минимизируя дисперсию внутри каждого класса $\sigma_1^2(t)$ и $\sigma_2^2(t)$ и максимизируя дисперсию между классами.

В среде MATLAB порог t методом Отсу может быть вычислен с использованием функции graythresh():

Листинг 6.3. Бинаризация методом Отсу.

```
I = imread ( ‘ pic . jpg ’);
t = graythresh ( I );
Inew = im2bw (I , t );
```

либо с использованием функции otsuthresh() на основе гистограммы изображения:

Листинг 6.4. Бинаризация методом Отсу на основе гистограммы.

```
I = imread ( ‘ pic . jpg ’);
Igray = rgb2gray ( I );
 [ counts , x ] = imhist ( Igray );
t = otsuthresh ( counts );
Inew = imbinarize ( Igray , t );
```

4. Адаптивные методы, работающие не со всем изображением, а лишь с его фрагментами. Такие подходы зачастую используются при работе с изображениями, на которых представлены неоднородно освещенные объекты. В среде MATLAB порог t адаптивным методом может быть вычислен при помощи функции *adaptthresh()*:

Листинг 6.5. Бинаризация адаптивным методом.

```
I = imread ( ‘ pic . jpg ’);
Igray = rgb2gray ( I );
t = adaptthresh ( Igray );
Inew = imbinarize ( Igray , t );
```

Помимо рассмотренных методов существуют и многие другие, например, методы Бернсена, Эйквела, Ниблэка, Яновица и Брукштейна и др.

Сегментация изображений

Рассмотрим несколько основных методов сегментации изображений.

На основе принципа Вебера

Алгоритм предназначен для сегментации полутоновых изображений. Принцип Вебера подразумевает, что человеческий глаз плохо воспринимает разницу уровней серого между $I(n)$ и $I(n) + W(I(n))$, где $W(I(n))$ – функция Вебера, n – номер класса, I– кусочно-нелинейная функция градаций серого.

Можно объединить уровни серого из диапазона $[I(n),I(n) + W(I(n))]$ заменив их одним значением интенсивности.

Алгоритм сегментации состоит из следующих шагов:

1. Инициализация начальных условий: номер первого класса $n = 1$, уровень серого $I(n) = 0$.

2. Вычисление значения $W(I(n))$ по формуле Вебера и присваивание значения $I(n)$ всем пикселям, интенсивность которых находится в диапазоне $[I(n),I(n) + W(I(n))]$.

3. Поиск пикселей с интенсивностью выше $G = I(n) + W(I(n)) + 1$. Если найдены, то увеличение номера класса $n = n + 1$, присваивание $I(n) = G$ и переход на второй шаг. В противном случае закончить работу. Изображение будет сегментировано на n классов интенсивностью $W(I(n))$.

На основе кластеризации методом k-средних.

Идея метода заключается в определении центров k-кластеров и отнесении к каждому кластеру пикселей, наиболее близко относящихся к этим центрам. Все пиксели рассматриваются как векторы $x_i, i = 1, p$. Алгоритм сегментации состоит из следующих шагов:

1. Определение случайным образом k векторов $m_j, j = 1, k$, которые объявляются начальными центрами кластеров.

2. Обновление значений средних векторов m_j путем вычисления расстояний от каждого вектора xi до каждого mj и их классификации по критерию минимальности расстояния от вектора до кластера, пересчет средних значений mj по всем кластерам.

3. Повторение шагов 2 и 3 до тех пора, пока центры кластеров не перестанут изменяться.

Реализация метода очень похожа на предыдущий подход и содержит ряд аналогичных действий (используем исходное изображение рис. 4.1). Будем работать в цветовом пространстве Lab, поэтому первым шагом перейдем из пространства RGB в Lab:

Листинг 6.6. Сегментация на основе кластеризации методом k-средних.

```
I = imread ( ‘ pic2 . jpg ’);
Ilab = rgb2lab ( I );
L = Ilab (: ,: ,1);
a = Ilab (: ,: ,2);
b = Ilab (: ,: ,3);
```

Рассмотрим координатную плоскость (*a*,*b*). Для этого сформируем трехмерный массив ab, а затем функцией *reshape()* превратим его в двумерный вектор, содержащий все пиксели изображения:

```
ab (: ,: ,1) = a ;
ab (: ,: ,2) = b ;
nrows = size (I , 1);
ncols = size (I , 2);
1ab = reshape ( ab , nrows * ncols , 2);
```

Кластеризация методом k-средних в среде MATLAB осуществляется функцией kmeans(). Аналогично предыдущему способу разобьем изображение на три области соответствующих цветов.

Используем евклидову метрику (параметр ‘distance’ со значением ‘sqEuclidean’ и для повышения точности повторим процедуру:

```
кластеризации три раза (параметр ‘Replicates’ со значением
3)):
k = 3;
 [ ids , centers ] = kmeans ( ab , k , ‘ distance ’ ,...
 ‘ sqEuclidean ’ , ‘ Replicates ’ , 3);
 label = reshape ( ids , nrows , ncols );
```

Матрица label размера равном исходному изображению будет содержать идентификаторы классов для каждого пикселя. Для сегментации изображения на фрагменты segmentedFrames используем следующий листинг:

```
segmentedFrames = cell (1 , 3);
rgbLabel = repmat ( label , [1 1 3]);
for i = 1:1: k
color = I ;
color ( rgbLabel ~= i ) = 0;
segmentedFrames { i } = color ;
figure , imshow ( segmentedFrames { i });
end
```

Массив L содержит значение о светлоте изображения. Используя эти данные можно, например, сегментированные красные области разделить на светло-красные и темно-красные сегменты.

Порядок выполнения работы

1. Бинаризация. Выбрать произвольное изображение. Выполнить бинаризацию изображения при помощи рассмотренных методов. В зависимости от изображения использовать бинаризацию по верхнему или нижнему порогу.

2. Сегментация 1. Выбрать произвольное изображение, содержащее лицо(-а). Выполнить сегментацию изображения либо по принципу Вебера, либо на основе цвета кожи (на выбор).

3. Сегментация 2. Выбрать произвольное изображение, содержащее ограниченное количество цветных объектов. Выполнить сегментацию изображения в пространстве CIE Lab либо по методу ближайших соседей, либо по методу *k*-средних (на выбор).

4. Сегментация 3. Выбрать произвольное изображение, содержащее две разнородные текстуры. Выполнить текстурную сегментацию изображения,

оценить не менее трех параметров выделенных текстур, определить к какому классу относятся текстуры.

Содержание отчета.

1. Цель работы.

2. Теоретическое обоснование применяемых методов и функций сегментации изображений.

3. Ход выполнения работы:

(a) исходные изображения;

(b) листинги программных реализаций;

(c) комментарии;

(d) результирующие изображения.

4. Выводы о проделанной работе.

Вопросы к защите лабораторной работы:

1. В каких случаях целесообразно использовать сегментацию по принципу Вебера?

2. Какие значения имеют цветовые координаты a и b цветового пространства CIE Lab в полутновом изображении?

3. Зачем производить сегментацию в цветовом пространстве CIE Lab, а не в исходном RGB?

4. Что такое цветовое пространство и цветовой охват?

Литература

1. Гонсалес, Р. Цифровая обработка изображений в среде MATLAB / Р. Гонсалес, Р. Вудс, С. Эддинс. – М. : Техносфера, 2006. – 616 с.
2. Дьяконов, В. П. MATLAB 6.0/6.1/6.5/6.5 + SP1 + Simulink 4/5. Обработка сигналов и изображений / В. П. Дьяконов. – М. : СОЛОН-Пресс, 2004. – 592 с.
3. Дьяконов, В. П. MATLAB 6.5 SP1/7 + Simulink 5/6. Работа с изображениями и видеопотоками / В. П. Дьяконов. – М. : СОЛОН-Пресс, 2005. – 400 с.
4. Дьяконов, В. П., Абраменкова И. В. MATLAB. Обработка сигналов и изображений. Специальный справочник / В. П. Дьяконов. – СПб. : Питер, 2002. – 608 с.
5. Методы компьютерной обработки изображений / Под ред. В. А. Сойфера. – М. : Физматлит, 2003. – 784 с.
6. Сергиенко, А. Б. Цифровая обработка сигналов : учеб. пособие / А. Б Сергиенко. – 2-е изд. – СПб. : Питер, 2006. – 752 с.
7. Солонина, А. И. Основы цифровой обработки сигналов : курс лекций / А. И. Солонина, Д. А. Улахович [и др.] – СПб : БХВ-Петербург, 2005. – 768 с.
8. Фисенко, В. Т. Компьютерная обработка и распознавание изображений : учеб. пособие / В. Т. Фисенко, Т. Ю. Фисенко – СПб. : СПбГУ ИТМО, 2008. – 192 с.
9. Яне, Б. Цифровая обработка изображений / Б. Яне. – М. : Техносфера, 2007. – 584 с.

Учебное издание

Кириллов А. Г.

Основы систем технического зрения

Методические указания
по выполнению лабораторных работ

Редактор Р.А. Никифорова
Корректор А.С. Прокопюк
Компьютерная верстка А.Г. Кириллов

Подписано к печати 16.05.2024. Усл. печ. листов .
Уч.-изд. листов 3,8. Заказ № 112.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 72.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1497 от 30 мая 2017 г.