МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

О.І. Михальов, Вікторія В. Гнатушенко,
Володимир В. Гнатушенко

М Е Т О Д И Ч Н І В К А З І В К И

до виконання лабораторних робіт з дисципліни

“Методи обробки графічної інформації та синтезу віртуальної реальності”

для студентів напряму 122 – “Комп’ютерні науки”

Дніпро НМетАУ - 2019

УДК 681.3.06+519.68
Методичні вказівки до виконання лабораторних робіт з дисципліні "Комп'ютерні методи обробки зображень". Для студентів напряму 0501 – "Комп'ютерні науки". - Част. 1 / Укл.: О.І. Михальов, Вік.В. Гнатушенко, Вол.В. Гнатушенко. Під ред. О.І. Михальова. – Дніпро: НМетАУ, 2019. – 44 с.

Методичні вказівки є першою частиною комплексу навчально-методичних матеріалів з дисципліні "Комп'ютерні методи обробки зображень", в якій розглянуті основні можливості системи MATLAB для обробки зображень, наведені оператори, функції і команди MATLAB та приклади їхнього застосування. Основна увага приділена градаційним перетворенням, лінійній та нелінійній фільтрації, обробці кольорових зображень, их конвертированию, статистической обработке і т.д. Практична частина містить завдання для самостійного виконання лабораторних робіт.

Методичні вказівки призначені для студентів напряму підготовки 122 – "Комп'ютерні науки", а також для слухачів курсів підвищення кваліфікації, студентів і аспірантів інших спеціальностей.

Укладачі: О.І. Михальов, д-р техн. наук, проф.
Вікторія В. Гнатушенко, д-р. техн. наук, доц.
Володимир В. Гнатушенко, д-р. техн. наук, проф.

Відповідальний за випуск О.І. Михальов, д-р техн. наук, проф.
Рецензент В.І. Корсун, д-р техн. наук, проф.

Друкується за авторською редакцією.

Затверджено на засіданні кафедри інформаційних технологій і систем, протокол № 6 від 06.03.2019.

Підписано до друку 10.05.2019. Формат 60x84 1/16. Папір типогр. Друк різограф. Облік.-вид. арк 2,15. Умов. друк. арк. 1,99.
Тираж 100 пр. Замовл. № / .

Національна металургійна академія України.
49600, Дніпро, пр. Гагаріна, 4

ДНВП "Системні технології"

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Многие отрасли техники, имеющие отношение к получению, обработке, хранению и передаче информации, в значительной степени ориентируются в настоящее время на развитие систем, в которых информация имеет характер изображений. Изображение, которое можно рассматривать как двумерный сигнал, является значительно более емким носителем информации, чем обычный одномерный (временной) сигнал. Вместе с тем, решение научных и инженерных задач при работе с визуальными данными требует особых усилий, опирающихся на знание специфических методов, поскольку традиционная идеология одномерных сигналов и систем мало пригодна в этих случаях. В связи с этим, в вузовских программах появляются дисциплины, направленные на изучение принципов обработки изображений, причем, приоритетное внимание уделяется цифровым методам, привлекательным своей гибкостью.

Под *компьютерной обработкой изображений* подразумевается обработка цифровых изображений с помощью цифровых вычислительных машин (компьютеров). Отметим, что цифровое изображение состоит из конечного числа элементов, каждый из которых расположен в конкретном месте и имеет определенное значение. Эти элементы принято называть *элементами изображения* или *пикселами* [1-3, 5].

В данном пособии рассматриваются компьютерные методы обработки изображений с помощью MATLAB, который является языком высокого уровня для выполнения технических и научных вычислений. В нем интегрированы вычисления, визуализация и программирование в удобной пользовательской среде, в которой задачи и их решения выражаются с помощью привычных математических обозначений. MATLAB представляет собой интерактивную систему, в которой базовым элементом выступает массив элементов, который не требует задания фиксированной размерности. Это позволяет легко формулировать условия и решения многих вычислительных задач, которым требуется матричное представление объектов. При этом необходимая работа занимает лишь малую долю времени, которое потребовалось бы для написания аналогичных программ на скалярном и неинтерактивном языке типа C или Fortran.

Напомним, что название MATLAB происходит от английского словосочетания *MATrix LABoratory*. Система MATLAB была написана для облегчения доступа к матричным программным продуктам, разработанным в рамках проектов LINPACK (Linear System Package) и EISPACK (Eigen System Package). В настоящее время ядро MATLAB встроено в библиотеки LAPACK (Linear Algebra Package) и BLAS (Basic Linear Algebra Subprograms), которые включают самое современное программное обеспечение для матричных вычислений. Система MATLAB имеет расширения в виде наборов специализированных программ, которые по-английски называются *toolbox* (набор инструментов). Пакет Image Processing Toolbox (IPT) состоит из функций MATLAB (они называются M-функции или M-файлы), которые расширяют возможности стандартной среды MATLAB для решения задач цифровой обработки изображений. Приложение поддерживает различные операции обработки изображений, включая

- пространственные преобразования изображений [1-6],
- морфологические операции [1-6],
- скользящую и блочную обработку [1-6, 11],
- линейную фильтрацию различными фильтрами [1-6, 10],
- анализ и улучшение изображений [1-3, 11],
- восстановление изображений [1-3, 14, 15, 17],
- удаление размытостей [1-3],
- обработку области интереса [1-3,19].

Другие наборы toolbox, которые иногда используются в IPT, — это Signal Processing Toolbox (пакет обработки сигналов), Neural Network Toolbox (пакет для нейронной сети), Fuzzy Logic Toolbox (пакет с нечеткой логикой) и Wavelet Toolbox (пакет для работы с вейвлетами).

Следует отметить, что ограниченный объем пособия не позволил охватить многие важные аспекты проблемы компьютерной обработки изображений.

ЛАБОРАТОРНАЯ РАБОТА №1
УЛУЧШЕНИЕ НИЗКОКОНТРАСТНЫХ ИЗОБРАЖЕНИЙ

Цель: Знакомство с некоторыми командами и функциями языка MatLab, предназначенными для работы с изображениями.

Считывание и отображение изображений

Очистим рабочее пространство MATLAB от всех переменных и закроем открытые окна отображений.

```
clear, close all
```

Для считывания изображений используется функция imread. В этом примере показано считывание одного изображения pout.tif, которое включено в приложение Image Processing Toolbox, и его запоминание в виде массива I.

```
I = imread('pout.tif');
```

Функция imread считывает данные из графического формата, представленного как TIFF (Tagged Image File Format). Список всех поддерживаемых форматов можно найти в описании функции imread или в [1].

Теперь рассмотрим вопрос отображения считанного изображения. Приложение включает две функции отображения изображений: imshow и imtool. imshow представляет собой фундаментальную функцию отображения изображений. Функция imtool запускает инструментарий Image Tool, который представляет собой интегрированную среду для визуализации изображений и выполняет некоторые операции по обработке изображений. Средство отображения изображений Image Tool включает ряд дополнительных функций, которые дают возможность предоставлять информацию о конкретном пикселе или изображении в целом, повышать контраст изображения и т.д.

```
imshow(I)
```

Рис.1.1 - Полутоновое изображение pout.tif

Построенное командой imshow изображение показано на рис.1.1.

Информация об изображении в рабочем пространстве

Функция imread передает данные изображения в рабочее пространство. В рабочем пространстве может отображаться информация о всех переменных, которые были созданы в MATLAB на протяжении одного сеанса работы. Функция imread возвращает данные изображения переменной I, которая представляется массивом в формате uint8 с размерностью 291x240 элементов. Система MATLAB может отображать также массивы и других форматов - uint8, uint16 или double. Существует возможность получения информации о всех переменных, которые находятся в рабочем пространстве. Для этого используется команда whos.

```
whos
Name      Size          Bytes      Class
 I       291x240        69840  uint8 array
Grand total is 69840 elements using 69840 bytes
```

Улучшение контраста изображений

Данные из файла pout.tif представляют собой изображение с низким уровнем контраста. Для просмотра гистограммы распределения интенсивностей элементов изображения pout.tif, необходимо воспользоваться функцией imhist для построения гистограммы (рис.1.2). (Использование функции imhist приводит к созданию гистограммы изображения I в отдельном окне просмотра).

figure, imhist(I)

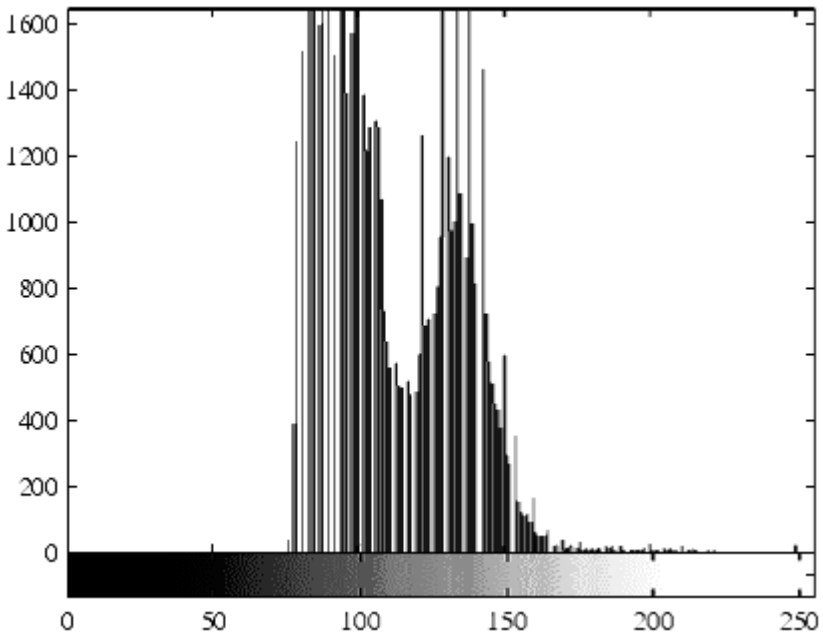

Рис.1.2

Отметим, что диапазон интенсивностей изображения является достаточно узким. Он не покрывает весь диапазон [0, 255], что является одной из причин низкой контрастности изображения. Приложение предоставляет несколько путей для повышения контрастности изображений. Один из них заключается в использовании функции histeq, которая обеспечивает равномерное распределение интенсивностей на весь диапазон. Этот процесс называется выравниваем гистограммы.

```
I2 = histeq(I);
```

Отобразим изображение I2 в новом окне просмотра.

```
figure, imshow(I2)
```

В данном пособии многие результаты работы, в том числе изображение pout.tif после эквализации гистограммы, не приводятся.

Также функция imhist создает гистограмму преобразованного изображения I2. Это позволяет сравнивать гистограммы до и после выравнивания (рис.1.3).

```
figure, imhist(I2)
```

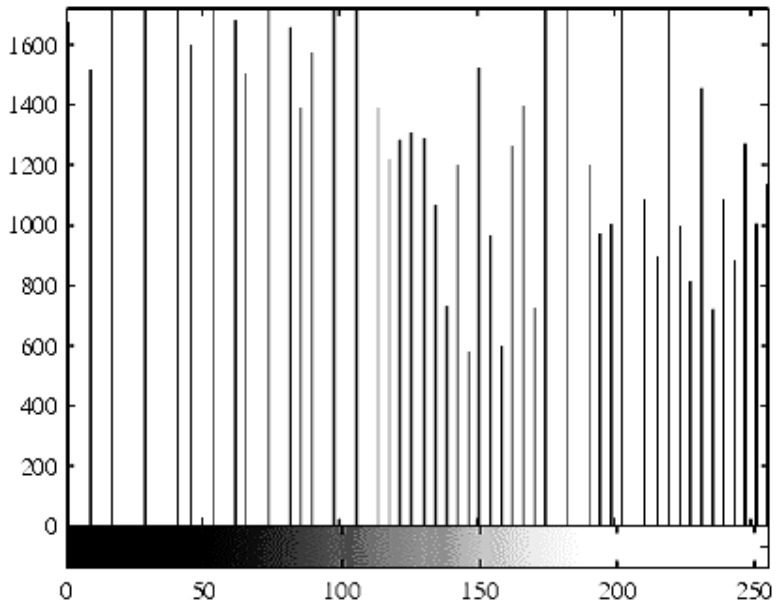

Рис.1.3

Приложение IPT включает также и несколько других функций, которые реализуют повышение контрастности, в частности, это функции imadjust и adapthisteq. В то же время, приложение включает также такое интерактивное средство как Adjust Contrast tool, с помощью которого можно улучшать контраст и корректировать интенсивности на изображении, отображаемом в Image Tool. Для этого используется функция imcontrast.

Запись изображений в файл на диск

Для записи нового улучшенного изображения I2 в файл на диск используется функция imwrite. Если эта функция включает название и расширение '.png', то функция imwrite записывает изображение в формате PNG (Portable Network Graphics). Также существует возможность записать изображение и в других форматах.

```
imwrite (I2, 'pout2.png');
```

Список всех доступных форматов можно посмотреть в описании функции imwrite.

Получение информации о графическом файле

Для получения информации о записанном на диск файле используется функция imfinfo. Эта функция предоставляет информацию о файле изображения, формате, размере и т.д.

```
imfinfo('pout2.png')
```

Справка о функции imadjust

```
J = imadjust(I, [LOW_IN HIGH_IN], [LOW_OUT HIGH_OUT], GAMMA)
```

Происходит преобразование значений интенсивности изображения I в новые значения J так, что значения, находящиеся в диапазоне LOW_IN и HIGH_IN, преобразуются в значения диапазона LOW_OUT and HIGH_OUT. Если используется пустая матрица ([]) для [LOW_IN HIGH_IN] или для [LOW_OUT HIGH_OUT], тогда подразумевается диапазон [0, 1]. GAMMA определяет форму кривой нелинейного управления яркостью. Если аргумент отсутствует, GAMMA по умолчанию равна 1 (линейное преобразование).

ЗАДАНИЕ

1. Проработать все описанные выше примеры, используя дополнительно файл `kids.tif`.
2. Самостоятельно изучить возможности функций `imadjust` и `adapthisteq`.
3. Оформить отчет о проделанной работе согласно ГОСТ и требованиям преподавателя.

ЛАБОРАТОРНАЯ РАБОТА №2
ОСНОВЫ ЦВЕТОВОГО КОДИРОВАНИЯ

Цель: Знакомство с принципами цветового кодирования, конвертирования цветовых систем, а также с некоторыми командами и функциями языка MatLab, предназначенными для работы с цветными изображениями.

Определение глубины цвета

Большинство компьютеров при отображении использует 8, 16 или 24 бит на пиксель. Этим определяется глубина цвета отображаемого изображения. Независимо от того сколько цветов отображает система, MATLAB может запоминать и обрабатывать изображения с различным числом бит на пиксель: 224 цветов для RGB-изображений в формате uint8, 248 цветов для RGB-изображений в формате uint16 и 2159 цветов для RGB-изображений в формате удвоенной точности. Эти изображения наилучшим образом отображаются в системе с 24-битным представлением цвета, хотя источники формирования изображений не всегда могут обеспечивать такую глубину цвета. В большинстве случаев они обеспечивают 16-битное представление цвета.

Описание выбора глубины цвета

В зависимости от используемой системы, можно устанавливать различные значения количества бит на пиксель. Это может быть также связано с графическим разрешением объектов. В большинстве случаев 24-битное представление обеспечивает хорошую визуализацию. Если есть необходимость использовать меньшее число бит на пиксель, то можно использовать 16-битное представление. При обработке полутоновых изображений, в большинстве случаев, достаточно 8 бит на пиксель.

Описание глубины цвета

Для определения глубины цвета, который может отображать система, используется следующий код.

```
get(0,'ScreenDepth')
ans =
      32
```

MATLAB возвращает число бит на пиксель:

Значение	Описание
8	8-битное представление отображается 256 цветами. 8-битные полутоновые изображения являются составной частью 24-битного представления графической информации.
16	16-битное представление обычно использует 5-бит на каждую цветовую компоненту, что равно 32 градациям (т.е. 2^5) на красную, зеленую и синюю составляющие. В результате такое представление поддерживает 32,768 (т.е., 2^{15}) различных цветов. Некоторые системы используют дополнительный бит для увеличения числа градаций отображаемого цвета. В нашем случае число различных цветов при 16-битном представлении равно 64,536 (т.е. 2^{16}).
24	24-битная визуализация использует 8 бит на каждую из трех цветовых составляющих, т.е. 256 (2^8) градаций на красную, зеленую и синюю компоненту. В результате получается 16,777,216 (т.е. 2^{24}) различных цветов.
32	32-битная визуализация использует 24 бита для запоминания цветовой информации а еще 8 бит используется для запоминания насыщенности (прозрачности) данных. Это так называемый альфа канал.

Уменьшение числа цветов на изображении

В этом пункте описано метод уменьшения числа цветов в индексных или RGB изображениях. Рассмотрим также метод диффузионного псевдосмешения цветов (dithering). Этот метод использует визуальное увеличение количества цветов на изображении.

Ниже приведены краткие описания функций уменьшения цветов на изображении в приложении Image Processing Toolbox.

Функция	Назначение
imapprox	Создает новое палитровое изображение из исходного палитрового, уменьшая количество используемых цветов.
rgb2ind	Создает палитровое изображение из полноцветного изображения RGB.

В системах с 24-битным отображением цвета RGB изображения могут отображаться 16,777,216 (т.е. 2^{24}) цветами. В системах с меньшим количеством отображаемых цветов RGB изображения также будут отображаться хорошо, потому что MATLAB автоматически использует аппроксимацию цветов и диффузионное псевдосмешение цветов.

Индексные изображения могут иметь проблемы с большим числом цветов. В основном, ограничиваются 256 цветами. Это связано со следующими причинами:

- В системах с 8-битным отображением при визуализации индексных изображений, которые имеют больше, чем 256 цветов, применяется метод диффузионного псевдосмешения цветов (dithering), поскольку не все цвета могут быть представлены системой.
- В некоторых системах палитра в принципе не может иметь больше, чем 256 позиций.
- Если индексное изображение содержит больше, чем 256 цветов, MATLAB запоминает данные изображения в массив не в формате uint8, а в формате удвоенной точности. Это приводит к увеличению объема запоминаемых данных, поскольку каждый пиксель занимает 64 бита.
- Файлы изображений больших размеров желательно записывать в формате, который использует при визуализации 256 цветов. Если при записи (с использованием функции imwrite) в этом же формате изображения имеют больше чем 256 цветов, то в дальнейшем возможны ошибки при визуализации.

Уменьшение числа цветов на индексном изображении

Использование `rgb2ind`

Функция `rgb2ind` выполняет преобразование RGB-изображения в индексное изображение, уменьшая количество отображаемых цветов. При обработке исходного изображения функция использует следующие методы аппроксимации цветов:

- Квантование
    - Равномерное квантование
    - Квантование с наименьшей дисперсией
- Отображение палитры

Качество результирующего изображения зависит от выбранного метода аппроксимации, диапазона цветов исходного изображения и использования

метода диффузионного псевдосмешения цветов (dithering). Отметим, что результат работы методов очень зависит также от конкретного изображения.

Квантование

Квантование приводит к уменьшение количества цветов на изображении. Функция `rgb2ind` использует квантование как часть алгоритма уменьшения цветов. Функция `rgb2ind` поддерживает два метода квантования: равномерное квантование и квантование с наименьшей дисперсией. При рассмотрении этого вопроса применяется понятие куб RGB цветов. Куб RGB цветов представляет собой трехмерный массив всех цветов, которые определены для этого типа данных. Поскольку изображения в MATLAB могут быть представлены в различных форматах (uint8, uint16 или double), то это будет влиять на дискретизацию цветов в кубе RGB.

Равномерное квантование. Для выполнения равномерного квантования используется функция rgb2ind с соответствующими параметрами.

Внизу приведен пример равномерного квантования. Второй аргумент влияет на дискретность квантования.

```
RGB = imread('peppers.png');
[x,map] = rgb2ind(RGB, 0.1);
```

Квантование с наименьшей дисперсией. Для реализации квантования с наименьшей дисперсией используется функция `rgb2ind` с указанием максимального числа цветов на результирующем изображении. Это число определяет количество ячеек, на которое будет разбит цветовой куб RGB. Рассмотрим пример реализации метода квантования для создания индексного изображения с использованием 185 цветов.

```
RGB = imread('peppers.png');
[X,map] = rgb2ind(RGB,185);
```

В основу метода квантования с наименьшей дисперсией положено объединение пикселей в группы на основании отклонений между их значениями. Т.е. выбранный пиксель должен иметь наименьшее суммарное отклонение от всех пикселей группы.

Уменьшение количества цветов на индексном изображении

Для уменьшения количества цветов на изображении используется также функция imapprox. Функция imapprox использует некоторые методы аппроксимации. По сути, функция imapprox сначала с помощью функции ind2rgb выполняет преобразование изображения в формат RGB, а потом использует

функцию rgb2ind для преобразования в индексное изображение с измененным количеством цветов.

Пример.

Рассмотрим пример формирования изображений, которые содержат 128 и 16 цветов с использованием функций `rgb2ind` и imapprox соответственно.

```
L=imread('peppers.png');
figure,imshow(L);
L=im2double(L);
[x,map] = rgb2ind(L, 128);
figure,imshow(x,map); %Изображение со 128 цветами
[Y,newmap] = imapprox(x,map,16);
figure,imshow(Y, newmap); %Изображение с 16 цветами
```

Качество обработанного изображения зависит от того какой метод аппроксимации используется, от количества цветов на исходном изображении, используется или нет метод диффузионного псевдосмешения цветов (dithering). Отметим, что различные методы дают различный результат для разных изображений.

Сглаживание цветовых переходов

Метод диффузионного псевдосмешения цветов (dithering)

При использовании функций `rgb2ind` или `imapprox` для уменьшения количества цветов на изображении, качество результирующего изображения немного ниже. Это связано с уменьшением количества цветов, с помощью которых отображается изображение. Обе функции - `rgb2ind` и `imapprox` - применяют метод диффузионного псевдосмешения цветов (dithering). Это приводит к визуальному увеличению количества отображаемых цветов. Метод dithering изменяет цвета пикселей окрестности таким образом, что средний цвет окрестности аппроксимирует исходный RGB-цвет.

Рассмотрим пример работы метода диффузионного псевдосмешения цветов (dithering):

1. Считывание и визуализация исходного изображения.

```
rgb=imread('onion.png');
imshow(rgb);
```

2. Создание индексного изображения с восьмью цветами без применения метода диффузионного псевдосмешения цветов (dithering).

```
[X_no_dither,map]=rgb2ind(rgb,8,'nodither');
figure, imshow(X_no_dither,map);
```

3. Создание индексного изображения с восьмью цветами с применением метода диффузионного псевдосмешения цветов (dithering).

```
[X_dither,map]=rgb2ind(rgb,8,'dither');
figure, imshow(X_dither,map);
```

Отметим, что обработка изображения методом диффузионного псевдосмешения цветов (dithering) приводит к визуальному увеличению отображаемых цветов. Однако возникает риск возникновения ложных контуров.

Выполнение преобразования цветовых пространств

Преобразование цветовых данных между цветовыми пространствами

Наиболее часто в Image Processing Toolbox для описания цифровых изображений используется цветовая система RGB [1-3]. В этом случае столбцы палитры представляют собой интенсивности красной, зеленой и синей составляющих. Палитровые изображения могут иметь произвольную глубину цвета, хотя наиболее широкое распространение получили палитровые изображения, глубина цвета которых составляет соответственно 4 и 8 бит на пиксель.

Известны два подхода к отображению цветов из большего цветового пространства в меньшем. Один из них заключается в том, что цвета, которые находятся за пределами цветового поля преобразуются в наиболее близкие по тону цвета внутри цветового поля. Этот подход называется отсечением (Clipping). Второй метод - это метод сжатия (Compression). Он заключается в том, что каждый цвет на входе независимо от того, попадает он в цветовое поле выводного устройства или нет, приводится к другому цвету из цветового диапазона выходного устройства (естественно, далеко не случайным образом). Существующие методы преобразования цветовых пространств отличаются друг от друга тремя главными особенностями: сжатием цветовой гаммы, тональной компрессией (приведение динамического диапазона вводного устройства к выводному) и отображением точки белого цвета.

Преобразования между устройство-зависимыми цветовыми пространствами

Любой цвет в пространстве RGB формируется как сумма разных количеств красной, зеленой и синей составляющих. Когда значения всех

составляющих равны нулю, тогда формируется черный цвет. Если все составляющие принимают максимально возможное значение, тогда формируется белый цвет. Однако понятие "белый цвет" является приближенным. Дело в том, что RGB-составляющие обеспечивают только хорошее приближение, а настоящий белый цвет может быть получен только сложением всех его спектральных составляющих, а не только R,G и B. В CMYK-пространстве белый цвет достигается обнулением всех его составляющих, а Cyan (C, Голубой), Magenta (M, Пурпурный) и Yellow (Y, Желтый) используются в нем для создания прочих цветов. Недостатком этого цветового пространства является то, что устройства, которые его используют, не могут отображать яркие и насыщенные цвета. Цветовые пространства RGB и CMYK являются устройство-зависимыми, так как цвет в них привязан к тому или иному устройству, для которого заданы эти значения. "Устройством" может быть принтер, сканер, монитор и пр. Действительно, каждый принтер, сканер или монитор одно и то же изображение будут отображать своими цветами, хотя значения RGB на них подаются одинаковые. В общем случае, цвет может быть описан координатами для данного набора основных цветов или координатами цветности и уровнем их яркости. Цвет можно описать с помощью линейной или нелинейной функции координат цвета или координат цветности и яркости. Линейные преобразования координат цвета представляют собой переход к новому набору цветов. Существующие системы координат цвета позволяют количественно описать цвета и формулы их преобразования.

Конвертирование из RGB в HSV

Синтаксис:

```
HSV=rgb2hsv(RGB)
hsvmap=rgb2hsv(rgbmap)
```

Описание:

Функция `HSV=rgb2hsv(RGB)` создает полноцветное изображение, значения пикселов которого представлены в цветовой системе HSV (hue — цветовой тон, saturation — насыщенность, value — яркость), из исходного полноцветного изображения в цветовой системе RGB. Результирующее изображение имеет формат представления данных double.

Функция `hsvmap=rgb2hsv(rgbmap)` создает палитру hsvmap, значения цвета в которой задаются в цветовой системе HSV, из исходной палитры rgbmap, хранящей цвета в цветовой системе RGB.

Зрение человека более чувствительно к изменениям яркости, чем к изменениям цвета. Этот факт делает целесообразным производить обработку отдельно для яркостной составляющей. Для этого полноцветное изображение из цветовой системы RGB следует преобразовать в систему, в которой одна из составляющих является сигналом яркости, а другие описывают цвет; например, таким требованиям отвечают системы HSV или YIQ (Y—яркостная составляющая, I — цветовой тон, Q — насыщенность).

Конвертирование из RGB в YIQ

Синтаксис:

```
YIQ=rgb2ntsc(RGB)
yiqmap=rgb2ntsc(rgbmap)
```

Описание:

Функция `YIQ=rgb2ntsc(RGB)` создает полноцветное изображение, значения пикселов которого представлены в цветовой системе YIQ, из исходного полноцветного изображения в цветовой системе RGB. Результирующее изображение имеет формат представления данных double.
Функция yiqmap=rgb2ntsc(rgbmap) создает палитру yiqmap, цвета в которой задаются в цветовой системе YIQ, из исходной rgbmap, хранящей цвета в цветовой системе RGB.

Цветовая система YIQ используется в телевизионной системе NTSC представления цветного телевизионного сигнала. Составляющая Y содержит информацию о яркости изображения, а составляющие I и Q - о его цветности. Иные системы телевидения, как аналоговые, так и цифровые, используют похожие цветовые системы. Их отличие от системы YIQ заключается, как правило, в коэффициентах, применяемых для формирования составляющих цветности.

Конвертирование из RGB в YCbCr

Синтаксис:

```
YCbCr=rgb2ycbcr(RGB)
ycbcrmap=rgb2ycbcr(rgbmap)
```

Описание:

Функция `YCbCr=rgb2ycbcr(RGB)` создает полноцветное изображение, значения пикселов которого представлены в цветовой системе YCbCr, из исходного полноцветного изображения в цветовой системе RGB. Формат представления данных исходного и результирующего изображений совпадают.

Функция `ycbcrmap=rgb2ycbcr(rgbmap)` создает палитру ycbcrmap, значения цвета в которой задаются в цветовой системе YCbCr, из исходной палитры rgbmap, хранящей цвета в цветовой системе RGB. Цветовая система YCbCr широко используется в цифровом видео. Составляющая Y содержит информацию о яркости изображения, а составляющие Cb и Cr (так называемые цветоразностные составляющие) - о его цветности. В результате преобразования составляющая Y принадлежит диапазону [16, 235], а составляющие Cb и Cr принадлежат диапазону [16, 240]. Оставшиеся допустимые значения вне указанных диапазонов ([0, 15] и [236, 255] для Y, [0, 15] и [241, 255] для Cb, Cr) используются для дополнительной информации (например, звука), которая передается вместе с видеопотоком.

Пример: Представление изображений в различных цветовых пространствах

Считаем изображение в формате RGB в рабочее пространство MATLAB и преобразуем цветовые данные в цветовое пространство XYZ:

1. Считывание исходных данных.

```
I_rgb = imread('peppers.png');
figure, imshow(I_rgb);
```

2. Создадим структуру преобразования цвета. В этой структуре определены преобразования между двумя цветовыми пространствами. Для формирования этой структуры используется функция `makecform`.
3. В этом примере создается структура преобразования цветовых данных из RGB в XYZ.

```
C = makecform('srgb2xyz');
```

4. Выполнение преобразований. Для выполнения описанных преобразований используется функция `applycform`, которая в качестве аргумента использует исходные данные и структуру преобразования цвета. Результатом работы функции `applycform` являются преобразованные данные.

```
I_xyz = applycform(I_rgb,C);
figure, imshow(I_rgb);

    whos
```

```
      Name           Size              Bytes    Class

      C              1x1                7744  struct array
      I_xyz        384x512x3          1179648  uint16
array
      I_rgb        384x512x3           589824  uint8 array
```

Утилита `colormapeditor`

Если средствами Matlab было активизировано некоторое изображение, тогда по команде

```
colormapeditor
```

на экране появляется окно редактора цветовой карты (рис.4). Этот же редактор можно запустить из меню окна, в котором размещено изображение (Edit > Colormap).

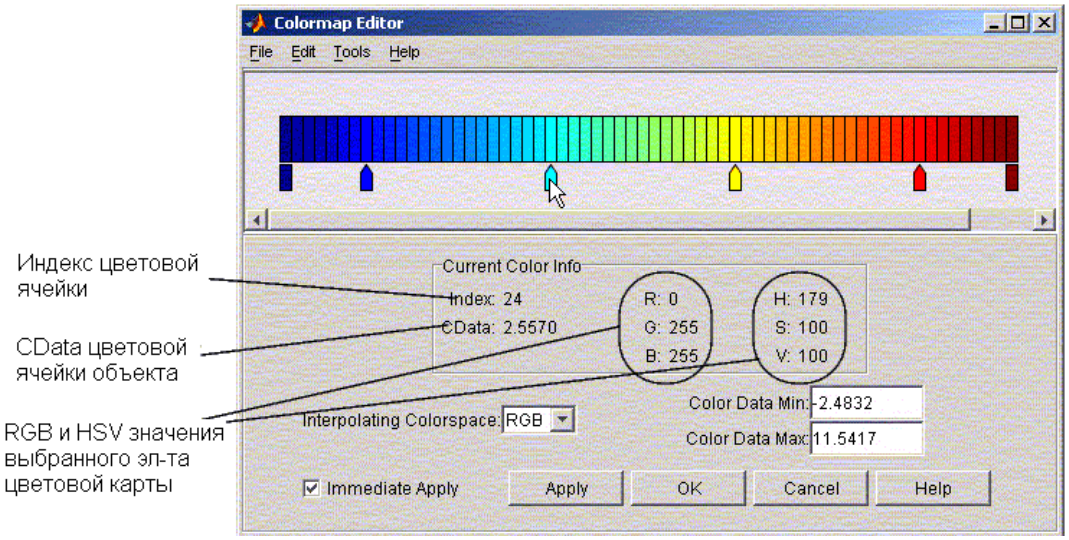

Рис.2.1- Окно редактора цветовой карты

В верхней части окна размещена собственно цветовая карта в виде ленты, составленной из цветных прямоугольников. Под этой лентой размещены цветные указатели в тех местах, где изменяется скорость изменения цветовых составляющих R, G, B. Эти указатели можно перемещать с помощью мыши, при этом их цвет остается неизменным, а изменяются области цветовой карты, примыкающие к перемещаемому указателю (путем линейной интерполяции RGB-значений в этих областях). Двойным щелчком по указателю можно включить режим редактирования цвета указателя. Кроме того, можно удалять существующие указатели и добавлять новые.

ЗАДАНИЕ

1.Проработать все описанные выше примеры.

2.Предположим, что исходное полноцветное изображение слишком темное и имеет низкий контраст. Данное изображение преобразуется в цветовую систему HSV, где V является яркостной составляющей. Для этой составляющей повышается контраст. Затем изображение преобразуется обратно в систему RGB.

```
%Пример демонстрирует подход к фильтрации и
% контрастированию цветных изображений.
%Чтение исходного изображения и вывод его на экран.
RGB=imread('flowers.tif');   % изображение – цветы, но у
%вас может быть иное изображение
imshow(RGB);
%Преобразование в цветовую систему HSV.
HSV=rgb2hsv(RGB);
%Контрастирование составляющей H - яркости изображения.
HSV(:,:,3)=imadjust(HSV(:,:,3), [0.02 0.68], [], 0.7);
%Преобразование в цветовую систему RGB.
RGB=hsv2rgb(HSV);
%Вывод результатов на экран.
figure, imshow(RGB);
```

Обработайте любое малоконтрастное изображение с помощью приведенной выше программы. Подберите параметры контрастирования так, чтобы контрастность результирующего изображения повысилась.

3.В командном окне Matlab наберите команду `Landsatdemo` – в результате на мониторе отобразится демонстрационный пример работы с сложноцветными изображениями. Самостоятельно разберитесь с возможностями и назначением данного примера.

ЛАБОРАТОРНАЯ РАБОТА №3
УЛУЧШЕНИЕ ВИЗУАЛЬНОГО КАЧЕСТВА ЦВЕТНЫХ ИЗОБРАЖЕНИЙ

Цель: Изучение некоторыми команд и функций MatLab, предназначенных для улучшения визуального качества цветных изображений.

Улучшение визуального качества цветных изображений

Большой практический интерес представляет задача улучшения качества цветных изображений. В большинстве случаев под улучшением имеется в виду повышение качества визуального восприятия. Рассмотрим несколько подходов, которые применяются при решении этой задачи. Это – выравнивание гистограммы значений яркостей элементов изображения (эквализация), растяжение динамического диапазона значений яркостей изображения, контрастирование и нечеткое маскирование. Особенностью рассматриваемой задачи является то, что большинство известных подходов разрабатывались для улучшения качества не цветных, а полутоновых изображений. Задача повышения качества цветных изображений, в отличии от полутоновых, имеет свои особенности. Однако, при решении задачи повышения визуального качества цветных изображений будем использовать только этот инструментарий и функции, которые реализованы в приложении Image Processing Toolbox.

Выравнивание гистограммы значений яркостей элементов изображения (эквализация)

Довольно часто исходным необработанным изображениям свойственны яркостные искажения. Причины могут быть самые разные, но в большинстве случаев это объясняется несовершенством аппаратуры формирования видеоданных. В результате, на таких изображениях детали различаются плохо или вообще не различаются. Для повышения контрастности таких изображений используют различные методы видоизменения гистограмм. Приведем пример повышения визуального качества изображения путем выравнивания гистограммы значений яркостей его элементов. Для этого будем использовать встроенную в приложение Image Processing Toolbox функцию `histeq`.

Считаем исходное изображение в рабочее пространство Matlab

```
L=imread('forest.bmp'); %в некоторых версиях это и другие
%изображения имеют расширения tif или png
L=double(L)./255;
```

и визуализируем его.

```
figure, imshow(L);
```

Известно, что в приложении Image Processing Toolbox изображения представляются в цветовом пространстве RGB или в виде индексированных изображений. Сначала реализуем выравнивание гистограмм значений яркостей отдельно по каждой цветовой составляющей в пространстве RGB.

```
L_R=L(:,:,1);L_G=L(:,:,2);L_B=L(:,:,3);
L_R=histeq(L_R);
L_G=histeq(L_G);
L_B=histeq(L_B);
L(:,:,1)=L_R;
L(:,:,2)=L_G;
L(:,:,3)=L_B;
figure, imshow(L);
```

При обработке цветных изображений довольно часто используется цветовая система HSV. В литературе утверждается, что эта цветовая система более адекватно отображает восприятие цвета человеком, чем RGB.

Преобразуем исходное изображение в формат HSV с помощью функции `rgb2hsv`.

```
L_hsv=rgb2hsv(L);
```

Далее реализуем выравнивание цветового тона.

```
L_hsv(:,:,3)=histeq(L_hsv(:,:,3));
```

Для визуализации результата необходимо провести обратное преобразование в цветовое пространство RGB.

```
L=hsv2rgb(L_hsv);
figure, imshow(L);
```

Примечание. *Многие результаты работы здесь и далее не представлены ввиду их малой информативности при печати. При отработки примеров они будут визуализироваться в цвете на экране монитора.*

Растяжение динамического диапазона значений яркостей изображения

Еще одним известным подходом, который широко используется для улучшения визуального качества изображений, является растяжение динамического диапазона значений его яркостей.

Введем исходное изображение и визуализируем его.

```
L=imread('krepost.bmp');
L=double(L)./255;
figure, imshow(L);
```

Исходное изображение представлено в цветовом пространстве RGB. Проведем растяжение динамического диапазона его яркостей по каждой цветовой составляющей.

```
L_R=L(:,:,1);L_G=L(:,:,2);L_B=L(:,:,3);
```

Определим минимальные и максимальные значения яркостей всех цветовых составляющих. Эти данные будут являться исходными для этого метода коррекции градационных искажений.

```
MIN_L_R=min(min(L_R));MIN_L_G=min(min(L_G))
MIN_L_B=min(min(L_B));
MAX_L_R=max(max(L_R));MAX_L_G=max(max(L_G))
MAX_L_B=max(max(L_B));
```

Характер преобразований определяется параметром α. Если α=1 , то преобразование линейное. Однако довольно часто, особенно если исходное изображение содержит темные или светлые области большой площади, более эффективно применение нелинейных преобразований при $\alpha \neq 1$.

```
a=.5;
L_R=((L_R-MIN_L_R)./(MAX_L_R-MIN_L_R)).^a;
L_G=((L_G-MIN_L_G)./(MAX_L_G-MIN_L_G)).^a;
L_B=((L_B-MIN_L_B)./(MAX_L_B-MIN_L_B)).^a;
L(:,:,1)=L_R;
L(:,:,2)=L_G;
L(:,:,3)=L_B;
```

Визуализируем результат преобразования.

```
figure, imshow(L);
```

Теперь, как и в предыдущем примере, представим изображение в цветовой системе HSV

```
L_hsv=rgb2hsv(L);
```

и реализуем эту же процедуру растяжения динамического диапазона значений цветового тона.

```
a=.5;
L_hsv(:,:,3)=((L_hsv(:,:,3)-
min(min(L_hsv(:,:,3)))./(max(max(L_hsv(:,:,3)))-
min(min(L_hsv(:,:,3))))).^a;
```

```
L=hsv2rgb(L_hsv);
```

Представим результат.

```
figure, imshow(L);
```

Контрастирование цветных изображений

Рассмотрим еще один подход, который используется для повышения визуального качества изображений – контрастрование. При реализации этого подхода будем использовать функцию `imfilter`.

Считаем исходное изображение и визуализируем его.

```
L=imread('canoe.bmp');
L=double(L)./255;
figure, imshow(L);
```

Каждую цветовую составляющую изображения, которое представлено в формате RGB, будем обрабатывать отдельно.

```
L_R=L(:,:,1);L_G=L(:,:,2);L_B=L(:,:,3);
```

Для этого выберем маску фильтра, повышающую контраст изображения

```
h = fspecial('unsharp');
```

и применим ее к каждой цветовой составляющей.

```
L_R=imfilter(L_R,h);
L_G=imfilter(L_G,h);
L_B=imfilter(L_B,h);
L(:,:,1)=L_R;
L(:,:,2)=L_G;
L(:,:,3)=L_B;
```

Визуализируем результат контрастирования изображения, представленного в цветовом пространстве RGB.

```
figure, imshow(L);
```

Представим это же изображение в цветовой системе HSV и проведем аналогичные преобразования.

```
L_hsv=rgb2hsv(L);
L_hsv(:,:,3)=imfilter(L_hsv(:,:,3),h);
L=hsv2rgb(L_hsv)
```

Представим результат проведенных преобразований.

```
figure, imshow(L);
```

Методы нечеткого маскирования

Рассмотрим еще один класс методов – методы нечеткого маскирования, которые также применяются для повышения визуального качества изображений. Сначала считаем и визуализируем исходное изображение.

```
L=imread('hestain.bmp'); L=double(L)./255; figure, imshow(L);
```

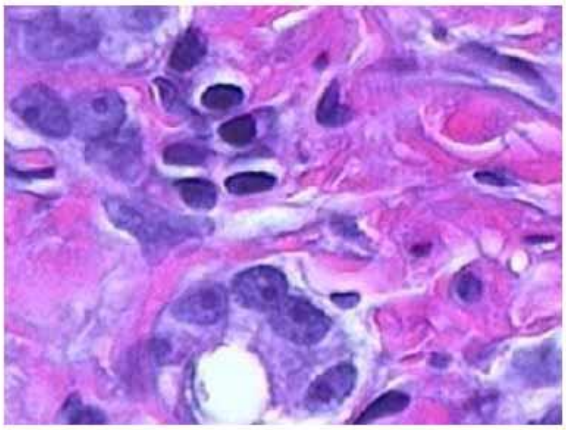

Рис.3.1

Далее выделим отдельно каждую цветовую составляющую и выполним ее преобразование методом нечеткого маскирования

```
L_R=L(:,:,1);L_G=L(:,:,2);L_B=L(:,:,3);
MEAN_L_R=mean(mean(L_R));MEAN_L_G=mean(mean(L_G))
MEAN_L_B=mean(mean(L_B)); a=3;
L_R=MEAN_L_R+a.*(L_R-MEAN_L_R);
L_G=MEAN_L_G+a.*(L_G-MEAN_L_G);
L_B=MEAN_L_B+a.*(L_B-MEAN_L_B); L(:,:,1)=L_R;
L(:,:,2)=L_G; L(:,:,3)=L_B;
```

Представим результат преобразования.

```
figure, imshow(L);
```

Далее, аналогично, как и в предыдущих примерах, представим изображение в цветовом пространстве HSV

```
L_hsv=rgb2hsv(L);
```

и применим метод нечеткого маскирования.

```
MEAN_L_hsv=mean(mean(L_hsv(:,:,3)));
L_hsv(:,:,3)=MEAN_L_hsv+a.*(L_hsv(:,:,3)-MEAN_L_hsv);
L=hsv2rgb(L_hsv);
```

Представим результат проведенного преобразования.

```
figure, imshow(L);
```

ЗАДАНИЕ

Проработать все описанные выше примеры и оформить отчет.

ЛАБОРАТОРНАЯ РАБОТА №4
ОСНОВЫ СТАТИСТИЧЕСКОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Цель: Знакомство с командами и функциями языка MatLab, предназначенными для предварительной и статистической обработки изображений.

В данной работе приведены некоторые элементы статистической обработки изображений. В примере показана статистическая обработка изображения, а также предварительная обработка для получения улучшенного результата, которая включает формирование равномерного фона на изображении и преобразование изображения в бинарное. Пример состоит из следующих основных шагов:

Шаг 1:	Считывание и отображение изображения
Шаг 2:	Оценка и аппроксимация значений пикселов фона
Шаг 3:	Аппроксимация и просмотр поверхности фона
Шаг 4:	Создание изображения с равномерным фоном
Шаг 5:	Повышение контраста изображения
Шаг 6:	Создание бинарного изображения
Шаг 7:	Определение числа объектов на изображении
Шаг 8:	Анализ изображения
Шаг 9:	Отображение матрицы меток как псевдоцветового индексного изображения
Шаг 10:	Измерение свойств объектов на изображении
Шаг 11:	Вычисление статистических данных об объектах на изображении

Считывание и отображение изображения

Очистим рабочее пространство MATLAB, закроем все открытые окна просмотра и закроем все открытые средства Image Tools.

```
clear, close all, imtool close all
```

Считаем и визуализируем полутоновое изображение, например, `rice.png`.

```
I = imread('rice.png');
imshow(I)
```

Построенное командой imshow изображение показано на рис.4.1.

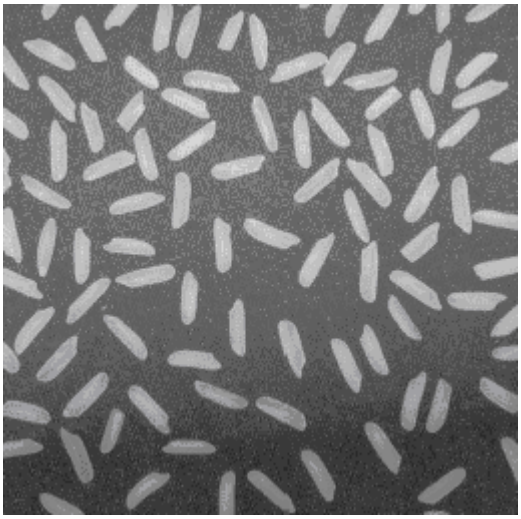

Рис. 4.1 - Полутоновое изображение rice.png

Оценка и аппроксимация значений пикселей фона

Рассматриваемое изображение характеризуется неравномерной засветкой фона по полю изображения. Поэтому сначала с помощью операции морфологического раскрытия оценим интенсивность фона. При выполнении операции морфологического раскрытия используется структурный элемент.

В примере используется функция `imopen`, которая выполняет морфологическое раскрытие. При этом также используется функция `strel` для создания структурного элемента в виде диска с радиусом 15.

```
background = imopen(I,strel('disk',15));
```

Для просмотра и оценки фона используются следующие команды

```
figure, imshow(background)
```

Аппроксимация и просмотр поверхности фона

Используем команду `surf` для отображения поверхности фона (см. рис.4.2). С помощью команды `surf` создается цветная параметрическая поверхность, которая дает возможность просматривать прямоугольную область исследуемого изображения. Функция `surf` работает с данными, которые представлены в формате double. Поэтому, прежде чем применять эту функцию, данные необходимо перевести в формат double.

```
figure, surf(double(background(1:8:end,1:8:end))),zlim([0
255]);
set(gca,'ydir','reverse');
```

В данном примере отображается только каждый восьмой пиксель по каждому направлению. Существует также возможность установки масштаба и других параметров.

На основе этого можно проводить анализ фона исследуемого изображения.

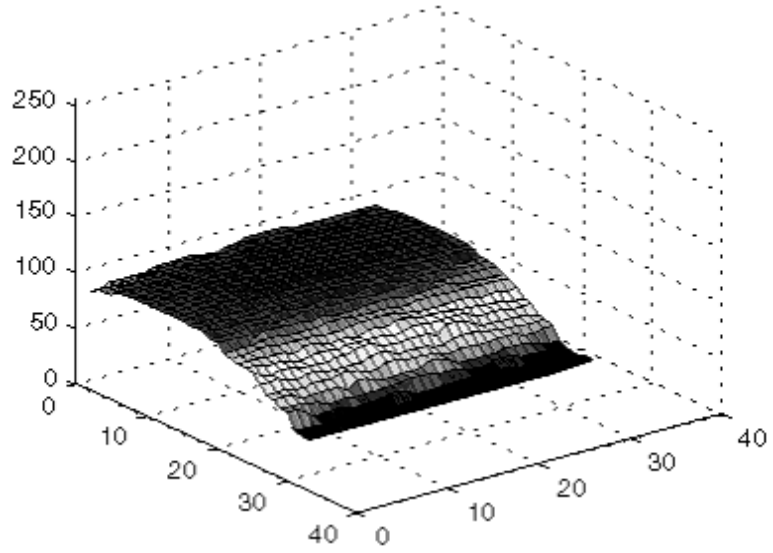

Рис. 4.2

Создание изображения с равномерным фоном

Для создания изображения с более равномерным фоном, вычтем изображение фона background из исходного изображения I.

```
I2 = imsubtract(I,background);
```

Отобразим полученное изображение с более равномерным фоном (рис.4.3).

```
figure, imshow(I2)
```

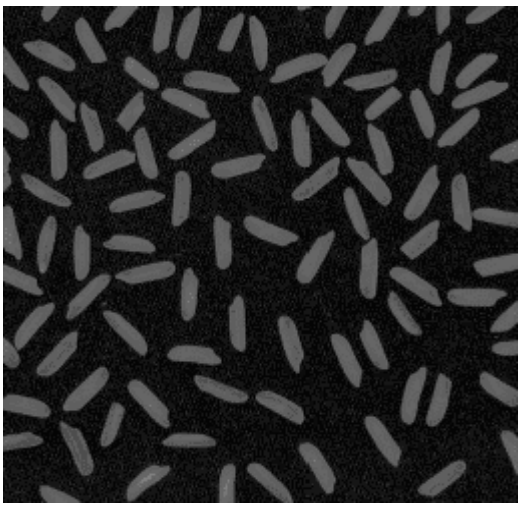

Рис. 4.3 - Изображение с равномерным фоном

Улучшение контраста на обрабатываемом изображении

После вычитания, полученное изображение будет иметь более равномерный фон и будет более темным. Используем функцию `imadjust` для повышения контраста изображения.

```
I3 = imadjust(I2);
```

Функция `imadjust` увеличивает контраст изображения путем растяжения значений интенсивностей динамического диапазона. Для более детальной информации см. описание функции imadjust. Отобразим улучшенное изображение I3 (рис.4.4).

```
figure, imshow(I3);
```

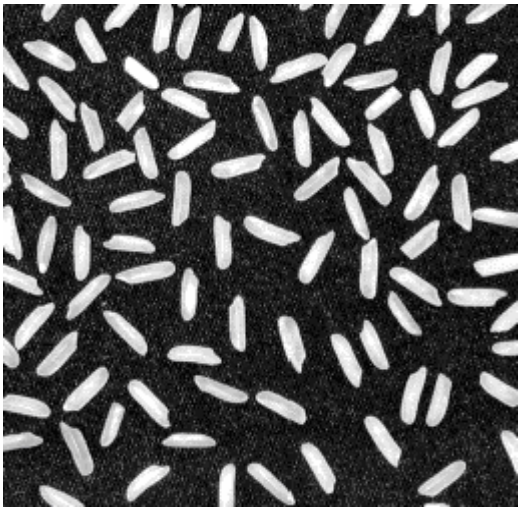

Рис. 4.4 - Изображение после коррекции интенсивностей

Создание бинарного изображения

Бинарное изображение можно создать, используя функцию `thresholding`. Однако функция `graythresh` автоматически определяет подходящий порог, который используется для преобразования полутонового изображения в бинарное. Функция `im2bw` выполняет это преобразование.

```
level = graythresh(I3);
bw = im2bw(I3,level);
figure, imshow(bw)
```

Возвращаемое функцией `im2bw` бинарное изображение bw представлено в формате logical. В этом можно убедится, воспользовавшись функцией `whos`. Приложение Image Processing Toolbox использует логические массивы для представления бинарных изображений.

Определение числа объектов на изображении

После преобразования изображения в бинарное, можно использовать функцию `bwlabel` для определения числа объектов (зерен риса) на изображении. Функция `bwlabel` отмечает все компоненты на бинарном изображении bw и возвращает их число в виде значения numObjects.

```
[labeled,numObjects] = bwlabel(bw,4);
numObjects
ans =
   101
```

Точность результата зависит от некоторых факторов, включая:

1. размер объектов;
2. соприкасаются ли между собой объекты (в этом случае они могут определяться как один объект);
3. точность аппроксимации фона.
4. выбор связности.

Анализ матрицы меток

Для проведения анализа формирования матрицы меток, возвращаемой функцией `bwlabel`, рассмотрим вопросы получения значений пикселей изображения. Для этого существует несколько путей. Например, используя функцию `imcrop`, можно просматривать небольшие порции изображения. Другой путь состоит в использовании приложения Pixel Region tool для просмотра значений пикселей.

Отобразим матрицу меток с использованием функции `imshow`:

```
figure, imshow(labeled);
```

Запуск приложения Pixel Region tool:

```
impixelregion
```

По умолчанию, оно автоматически связывает себя с изображением в текущем окне просмотра. Приложение Pixel Region tool рисует прямоугольник с центром в видимой части изображения. Расположение этого прямоугольника определяет, какие пиксели будут отображаться в Pixel Region tool. В то же время при перемещении прямоугольника значения отображаемых в окне Pixel Region tool пикселей обновятся.

На рис.4.5 представлен Image Viewer с прямоугольником Pixel Region, размещенным на границе двух рисовых гранул. Отметим, что значения пикселей фона и отдельных гранул являются одинаковыми. Так, в частности, значение фона равно 0.

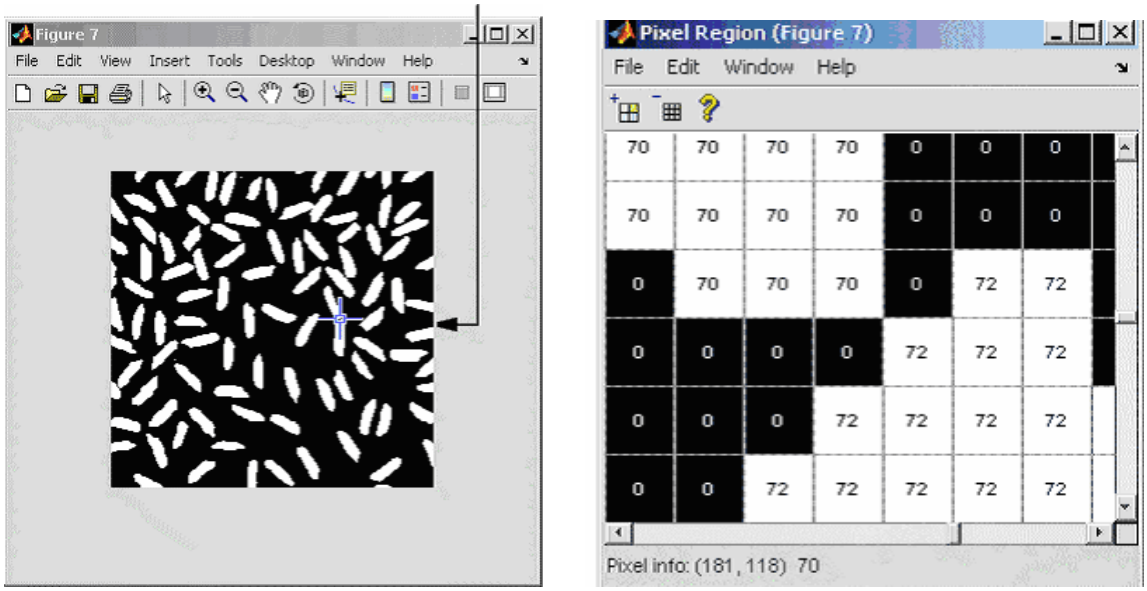

Рис.4.5 - Просмотр матрицы средствами Pixel Region Tool

Отображение матрицы меток в виде псевдоцветного индексного изображения

Один из возможных путей отображения матрицы меток состоит в использовании псевдоцветного индексного изображения. В псевдоцветном изображении числа, которые идентифицируют каждый объект в матрице меток, отображаются разным цветом, который связан с соответствующей палитрой цветов. Такое представление позволяет различать различные объекты.

Для просмотра матрицы меток используется функция `label2rgb`. Эта функция применяется при создании палитры, цвета фона и цвета каждого объекта изображения (рис.4.6).

```
pseudo_color = label2rgb(labeled, @spring, 'c',
'shuffle');
imshow(pseudo_color);
```

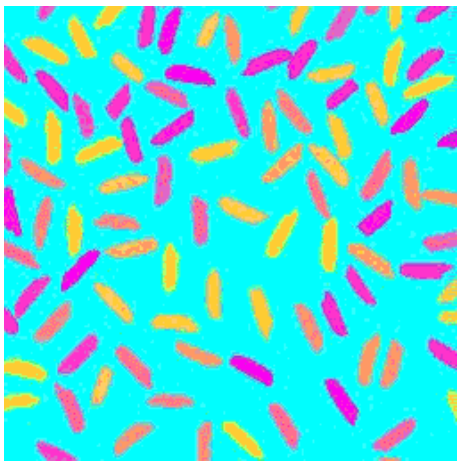

Рис.4.6 - Представление матрицы меток в виде псевдоцветного изображения

Измерение свойств объектов на изображении

Функция `regionprops` применяется для измерения свойств объектов в некоторой окрестности изображения и представляет результат в виде массива. Если применить это к изображению с отмеченными компонентами, то создается структура для каждой компоненты.

В этом примере используется функция `regionprops` для создания структурированного массива, содержащего некоторые основные свойства. Если установить свойствам параметр `'basic'`, тогда функция `regionprops` возвращает три измеренных параметра: площадь, центроид (или центр масс) и ограничивающий прямоугольник (локальная окрестность). В данном случае ограничивающий прямоугольник представляет собой локальную окрестность, отображающую гранулы риса.

```
graindata = regionprops(labeled,'basic')
```

Система MATLAB выдает такой результат

```
graindata =
101x1 struct array with fields:
    Area
    Centroid
    BoundingBox
```

Для поиска области с 51-м отмеченным компонентом, необходимо найти поле Area field и его 51 элемент в структуре массива graindata.

```
graindata(51).Area
```

В этом случае результат будет таким

```
ans =
   140
```

Для поиска наименее возможного прямоугольника и центроида (центра масс) некоторых компонентов используется следующий код:

```
graindata(51).BoundingBox, graindata(51).Centroid
ans =
107.5000    4.5000   13.0000   20.0000

ans =
114.5000   15.4500
```

Вычисление статистических свойств объектов изображения

Функции системы MATLAB могут использоваться для вычисления статистических свойств объектов. Сначала используется функция `max` для

поиска наибольшего зерна. (В нашем примере наибольшими являются два зерна риса, которые соприкасаются.)

```
max([graindata.Area])
```

Результат представляется в виде

```
ans =
404
```

Далее с помощью функции `find` найдем те компоненты, которыми отмечены зерна риса с этой площадью.

```
biggrain = find([graindata.Area]==404)
```

Результат представляется в виде

```
biggrain =
 59
```

Найдем средний размер всех зерен риса.

```
mean([graindata.Area])
```

Результат будет представлен в виде

```
ans =
175.0396
```

Построим гистограмму распределения зерен риса по их размеру (при этом будем использовать 20 позиций при градации по оси площади). Из гистограммы видно (см. рис.4.7), что основная часть зерен риса имеет площадь от 150 до 250 пикселей.

```
hist([graindata.Area],20)
```

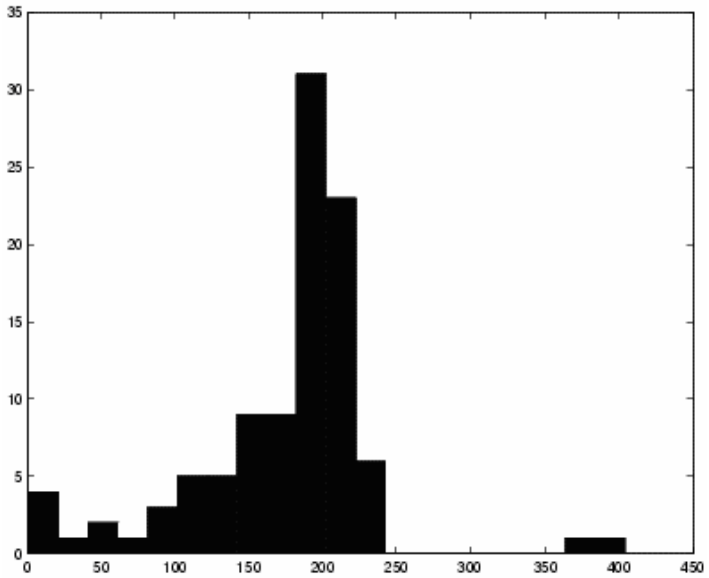

Рис.4.7

ЗАДАНИЕ

Проработать все описанные выше примеры и оформить отчет.

ЛАБОРАТОРНАЯ РАБОТА №5
АНАЛИЗ ИЗОБРАЖЕНИЙ

Цель. Изучение некоторых функций, которые используются при анализе изображений, а также их текстуры.

Выделение границ

Для определения границ объектов изображения можно использовать функцию `edge`. В качестве одного из критериев для определения края используется резкий перепад интенсивностей пикселей изображения. Отметим, что некоторые методы выделения границ, в зависимости от своих особенностей, в большей степени выделяют вертикальные, горизонтальные или все границы одинаково.

Одним из наиболее эффективных методов выделения границ является метод Канни [3]. Метод Канни отличается от других известных методов тем, что при определении границ использует два порога (для слабых и сильных границ). Слабые границы отмечаются в результирующем изображении только тогда, когда они соединены с сильными. Для зашумленных изображений данный метод обеспечивает наилучшее обнаружение границ по сравнению с остальными методами, но требует существенно большего времени.

Рассмотрим пример применения метода выделения границ Канни, а также сравним его с другим известным методом выделения границ:

Считаем и отобразим исходное изображение.

```
I = imread('coins.png');
imshow(I)
```

Рис.5.1

Далее это изображение обработаем операторами выделения границ Собела [2, 3] и Канни с последующей визуализацией (рис.5.2).

```
BW1 = edge(I,'sobel');
BW2 = edge(I,'canny');
imshow(BW1)
figure, imshow(BW2)
```

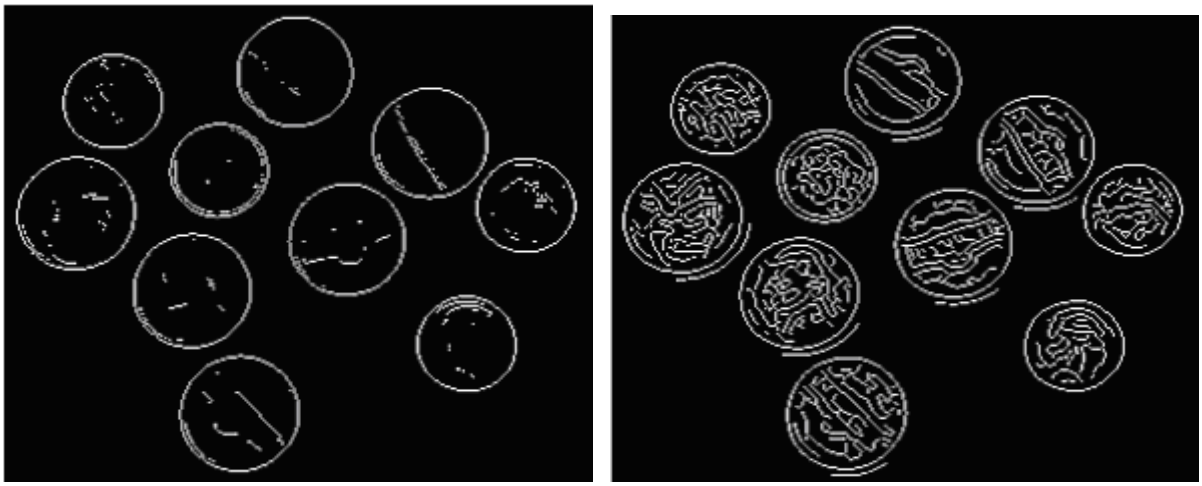

Рис.5.2 – Изображения после обработки операторами Собела (слева) и Канни (справа)

Отслеживание границ

Приложение включает две функции, которые могут быть использованы для поиска границ объектов на бинарном изображении. Это функции `bwtraceboundary` и `bwboundaries`.

Функция `bwtraceboundary` возвращает координаты строк и столбцов всех пикселей, которые принадлежат границе объектов изображения. Также необходимо описать начальную (стартовую) точку при отслеживании границ объектов изображения. Функция `bwboundaries` также возвращает координаты сток и столбцов всех пикселей, которые принадлежат границе объектов изображений.

Для обеих функций ненулевые пиксели на бинарном изображении принадлежат объекту, а пиксели со значением 0 составляют фон. Рассмотрим пример применения функции `bwtraceboundary` для отслеживания границ некоторого объекта бинарного изображения, а затем используем функцию `bwboundaries` для отслеживания границ всех объектов изображения.

Считаем и визуализируем изображение.

```
I = imread('coins.png');
imshow(I)
```

Преобразуем это изображение в бинарное. Функции `bwtraceboundary` и `bwboundaries` работают только с бинарными изображениями.

```
BW = im2bw(I);
imshow(BW)
```

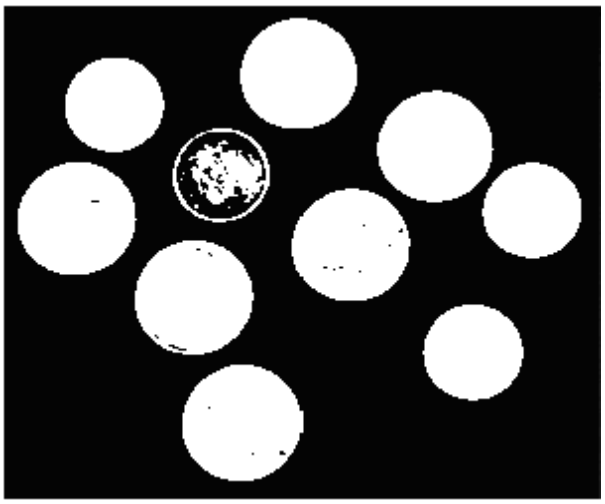

Рис.5.3

Определим координаты строки и столбца пикселя на границе объекта. Функция bwboundary использует эту точку в качестве начальной при отслеживании границ.

```
dim = size(BW)
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)))
```

Функция bwtraceboundary выполняет отслеживание границ от начальной точки. В качестве обязательных аргументов необходимо ввести само бинарное изображение, координаты начальной точки и направление первого шага. В примере использовано северное направление ('N').

```
boundary = bwtraceboundary(BW,[row, col],'N');
```

Отобразим сначала исходное полутоновое изображение, а потом координаты, которые возвращаются функцией `bwtraceboundary`.

```
imshow(I)
hold on;
plot(boundary(:,2),boundary(:,1),'g','LineWidth',3);
```

Рис.5.4

Для отслеживания границ всех монет на изображении используется функция `bwboundaries`. По умолчанию функция `bwboundaries` определяет границы всех объектов на изображении, включая вложенные объекты. В нашем примере на бинарном изображении некоторые монеты содержат темные области, которые функция `bwboundaries` интерпретирует как отдельные объекты. Для того, чтобы функция `bwboundaries` отслеживала только границы монет, используется функция imfill для заполнения областей внутри каждой монеты.

```
BW_filled = imfill(BW,'holes');
boundaries = bwboundaries(BW_filled);
```

Функция `bwboundaries` возвращает массив значений, где каждая ячейка содержит координаты пикселей объекта изображения. Отобразим границы всех монет на исходном полутоновом изображении, используя координаты, возвращаемые функцией `bwboundaries`.

```
for k=1:10
   b = boundaries{k};
   plot(b(:,2),b(:,1),'g','LineWidth',3);
end
```

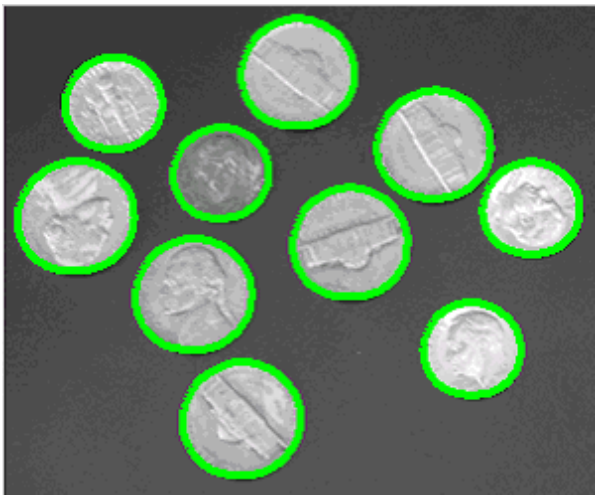

Рис.5.5

Выбор первого шага и направления при отслеживании границ

Как уже было сказано ранее для некоторых объектов в качестве исходных аргументов необходимо указывать начальную точку и направление отслеживания контура. Например, если объект содержит отверстия и в этой части объекта была выбрана начальная точка, тогда отслеживание контура может производиться как по внешней части объекта, так и по внутренней части отверстия. Это зависит от выбора первого шага (рис.5.6, 5.7).

Рассмотрим пример, где от выбора направления первого шага зависит, будет ли проводится отслеживание по внешнему или по внутреннему контуру. По умолчанию установлена 8-связность.

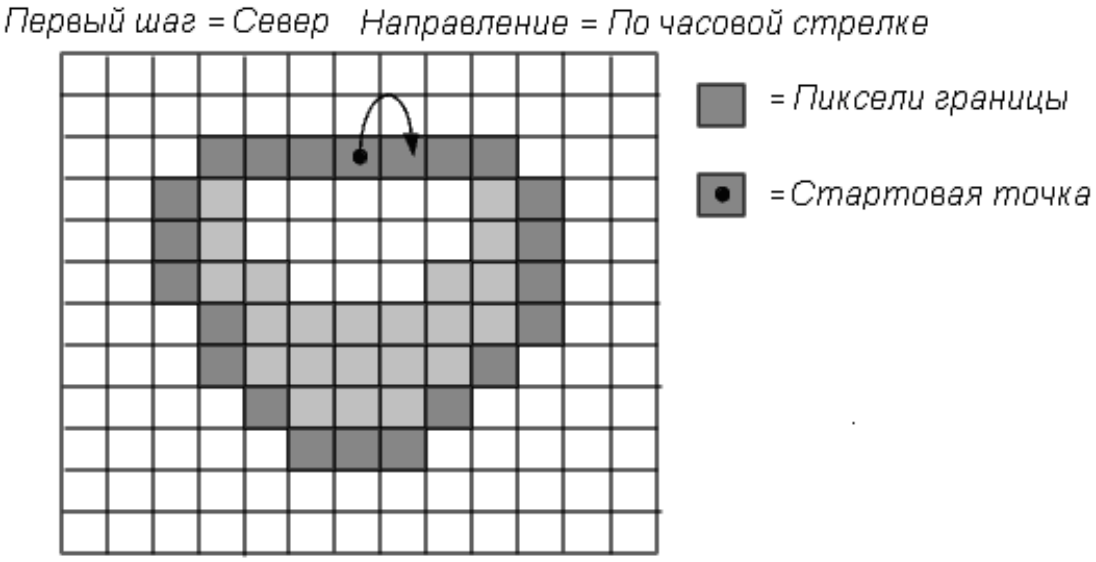

Рис.5.6

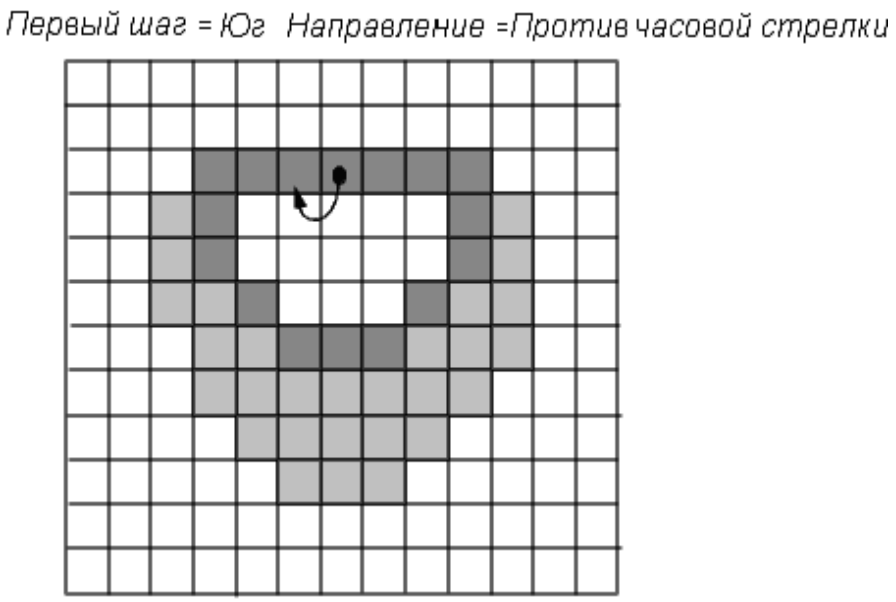

Рис.5.7

Влияние первого шага и направления на отслеживание границ

Детектирование линий с использованием преобразования Хоха

Приложение Image Processing Toolbox включает функции, которые выполняют преобразования Хоха (hough, houghpeaks, houghlines). Функция `hough` выполняет стандартное преобразование Хоха (Standard Hough Transform (SHT)). Преобразование Хоха [2, 3] разработано специально для детектирования линий. В этом преобразовании используется параметрическое представление линий: rho = x*cos(theta) + y*sin(theta). Переменная rho представляет собой расстояние от центра координат до линии вдоль вектора, перпендикулярного к линии. Параметр theta представляет собой угол между осью x и этим вектором. Функция `hough` генерирует параметры пространственной матрицы, где строки и столбцы соответствуют значениям rho и theta.

Функция `houghpeaks` выполняет поиск значений пиков в этом пространстве, которое представляет потенциальные линии на исходном изображении.

Функция `houghlines` выполняет поиск конечных точек линий в соответствии с пиками, которые получены при преобразовании Хоха и автоматически заполняет небольшие разрывы.

Рассмотрим пример использования этой функции при детектировании линий на изображении. Для этого сначала считаем изображение в рабочее пространство MATLAB.

```
I = imread('circuit.tif');
```

Развернем и вырежем изображение (рис.5.8).

```
rotI = imrotate(I,33,'crop');
```

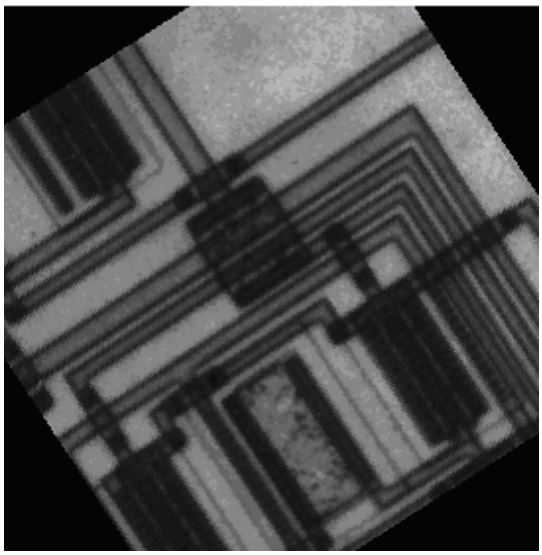

Рис.5.8

Поиск границ на изображении (рис.5.9).

```
BW = edge(rotI,'canny');
```

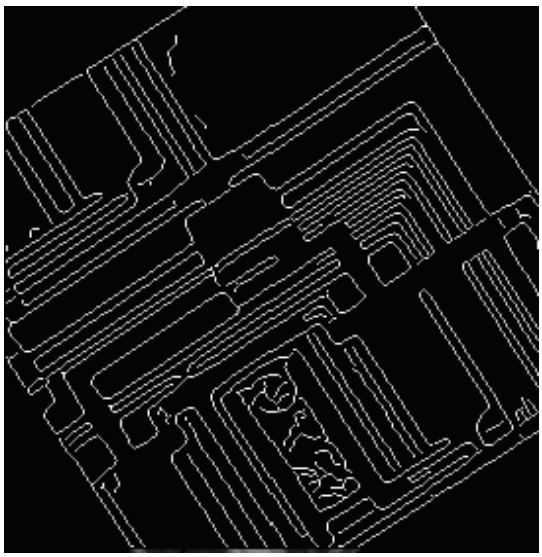

Рис.5.9

Выполнение преобразований Хоха с использованием функции hough.

```
[H,theta,rho] = hough(BW);
```

Отображение результата преобразований (рис.5.10).

```
imshow(H,[],'XData',theta,'YData',rho,...
        'InitialMagnification','fit');
xlabel('\theta'), ylabel('\rho');
```

```
axis on, axis normal, hold on;
```

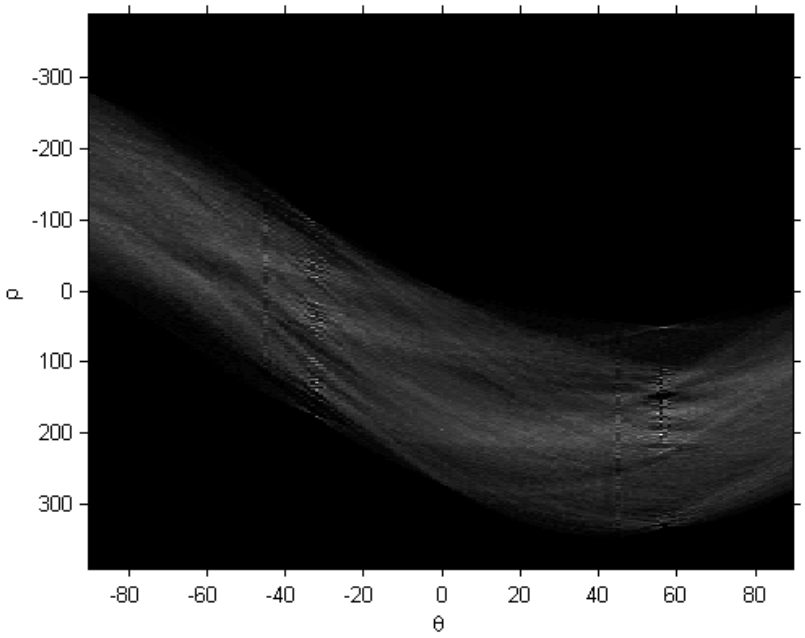

Рис.5.10

Поиск пиков в матрице преобразований Хоха H с помощью функции houghpeaks.

```
P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
```

Отображение пиков.

```
x = theta(P(:,2));
y = rho(P(:,1));
plot(x,y,'s','color','white');
```

Поиск линий на изображении.

```
lines =
houghlines(BW,theta,rho,P,'FillGap',5,'MinLength',7);
```

Наложение линий на исходное изображение.

```
figure, imshow(rotI), hold on
max_len = 0;
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

% Отображение начала и конца линий
plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
```

```
    % Определение конечной точки
    len = norm(lines(k).point1 - lines(k).point2);
    if ( len > max_len)
       max_len = len;
       xy_long = xy;
    end
end
% Окрашивание линий
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan');
```

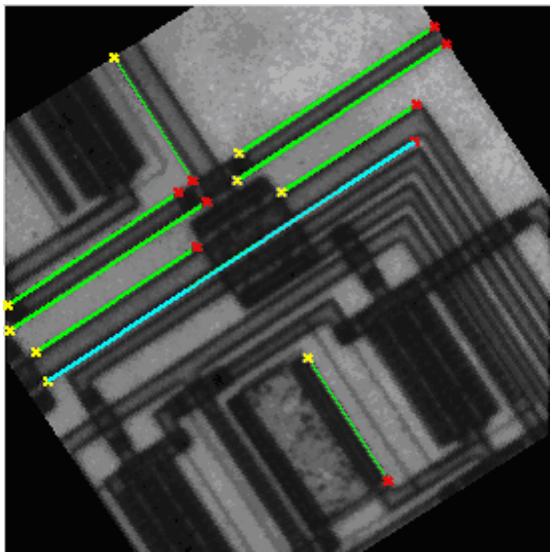

Рис.5.11

ЗАДАНИЕ

1. Проработать все описанные выше примеры.
2. Оформить отчет о проделанной работе согласно ГОСТ и требованиям преподавателя.

ЛИТЕРАТУРА

1. Дьяконов В., Абраменкова И. MATLAB. Обработка сигналов и изображений // Специальный справочник. – СПб.: Питер, 2002. – 608 с.
2. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB // Москва.: Техносфера, 2006. – 616 с.
3. Гонсалес Р., Вудс Р. Цифровая обработка изображений // Москва.: Техносфера, 2005. – 1072 с.
4. Даджион Д., Мерсеро Р. Цифровая обработка многомерных сигналов. - М.: Мир, 1988.
5. Ярославский Л.П. Введение в цифровую обработку изображений. - М.: Сов. радио, 1979.
6. Прэтт У. Цифровая обработка изображений. Кн.1. - М.: Мир, 1982.
7. Методы передачи изображений. Сокращение избыточности / У.К. Прэтт, Д.Д. Сакрисон, Х.Г.Д. Мусманн и др. Под ред. У.К.Прэтта. -М.: Радио и связь, 1983.
8. Левин Б.Р. Теоретические основы статистической радиотехники. Кн.2. - М.: Сов. радио, 1975.
9. Woods J.W. Two-Dimensional Digital Signal Processing 1. Berlin e.a., 1981.
10. Васильев К.К., Крашенинников В.Р. Методы фильтрации многомерных случайных полей. - Саратов: Сарат. гос. ун-т, 1990.
11. Быстрые алгоритмы в цифровой обработке изображений / Под ред. Т.С. Хуанга. - М.: Радио и связь, 1984.
12. Борн М., Вольф Э. Основы оптики.- М.: Наука, 1970.
13. Беллман Р. Введение в теорию матриц. / Пер. С англ.; Под ред. В.Б. Лидского . - М.: Наука, 1976.
14. Реконструкция изображений: Пер. с англ. / Под ред. Г.Старка. - М.: Мир, 1992.
15. Василенко Г.И., Тараторин А.М. Восстановление изображений. - М.: Радио и связь, 1986.
16. Рабинер Л., Гоулд Б. Теория и применение цифровой обработки сигналов. - М.: Мир, 1978.
17. Бейтс Р., Мак-Донелл М. Восстановление и реконструкция изображений. - М.: Мир, 1989.
18. Хорн Б.К.П. Зрение роботов.- М.: Мир, 1989.
19. Марр Д. Зрение: информационный подход к изучению представления и обработки зрительных образов. -М.: Радио и связь, 1987.