МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

|  |
| --- |
|  |



Кафедра «Вычислительная техника»

Кафедра «Вычислительная техника»

системы анализа данных  
космического зондирования

*Методические указания*

Самара

Самарский государственный технический университет

2015

Публикуется по решению Методического совета ФАИТ

УДК 004(07)

ББК 32.97

**Системы анализа данных космического зондирования:** методические указания */ Сост. Б.В. Мартемьянов.* – Самара: Самар. гос. техн. ун-т, 2015. – 71 с.: ил.

Изложены теоретические основы дисциплины, необходимые для выполнения цикла лабораторных работ. Сформулированы задания на лабораторные работы, методики их выполнения и контрольные вопросы по каждой работе. Изложение материала сопровождается наглядными иллюстрациями и таблицами.

Для обучающихся по магистерской программе направления подготовки 09.04.01.68 – «Информатика и вычислительная техника».

© Б.В. Мартемьянов, 2015

© Самарский государственный   
 технический университет, 2015

# Введение

Данные методические указания являются приложением к рабочей программе дисциплины «Системы анализа данных космического зондирования», ориентированной на магистрантов. В отличие от инженерного образования квалификация магистр является ученой степенью и поэтому предполагает, что носитель такой степени должен уметь не только использовать готовые программные средства для решения прикладных задач, но и быть готовым к разработке собственных методов, алгоритмов и программных продуктов, ориентированных на работу с объектами, изучаемыми в рамках дисциплины.

Дисциплина «Системы анализа данных космического зондирования» предполагает, что объектами изучения, исследования и обработки являются изображения, полученные космическими аппаратами (КА) дистанционного зондирования Земли (ДЗЗ). С учетом сказанного выше, студент-магистрант, обучающийся по направлению "Информатика и вычислительная техника", должен уметь работать с файлами, содержащими изображения ДЗЗ, а так же владеть навыками по разработке базовых операций визуализации и обработки этих изображений. Поэтому в рамках цикла лабораторных работ внимание сосредоточено на разработке собственных программных средств для работы с изображениями ДЗЗ. При этом предполагается, что исходные изображения представлены в формате tiff. Именно в таком формате представляются изображения ДЗЗ высокого разрешения как спектрозональные, так и панхроматические.

# Лабораторная работа №1 Визуализация изображений ДЗЗ

**Цель работы**: ознакомление с форматом tiff, приобретение навыков программирования средств для работы с файлами и изображениями большого размера.

# 1.1 Изображения ДЗЗ высокого разрешения

# 1.1.1 Особенности изображений ДЗЗ высокого разрешения

Отличительными особенностями изображений ДЗЗ являются их большой размер и сравнительно высокое разрешение изображения по глубине. Так, отдельный файл изображения, сформированного КА ДЗЗ «Ресурс-ДК1», содержит информацию о яркости пикселей монохромного (спектрозонального, либо панхроматического) изображения и может содержать сотни тысяч строк по 6104 пиксела в строке. При этом в файле на каждый пиксел отводится два байта, из которых информативными являются 10 младших бит. Такое разрешение изображения по глубине поддерживает диапазон яркостей пикселов от 0 до 1023 единиц.

С точки зрения удобства наблюдения монохромного изображения на экране монитора его лучше представлять как полутоновое в оттенках серого цвета, то есть от черного до белого.

Видео система компьютера обеспечивает наилучшее разрешение изображения по глубине в режиме True Color. При этом цвет отдельного пикселя кодируется с помощью 4-х байт, используя так называемую *RGB* модель цвета. Структура кода пикселя в режиме True Color показана на рисунке 1.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Резервный байт | Байт интенсивности голубого (***B***lue) цвета | Байт интенсивности зеленого (***G***reen) цвета | Байт интенсивности красного (***R***ed) цвета |

Рисунок 1.1 – Структура кода цвета пикселя в режиме True Color

В *RGB* модели три цвета *Red* (красный), *Green* (зеленый) и *Blue* (голубой) называются основными цветами. Все прочие цвета получаются смешиванием в определенных пропорциях этих основных цветов. Для получения на экране монитора монохромного изображения в оттенках серого цвета необходимо все три основных цвета брать в пропорции 1:1:1. Поэтому при распаковке содержимого файла с монохромным изображением яркость каждого пиксела записывается в приведенную на рисунке 1.1 структуру в качестве интенсивности каждого из основных цветов. Понятно, что при этом возникает противоречие между двухбайтовыми (16-ти битными) кодами яркости пикселей, записанными в файле, с 8-ми битными кодами яркости цветовых составляющих пикселей, поддерживаемыми видео системой компьютера. Рассмотрим это противоречие подробней.

Пусть, например, яркость пикселя монохромного (в оттенках серого) изображения равна 200 единицам. В этом случае каждый из трех байт кода цвета пикселя получит одно и то же значение 200, которое в двоичном коде равно 11001000. В результате 3 байта кода цвета будут содержать код 11001000 11001000 11001000. Поскольку максимальное значение кода для отдельного байта равно 255, то любой пиксель, которому в файле сопоставлено значение не более, чем 255, на экране монитора будет отображаться правильно в соответствии с заданной в файле яркостью. Для множества таких пикселей на экране будет правильно отображаться их относительная яркость.

Теперь рассмотрим случай, когда в файле яркости пикселей могут принимать значения, соответствующие диапазону значений 10-битного кода. Стандартные пакеты, ориентированные на работу с растровой графикой, не предполагают подобной ситуации, поэтому при упаковке яркости 10-,битного кода пиксела в *RGB* структуру допускают сильные искажения числовых значений интенсивностей основных цветов. Конкретные искажения зависят от особенностей алгоритма упаковки кода яркости в *RGB* структуру. Проиллюстрируем работу одного из возможных вариантов такого алгоритма.

Пусть 10-разрядный двоичный код яркости пикселя равен 1100110000 (=816). На рисунке 1.2 показано расположение трех экземпляров этого кода после выравнивания младшего разряда кода по младшему разряду байтов каждого из трех основных цветов. На рисунке предполагается, что не заполненным разрядам 24-разрядных кодов соответствуют значения 0. Эти разряды не заполнены для большей наглядности иллюстрации расположения трех экземпляров заданного 10-разрядного кода.

На рисунке видно, что два старших бита яркости красного цвета попали на места расположения двух младших бит яркости зеленого цвета. Два старших бита яркости зеленого цвета попали на места расположения двух младших бит яркости голубого цвета. Два старших бита яркости голубого цвета оказались за левой границей 24-разрядного кода цвета пикселя. Если все показанные коды сложить арифметически, либо логически (в данном случае результат будет одинаковым) то вместо желаемого, но невозможного, случая *R* = *G* = *B* = 1100110000 = 816 получим:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Байт ***B*** | | | | | | | | Байт ***G*** | | | | | | | | Байт ***R*** | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Результат упаковки 10-разрядного кода в *RGB* структуру: | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

Рисунок 1.2 – Пример упаковки 10-битного кода яркости в *RGB* cтруктуру

- R = 00110000 = 48 (утрачено 816 – 48 = 768 единиц яркости);

- G = 00110011 = 51 (утрачено 816 – 51 = 765 единиц яркости);

- B = 00110011 = 51 (утрачено 816 – 51 = 765 единиц яркости).

Таким образом, после упаковки заданной яркости в *RGB* структуру сохранилось около 6% заданной яркости: 50/816 ≈ 0,06. Кроме того, имеет место отклонение от серой палитры, поскольку *R ≠ G и R ≠ B*.

Понятно, что подобные потери старших разрядов кодов яркостей пикселов недопустимы. Исключить такие потери можно единственным способом: перед упаковкой яркостей пикселов в *RGB* структуру 10-битные коды пикселов сдвигать вправо на минимально необходимое количество разрядов, то есть не более, чем на два разряда. При этом может теряться точность отображения содержимого файла на экран монитора, но другого решения нет.

# 1.1.2 Структура tiff файлов

При подготовке излагаемого далее материала по структуре файлов формата tiff использована информация из источника [1].

Изображения, формируемые, например, КА ДЗЗ «Ресурс-ДК1», упаковываются в файлы формата tif (tiff).

# *1.1.2.1 Основные сведения и историческая справка о tiff формате*

Для разработки программы распаковки данных из файлов со специальными форматами представления информации необходимо эти форматы знать. Поэтому приведем подробное описание структуры файлов tif (tiff) форматов [1].

Начальные сведения о tif (tiff) форматах файлов представлены в таблице 1.1.

Структура файлов формата TIFF позволяет применять его в любых операционных средах. Формат используется большинством платформ, требующих хранения изображений.

TIFF является одним из наиболее многоцелевых и разносторонних из существующих растровых форматов. Способность расширяться и поддерживать схемы сжатия числовых данных позволяет применять этот формат для различных целей.

Спецификация TIFF была выпущена Aldus Corporation в 1986 году в качестве стандартного метода хранения черно-белых изображений. Этот первый общедоступный вариант был третьей модификацией формата TIFF, хотя ему и не был присвоен номер версии. Модификация 4.0 (1987г.) поддерживала обработку несжатых цветных RGB- изображений. TIFF модификации 5.0 (1988г), позволял хранить палитровые цветные изображения и поддерживал алгоритм сжатия LZW. TIFF 6.0 (1992г.) обладал способностью поддерживать цветные изображения в стандартах CMYK и YCbCr, а также метод сжатия JPEG.

Таблица 1.1 – Основные сведения о tiff формате

|  |  |
| --- | --- |
| Имя: | TIFF |
| Также известен как: | Tag Image File Format – теговый формат файлов с изображениями |
| Тип: | Растровый |
| Цвета: | От 1 до 24 |
| Поддерживаемые методы сжатия размеров файлов | RLE, LZW, CCITT Group 3 и Group 4, JPEG, без сжатия |
| Максимальный размер изображения: | 232-1 |
| Больше одного изображения в файле: | да |
| Создатель: | Aldus |
| Платформы: | MS-DOS, Macintosh, UNIX |
| Поддерживается приложениями: | Большинством программ, работающих с изображениями и настольными издательскими системами |
| Применение: | Предназначен для хранения и обмена данными. |

Уже два десятилетия TIFF – это стандартный файловый формат, поддерживаемый большинством программ рисования и работы с изображениями, а также настольными издательскими системами. Способность к расширению, позволяющая записывать растровые изображения любой пиксельной глубины, делает этот форма практически идеальным для хранения и обработки изображений.

TIFF имеет репутацию мощного и гибкого формата, с одной стороны, и сложного и загадочного – с другой. Открытость TIFF позволяет программисту при проектировании реализовать практически все задуманное. Однако открытость TIFF и его большие возможности превращают этот формат в один из наиболее запутанных с точки зрения понимания и применения.

Иногда случается так, что программа, способная работать с файлами TIFF, не читает конкретный файл, хотя тип содержащихся в нем данных поддерживается этой программой. Существует несколько причин, по которым «отличный» файл TIFF не воспринимается программой. Как правило, они основаны на том, что разработчик программы недостаточно разобрался в формате TIFF.

Главным источником проблем, с которыми приходится сталкиваться при работе с программами чтения TIFF, является порядок записи байтов. Байты в 16- и 32-битовых словах данных записываются в различных архитектурах процессоров по-разному. Например, архитектура процессоров фирмы Intel использует порядок расположения много байтовых кодов в оперативной памяти по принципу «младший в младшем». Архитекрута процессоров Motorola MC68000A использует для этого принцип – «старший в младшем». Чтение данных, хранящихся в порядке байтов «старший в младшем», в формате «младший в младшем» (либо наоборот) приведет к образованию «мусора».

Другим источником проблем может быть алгоритм кодирования, примененный для сжатия данных изображения. Конкретная программа может поддерживать лишь некоторые методы сжатия данных из множества принципиально возможных в формате tif. Например, программа может не поддерживать распаковку данных, закодированных методом LZW.

Среди других проблем, часто называемых фатальными, следует отметить некорректную интерпретацию программой данных тегов, отсутствие поддержки изображений с цветовыми таблицами или неспособность читать строки развертки растра, содержащие нечетное количество байтов.

# *1.1.2.2 Организация файла*

Файлы формата TIFF состоят из трех разделов:

- заголовка файла изображений (Image File Header – IFH);

- директории файла изображений (Image File Directory – IFD);

- растровых данных.

Из перечисленных разделов необходимыми являются только IFH и IFD. Следовательно, допускается возможность существования файла TIFF, не содержащего растровых данных (правда, такой файл будет бесполезным). Файл TIFF, содержащий несколько изображений, будет включать столько же директорий файла и разделов растровых данных (по одному для каждого изображения).

TIFF имеет репутацию сложного формата, поскольку местоположение каждой директории и ее данных (включая растровые данные) может изменяться. Единственной составной частью файла TIFF, которая имеет постоянное место, является заголовок – он всегда занимает первые 8 байт каждого файла TIFF. Все остальные данные файла создаются с использованием информации IFD. Директория файла изображения и связанный с нею растр составляют *субфайл* TIFF. Не существует ограничений на количество субфайлов в файле TIFF.

Каждая директория файла содержит одну или несколько структур данных, называемых *тегами*. Каждый тег - это 12-байтовая запись, содержащая определенную информацию о растровых данных. Тег может хранить данные любого типа, и спецификация TIFF определяет свыше 70 тегов, которые применяются для представления заданной информации. Теги каждой директории объединяются в непрерывные группы.

Теги, определенные спецификацией TIFF, называются *общедоступными* и не могут изменяться в большей мере, нежели «предписано» последней спецификацией. Теги, определенные пользователем, называются *частными* и предназначены для применения в пользовательских программах через Developer′s Desk фирмы Aldus. Дополнительная информация о частных тегах содержится в спецификации TIFF 6.0.

Следует обратить внимание на то, что в спецификации TIFF 6.0. термин *тег* заменен на термин *поле*. Теперь на всю 12-байтовую запись данных указывает термин *поле*, а термин *тег* переопределен для указания только на число, идентифицирующее это поле. Здесь во избежание путаницы будет применяться термин *тег*.

На рисунке 1.3 представлены три возможных варианта внутренней структуры файла TIFF, содержащего три изображения.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Заголовок   файла TIFF* |  | *Заголовок   файла TIFF* |  | *Заголовок   файла TIFF* |
| IFD 0 |  | IFD 0 |  | Изображение 0 |
| IFD 1 |  | Изображение 0 |  | Изображение 1 |
| IFD 2 |  | IFD 1 |  | Изображение 2 |
| Изображение 0 |  | Изображение 1 |  | IFD 0 |
| Изображение 1 |  | IFD 2 |  | IFD 1 |
| Изображение 2 |  | Изображение 2 |  | IFD 2 |
| *а*) |  | *б*) |  | *в*) |
| IFD (Image File Directory) – Директория файла изображения | | | | |

Рисунок 1.3 - Три возможных варианта структуры данных TIFF файла

Во всех случаях первым в файле располагается его заголовок.

В первом случае сразу после заголовка записаны все директории файла, затем – все растровые данные. Такая структура наиболее эффективна для быстрого чтения данных директории. Во втором примере после каждой директории файла записаны ее растровые данные. Такая структура наиболее эффективна для файлов TIFF, содержащих несколько изображений. В последнем, третьем, варианте растровые данные хранятся сразу после заголовка, а затем располагаются все директории файла. Такая, на первый взгляд бесполезная, структура может возникнуть, если растровые данные записываются до того, как формируется информация директории.

Каждая IFD (директория файла изображения) является своего рода «дорожной картой», которая связывает все растровые данные файла TIFF. Данные директории можно прочесть либо непосредственно из структуры IFD, либо найдя их по смещению, записанному в IFD. Поскольку внутренние компоненты TIFF-файла связываются смещениями, (так как не имеют фиксированных позиций в файле), программы. читающие и записывающие такие файлы, зачастую очень сложны.

Возможные способы объединения структур данных в файле TIFF отображены на рисунке 1.4.

|  |
| --- |
| ***IFH (Image File Header)*** *Заголовок файла  изображения:* |
| Порядок байтов |
| Номер версии |
| Смещение первой IFD |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IFD**  Директория  файла изображения 0 |  | **IFD**  Директория  файла изображения 1 |  | **IFD**  Директория  файла изображения 2 |
| Количество тегов |  | Количество тегов |  | Количество тегов |
| Тег 0 |  | Тег 0 |  | Тег 0 |
| Тег 1 |  | Тег 1 |  | Тег 1 |
| Тег n1 |  | Тег n2 |  | Тег n3 |
| Смещение  следующей IFD |  | Смещение  следующей IFD |  | Смещение  следующей IFD |
|  |  |  |  |  |
| Данные  изображения 0 |  | Данные  изображения 1 |  | Данные  изображения 2 |

Рисунок 1.4 - Логическая структура файла TIFF,   
содержащего несколько изображений

С помощью смещения в файле TIFF можно задать одно из трех перечисленных ниже местоположений:

- смещение, указанное в последних четырех байтах заголовка, определяет позицию первой директории;

- последние четыре байта каждой директории содержат информацию о смещении следующей директории;

- последние четыре байта каждого тега – либо информацию о смещении данных, либо сами данные.

Величина смещения всегда определяется количеством байтов от начала файла TIFF.

# *1.1.2.3 Подробное описание структуры TIFF файла*

### *Заголовок файла*

Несмотря на сложность формата TIFF, записанные в нем файлы имеют очень простой заголовков (IFH): он содержит три поля информации и имеет длину 8 байтов (таблица 1.2). Поля заголовка следуют в файле именно в указанной последовательности.

Таблица 1.2 – Заголовок tiff файла

|  |  |  |
| --- | --- | --- |
| Имя поля заголовка | Тип данных | Комментарии |
| Identifier | word (2 байта) | Идентификатор порядка байтов |
| Version | word (2 байта) | Номер версии TIFF |
| IFDOffset | dword (4 байта) | Смещение первой директории файла |

На языке Pascal заголовок файла описывается следующим типом данных:

type

TTiffHeader = record

Identifier: word;

Version : word;

IFDOffset : dword;

end; // TTiffHeader

Поле Identifier имеет значение 4949h(II) или 4D4Dh (MM) и определяет порядок байтов, в котором записаны данные, - «младший в младшем» (Intel-формат) или «старший в младшем» (Motorola-формат) соответственно. Все данные, расположенные после этого поля, хранятся в порядке байтов, заданном идентификатором. *Указанные значения идентификатора были выбраны из тех соображений, что они не зависят от порядка байтов в файле: имеют одно и то же значение как при упаковке байт по принципу «младший в младшем», так и при использовании принципа «старший в мадшем»*.

Поле Version содержит номер версии формата TIFF. Номер версии (независимо от модификации) всегда равен 42, поэтому значение поля следует рассматривать как число идентифицирующее файл формата tiff.

Узнать, имеет ли данный файл формат TIFF, достаточно легко. Для этого необходимо прочесть лишь первые четыре байта файла. Если получаются 16-ричные коды 49h 49h 2Ah 00h, либо 4Dh 4Dh 00h 2Ah, то почти наверняка можно быть уверенным, что это TIFF-файл.

Поле IFDOffset является 32-битовым смещением позиции первой IDF. Его значение может передаваться как параметр функции поиска, что позволит найти начало информации файла. Если директория расположена непосредственно после заголовка, то поле IFDOffset буде иметь значение 08h, поскольку заголовок занимает 8 байт.

### *Директория файла*

Директория файла TIFF (IFD) представляет собой набор информации, которая применяется для описания связанных с ней растровых данных. Подобно заголовкам других форматов, директория содержит информацию о ширине, высоте и пиксельной глубине изображения, количестве цветовых плоскостей и типе сжатия данных. Однако, в отличие от обычных заголовков с постоянной структурой и размером, IFD является динамической структурой и может не только иметь различные размеры, но и находиться в любом месте файла TIFF. Кроме того, в одном файле TIFF могут существовать несколько директорий. Формат директории приведен на рисунке 1.4.

Файл TIFF может содержать любое количество изображений (включая нулевое). Каждое изображение рассматривается как отдельный растровый *субфайл*, данные которого описываются информацией IFD. Каждый субфайл TIFF может быть записан в виде отдельного файла или вместе с другими субфайлами объединен в один файл TIFF. Каждый растровый субфайл имеет свою IFD и может располагаться в любом месте файла (после заголовка). Каждому изображению может соответствовать только одна директория, и наоборот.

*Относительно изображений ДЗЗ, которые будут рассматриваться в данной дисциплине, можно считать, что каждый файл содержит ровно одно изображение.*

Подробно последовательность и содержание полей информации в IDF приведено в таблице 1.3.

Таблица 1.3 – Содержание полей в IDF

|  |  |  |
| --- | --- | --- |
| Имя поля директории | Тип данных | Комментарии |
| NumDirEntries | word (2 байта) | Количество тегов в данной IDF |
| Тег № 0 | 12 байт | Массив из NumDirEntries тегов |
| Тег № 1 | 12 байт |
| … | … |
| Тег № ( NumDirEntries-1) | 12 байт |
| NextIFDOffset | dword (4 байта) | Смещение следующей IFD  (относительно начала файла) |

Как видно из таблицы 1.3, размер директории файла может изменяться, поскольку она, как правило, содержит переменное количество записей данных, *тегов*. Каждый тег хранит уникальную информацию. Количество тегов в IFD может изменяться от файла к файлу.

Директорию файла на языке Pascal удобно описать следующим типом данных:

TTiffIFD = record

NumDirEntries: word;

TagListPtr : array of TTiffTagPtr;

NextIFDOffset: dword;

end; // TTiffIFD

Здесь массив тегов описан как массив указателей на теги. Описание отдельного тега будет приведено ниже.

Поле NumDirEntries является 2-байтовой величиной, задающей количество тегов в IFD. После этого поля следует серия тегов, причем их количество соответствует значению поля NumDirEntries. Каждая теговая структура имеет размер 12 байтов и может быть представлена в виде элемента массива структур типа TIFTAG (дополнительная информация о тегах формата TIFF приведена ниже). Количество тегов в одной IFD не должно превышать 65535, поскольку именно это число является максимально возможным для 2-байтовой без знаковой переменной.

Поле NextIFDOffset содержит информацию о смещении начала следующей директории относительно первого байта файла. Если последующих IFD нет, то значение этого поля равно 00h.

### *Теги TIFF файла*

Каждый тег представляет собой структуру данных постоянного размера: 12 байт. При этом содержимое тега либо задает собственно значение некоторого конкретного (ровно одного) свойства изображения (например, количество строк в изображении), либо указывает на расположение данных, которые в этом случае могут занимать любое количество байтов и располагаться в любом месте директории. Во втором случае тег содержит указатель на расположение данных в виде 4-х байтового кода, задающего смещение первого байта указываемой порции данных относительно начала файла.

Тег в файле формата TIFF имеет структуру, описанную в таблице 1.4. В файле данные записаны именно в том порядке, который указан в этой таблице.

Таблица 1.4 – Структура тегов

|  |  |  |
| --- | --- | --- |
| Имя поля тега | Тип данных | Комментарии |
| TagId | word (2 байта) | Идентификатор тега |
| DataType; | word (2 байта) | Скалярный тип элементов данных |
| DataCount | dword (4 байта) | Количество элементов данных тега |
| DataOffset | dword (4 байта) | Смещение элементов данных |

Структуру тега удобно описать следующим типом данных:

TTiffTag = record

TagID : word;

DataType : word;

DataCount: dword;

DataOffset: dword;

end; // TTiffTag

TTiffTagPtr = ^TTiffTag;

В описании IFD, приведенном выше в таблице 1.3, массив тегов – это массив переменных типа TTiffTagPtr, каждая из которых является указателем на описание соответствующего тега.

Поле TagId задает числовое значение, идентифицирующее логическое содержание информации, содержащейся непосредственно в данном теге, либо информации, на положение которой указывает этот тег. Обычно в каждом файле TIFF имеются теги, задающие высоту и ширину изображения, пиксельную глубину и схему кодирования данных, примененную для сжатия данных растра. Теги идентифицируются по содержимому поля TagId и всегда записываются в директории в порядке возрастания значений этого поля.

Поле DataType может иметь значения, перечисленные в таблице 1.5 и указывающие на тип скалярных данных, содержащихся в теге.

Подчеркнем, что поле DataType содержит не имена типов скалярных данных, а условные номера типов, заданные в таблице 1.5.

Таблица 1.5 – Типы скалярных данных

|  |  |  |
| --- | --- | --- |
| 1 | BYTE | 8-битовое целое без знака |
| 2 | ASCII | Строка, оканчивающаяся NULL |
| 3 | SHORT | 16-битовое целое без знака |
| 4 | LONG | 32-битовое целое без знака |
| 5 | RATIONAL | Два 32-битовых целых без знака |

Типы данных BYTE, SHORT и LONG соответствуют типам данных BYTE, WORD и DWORD. Тип данных ASCII определяет строки 7-битовых символьных ASCII-кодов, причем эти строки всегда оканчиваются NULL и могут быть заполнены (по необходимости) до четного значения длины. Тип данных RATIONAL фактически описывает два значения типа LONG и применяется для хранения двух компонентов дроби. Первое значение является ее числителем, второе – знаменателем.

Начиная с модификации TIFF 6.0, номенклатура типов данных дополнена типами, приведенными в таблице 1.6

Таблица 1.6 – Дополнительные типы скалярных данных

|  |  |  |
| --- | --- | --- |
| 6 | SBYTE | 8-битовое целое со знаком |
| 7 | UNDEFINE | 8-битовый байт |
| 8 | SSHORT | 16-битовое целое со знаком |
| 9 | SLONG | 32-битовое целое со знаком |
| 10 | SRATIONAL | Два 32-битовых целых со знаком |
| 11 | FLOAT | 4-байтовое IEEE–значение с плавающей запятой одинарной точности |
| 12 | DOUBLE | 8-байтовое IEEE–значение с плавающей запятой двойной точности |

Типы данных SBYTE, SSHORT и SLONG применяются для хранения чисел со знаком. В FLOAT и DOUBLE хранятся числа одинарной и двойной точности, записанные в формате IEEE (таблица 1.7).

Тип UNDEFINE представляет собой 8-битовый байт, который может содержать нетипичные данные, и обычно применяется в частных тегах. Пример использования этого типа данных – хранение структур данных (целиком) в частном теге, заданном значением 7 в поле DataType, а поле DataCount в этом случае будет содержать информацию о количестве байтов в этой структуре.

Таблица 1.7 – Параметры форматов числовых данных   
в стандарте IEEE 754

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Формат | | |
| Одинарный | Двойной | Расширенный |
| Ширина слова, бит | 32 | 64 | 80 |
| Ширина порядка, бит | 8 | 11 | 15 |
| Длина мантиссы, бит | 23 | 52 | 64 |
| Смещение порядка | 127 | 1023 | 16383 |
| Максимальный порядок | 127 | 1023 | 16383 |
| Минимальный порядок | -126 | -1022 | -16382 |

В тегах формата TIFF6.0. могут применяться только указанные типы данных. Исключение составляют теги SMinSampleValue и SMaxSampleValue, которые используют любые типы данных. В поле DataCount записаны сведения о количестве элементов, определяемых тегом (но не фактический размер самих данных). Следовательно, значение 08h поля DataCount не обязательно указывает на то, что тег содержит 8 байтов данных; это значение может свидетельствовать и о том, что тег содержит восемь элементов данных заданного типа. Например, значение 8 поля DataCount и значение 3 поля DataType говорят о том, что данными тега являются восемь последовательно расположенных 16-битовых целых без знака общим объемом 16 байт. Значение 28h = 40 поля DataCount и значение 2 поля DataType определяют строку символов ASCII длиной 40 байтов с учетом ограничителя NULL (но не определяют наличия символов-заполнителей, даже если они есть). Значение 1 поля DataCount и значение 5 поля DataType задают одно значение типа RATIONAL общей длиной 8 байтов.

Поле DataOffset является 4-байтовым и содержит информацию о смещении данных тега в файле TIFF. Если данные тега имеют размер 4 байта и менее. То они могут быть записаны в указанном поле. Если же размер данных тега составляет более 4 байтов, то поле содержит сведения о смещении позиций этих данных. Упаковка данных в поле DataOffset рассматривается спецификацией TIFF как процесс оптимизации и не является обязательной для применения. Обычно большая часть данных сохраняется за пределами тега (до или после директории файла).

В таблице 1.8 перечислены некоторые общедоступные теги, включенные в спецификации TIFF 4.0, 5.0 и 6.0.

В представленной ниже таблице символ «\*» указывает на то, что данный тег определен, символом «-» отмечены теги, которые не имеют определения, а символом «х» - устаревшие теги.

Таблица 1.8 - Типы тегов TIFF файла

| Имя тега | TagID | Тип данных | 4.0 | 5.0 | 6.0 |
| --- | --- | --- | --- | --- | --- |
| BitsPerSample | 258 | SHORT | \* | \* | \* |
| CellLength | 265 | SHORT | \* | \* | \* |
| CellWidth | 264 | SHORT | \* | \* | \* |
| ColorMap | 320 | SHORT | - | \* | \* |
| Compression | 259 | SHORT | \* | \* | \* |
| Uncompressed | 1 |  | \* | \* | \* |
| CCITT ID | 2 |  | \* | \* | \* |
| CCITT Group 3 | 3 |  | \* | \* | \* |
| CCITT Group 4 | 4 |  | \* | \* | \* |
| LZW | 5 |  | - | \* | \* |
| JPEG | 6 |  | - | - | \* |
| Uncompressed | 32771 |  | \* | х | х |
| Copyright | 33432 | ASCII | - | - | \* |
| DateTime | 306 | ASCII | - | \* | \* |
| DocumentName | 269 | ASCII | \* | \* | \* |
| ImageDescription | 270 | ASCII | \* | \* | \* |
| ImageHeight | 257 | SHORT or LONG (32 бита) | \* | \* | \* |
| ImageWidth | 256 | SHORT or LONG | \* | \* | \* |
| JPEGACTTables | 521 | LONG | - | - | \* |
| JPEGDCTTables | 520 | LONG | - | - | \* |
| JPEGInterchangeFormat | 513 | LONG | - | - | \* |
| JPEGInterchangeFormatLength | 514 | LONG | - | - | \* |
| MaxSampleValue | 281 | SHORT | \* | \* | \* |
| MinSampleValue | 280 | SHORT | \* | \* | \* | |
| Orientation | 274 | SHORT | \* | \* | \* | |
| PhotometricInterpretation | 262 | SHORT | \* | \* | \* | |
| WhiteIsZero | 0 |  | \* | \* | \* | |
| BlacrIsZero | 1 |  | \* | \* | \* | |
| RGB | 2 |  | \* | \* | \* | |
| RGB Palette | 3 |  | - | \* | \* | |
| Tranparency Mask | 4 |  | - | - | \* | |
| CMYK | 5 |  | - | - | \* | |
| YCbCr | 6 |  | - | - | \* | |
| PlanarConfiguration | 284 | SHORT | \* | \* | \* | |
| ResolutionUnit | 296 | SHORT | \* | \* | \* | |
| RowsPerStrip | 278 | SHORT or LONG | \* | \* | \* | |
| SamplesPerPixel | 277 | SHORT | \* | \* | \* | |
| SMaxSampleValue | 341 | Any | - | - | \* | |
| SMinSampleValue | 340 | Any | - | - | \* | |
| Software | 305 | ASCII | - | \* | \* | |
| SripByteCounts | 279 | LONG or SHORT | \* | \* | \* | |
| StripOffsets | 273 | SHORT or LONG | \* | \* | \* | |
| XPosition | 286 | RATIONAL | \* | \* | \* | |
| XResolution | 282 | RATIONAL | \* | \* | \* | |
| YCbCrCoefficients | 529 | RATIONAL | - | - | \* | |
| YCbCrPositioning | 531 | SHORT | - | - | \* | |
| YCbCrSubSampling | 530 | SHORT | - | - | \* | |
| YPosition | 287 | RATIONAL | \* | \* | \* | |
| YResolution | 283 | RATIONAL | \* | \* | \* | |
| WhitePoint | 318 | RATIONAL | - | \* | \* | |

В таблице серым фоном выделены наиболее необходимые теги: теги которые могут быть востребованы в рамках лабораторного практикума.

### *Организация данных тегов*

Чтобы избавить разработчиков от необходимости угадывать, какие теги должны быть записаны в файл TIFF и какие важно прочесть, спецификация TIFF предлагает концепцию основных типов изображений: двухуровневое (бинарное), полутоновое, цветное палитровое и полностью цветное. Для каждого основного типа изображений установлен обязательный минимальный объем тегов.

В спецификации TIFF 5.0 перечисленные основные типы изображений были названы классами. Каждый файл TIFF строится из общего класса (Class X) и может модифицироваться дополнительным классом, зависящим от типа хранимого изображения: Class B (двухуровневое), Class G (полутоновое), Class P (палитровое) и Class R (полностью цветное RGB).

В спецификации TIFF 6.0 эти классы переопределены в четыре отдельные конфигурации базовых файлов. Класс Х объединен с каждым из остальных четырех классов для формирования конфигураций двухуровневых, полутоновых, цветных палитровых и полностью цветных файлов. И хотя спецификация TIFF 6.0 практически «покончила» с концепцией классов, весьма вероятно, что еще продолжительное время TIFF-файлы будут определяться согласно этим обозначениям классов.

Фактически существует еще один класс TIFF – Class F, предназначенный для хранения факсимильных изображений в данном формате. Этот класс файлов (также известен под именем Everex Fax File Format) был создан фирмой Cygnet Technologies. И несмотря на то, что данная фирма перестала существовать, формат TIFF Class F продолжает применяться, в частности в продуктах Everex, поскольку отлично поддерживает хранение факсимильных данных.

В таблице 1.9 перечислены самые необходимые теги, которые будут встречаться в большинстве файлов изображений ДЗЗ.

Таблица 1.9 - Минимально необходимый набор тегов

|  |  |  |
| --- | --- | --- |
| Имя класса | Тип тега  (TagID) | Имя тега |
| Bi-level and  Gray-scale | 256 | ImageWidth |
| 257 | ImageLength |
| 258 | BitsPerSample |
| 259 | Compression |
| 262 | PhotometricInterpretation |
| 273 | StripOffsets |
| 277 | SamplesPerPixel |
| 278 | RowsPerStrip |
| 279 | SripByteCounts |

Все остальные указанные в спецификации TIFF теги используются по усмотрению разработчика. Если программа чтения TIFF должна поддерживать лексический анализ и интерпретацию всех необходимых ей тегов, то *программа записи не обязательно должна включать в каждый файл все возможные теги*.

### *Полосы изображения*

Файлы TIFF хранят только растровые данные.

Растровые данные в TIFF файле расположены не всегда сразу после заголовка: они могут находиться почти в любом месте TIFF файла. Необходимо однозначно понимать, как данные хранятся в TIFF файле.

Изображения, записанные в формате TIFF 6.0, могут быть организованы в виде полос, и в виде фрагментов.

Полоса представляет собой отдельный набор из одной и более последовательно расположенных строк растровых данных изображений. Разделение данных изображений на полосы значительно упрощает буферирование, произвольный доступ и чередование.

Для определения полосы растровых данных в файле TIFF необходимо использовать три тега: RowsPerStrip, StripOffsets и SripByteCounts.

Тег RowsPerStrip задает количество строк растровых данных в каждой полосе. Для всех полос в субфайлах TIFF применяется однотипная схема сжатия, все они имеют одинаковый битовый и цветной пол, пиксельную глубину и т.д.

Без StripOffsets – второго обязательного тега - программа чтения TIFF никогда не найдет в файле данные изображения. Этот тег содержит массив смещений (по одному на полосу), которые указывают позицию первого байта каждой полосы в файле TIFF. Первый элемент массива указывает смещение первой полосы, второй – смещение второй полосы и т.д.

Данные о кодах пикселей могут быть разделены на плоскости: тег PlanarConfiguration = 2. Но в цикле лабораторных работ такие файлы не будут использоваться, поэтому соответствующая информация здесь детализироваться не будем.

Значения тега StripOffsets всегда рассматриваются как показатели смещения от начала файла.

Тег StripOffsets позволяет размещать каждую полосу файла TIFF целиком и независимо от других полос того же субфайла. Это значит, что они могут размещаться в файле в любом месте и в любом порядке. Большинство «шустрых» программ чтения TIFF находят начало первой полосы, а потом пытаются читать данные изображения одной большой порцией, без определения позиции каждой полосы с помощью массива StripOffsets. Это возможно только в том случае, когда все полосы записаны в файле TIFF в том же порядке, в каком расположены строки оригинального растра. Если же полосы записываются в другой последовательности, например в плоскостной или чередующейся манере, то изображение появляется на экране слоями либо в виде перемешанных строк. Если же полосы записываются в файл в произвольном порядке и на свободно выбранные места, то часть прочитанных данных может не только не относится к данным изображения, но даже и не принадлежать этому файлу TIFF. В таком случае на экране появляется только «мусор».

Тег RowsPerStrip и все элементы массива тега StripOffsets обычно имеют значения типа LONG (32 бита). Спецификация TIFF5.0 позволила этим тегам использовать и 16-битовые значения типа SHORT. Очень старые программы чтения TIFF могут предполагать, что значением этих тегов всегда будет LONG, и в случае значения SHORT будут работать «неправильно». Такое новшество было введено в первую очередь потому, что существовали программы чтения TIFF, которые, прежде чем использовать значения StripOffsets, читали их в массив памяти. Спецификация TIFF6.0 предлагает ограничить размер смещений 64 Кб.

Тег SripByteCounts описывает массив значений, указывающих размер каждой полосы в байтах. Подобно тегу StripOffsets он содержит одномерный (в случае порций) или двухмерный (в случае плоскостей) массив значений – по одному на полосу. Каждое из этих значений характеризует количество байтов сжатых растровых данных, записанных в соответствующей полосе.

Целесообразность этого тега объясняется следующим. В некоторых случаях полосы изображения занимают в файле различное количество байтов, например из-за сжатия растровых данных изображения. Как говорилось выше, значение тега SripByteCounts определяет размер полосы изображения *после* сжатия данных изображения. Хотя несжатые строки изображения, как правило, занимают постоянное количество байтов, размер сжатой строки зависит от типа содержащихся в ней данных. Поскольку мы обычно записываем в полосу фиксированное количество строк (а не байтов) то очевидно, что большинство полос будут различаться по длине – ведь каждая сжатая строка имеет свой размер. Если же растровые данные не сжимались, то все полосы будут одинаковыми по размеру.

Однако и в этом случае могут быть исключения, чаще всего связанные с размером последней полосы изображения. Программы записи TIFF обычно пытаются создавать полосы таким образом, чтобы каждая из них включала в себя одинаковое количество строк. Например, растр из 2200 строк может быть разделен на 22 полосы, каждая из которых будет содержать 100 строк растровых данных. Но такое деление не всегда возможно. Если растр, который состоит из 482 строк, разделить на полосы, содержащие по пять строк, то получим 97 полос, причем 96 из них будут состоять из пяти строк данных, а 97-я – только из двух. В этом случае значение пятого тега RowsPerStrip будет корректным для всех полос, кроме последней.

Программе чтения TIFF вовсе не обязательно знать количество строк в каждой полосе. Для того, чтобы прочесть данные изображения, ей нужно только обладать информацией о количестве байтов. С другой стороны, спецификация TIFF могла бы потребовать, чтобы к каждой «короткой» полосе были добавлены дополнительные строки-заполнители. Однако в этом нет необходимости. Нужно лишь прочесть последнее значение тега SripByteCounts, чтобы определить количество байтов, которое следует прочесть для формирования последней полосы. Из спецификации TIFF не очевидно, что значения тега RowsPerStrip определяет *максимальное* количество строк в полосе, что не всегда корректно для последней полосы изображения. На практике многие файлы TIFF содержат всего одну полосу данных, задавая произвольно большое значение RowsPerStrip.

В организации растровых данных в виде полос есть свои преимущества.

Во-первых, не все программы могут прочесть в память файл целиком. Но даже если система обладает достаточно большим объемом памяти, программа чтения TIFF не всегда имеет возможность ее использовать. Обычно она пытается запросить как можно больший буфер и начинает читать растровые данные по одной полосе до тех пор, пока буфер не заполнится. После обработки данных читаются следующие полосы. Если изображение помещается в памяти целиком, то сначала читаются все полосы файла TIFF, а после этого происходит обработка данных.

Во-вторых, создав таблицу смещений полос, можно значительно упростить произвольный доступ к растровым данным. Если необходимо отобразить, например, последние 100 строк изображения, состоящего из 480 строк, а растровые данные разделены на 48 полос по 10 строк в каждой, то программа чтения TIFF может пропустить первые 380 строк и обработать только те полосы, смещения которых записаны в последних десяти элементах массива тега StripOffsets. Если смещение не указано, то для определения начальной позиции последних 100 строк необходимо читать все изображение с его начала.

Запись растра в файл TIFF в виде одной большой полосы (таким образом записаны многие файлы TIFF) не всегда целесообразна. Подобные изображения, как правило, плохо обрабатываются, поскольку программа вынуждена запрашивать слишком много памяти для их размещения целиком. В случае большого изображения или слабой системы такой запрос может быть не выполнен.

Тег SamplesPerPixel определяет структуру кода пиксела: количество кодов, соответствующих одному пикселу. Для монохромных изображений SamplesPerPixel= 1. Для изображений формата RGB SamplesPerPixel= 3. При этом относительно изображений высокого разрешения один код (один Sample) состоит из 16 бит, что определяется содержимым тега BitsPerSample.

### *Фрагменты изображения*

Полосы не являются единственным возможным способом организации растровых данных. В TIFF 6.0 представлена концепция разделения растровых данных на *фрагменты*. Полоса – это одномерный объект, имеющий только длину, а фрагмент можно рассматривать как двух мерную полосу, имеющую длину и ширину (подобно растру). Фактически каждый фрагмент изображения можно рассматривать как маленький растр, являющийся частью большого растра. Чтобы получить изображение, необходимо собрать все фрагменты, учитывая их расположение в оригинальном изображении.

В рамках цикла лабораторных работ концепция фрагментов не будет востребована, поэтому рассматривать ее детальней не будем.

### *Сжатие изображений*

TIFF поддерживает больше типов сжатия данных, чем любой другой файловый формат изображений. Кроме того, он позволяет применять схему сжатия, заключающуюся в добавлении необходимых тегов, определяемых пользователем. Формат TIFF 4.0 поддерживал лишь групповое кодирование (RLE) и CCITT (Т.4 и Т.6). В TIFF 5.0 добавлена схема сжатия LZW, обычно используемая при работе с цветными изображениями, а в TIFF 6.0 – метод JPEG , применяемый для сжатия полутоновых цветных и полутоновых монохромных изображений.

В TIFF применяется также схема сжатия PackBits RLE, используемая инструментальными средствами Macintosh.

В рамках цикла лабораторных работ будут использоваться только изображения, упакованные в файл без сжатия.

# *1.1.2.4 Пример структуры данных TIFF файла*

Далее приводится пример конкретной структуры данных файла TIFF, содержащего некоторое изображение.

В таблице 1.10 показано содержимое первых 8 байт файла, которые во всех файлах TIFF задают заголовок файла.

Таблица 1.10 – Содержание заголовка файла

|  |  |  |
| --- | --- | --- |
| Поле заголовка | Значение поля | Комментарий |
| Identifier | 18761 = 4949h | ⇒ данные в файле упакованы младшими байтами вперед |
| Version | 42 = 2Ah | ⇒ это TIFF файл |
| IFDOffset | 14 839 910 [байт] | - смещение начала описания IFD  относительно первого байта файла |

Значение поля IFDOffset = 14 839 910 заголовка файла позволяет найти в файле информацию о первой директории файла (IFD). Такое большое значение содержимого поля показывает, что в данном случае структура tiff файла имеет вид, показанный на рисунке 1.3*в*.

Первое поле NumDirEntries директории (таблица 1.3) задает количество тегов, описывающих изображение и его основные параметры (NumDirEntries = 10).

Непосредственно за полем NumDirEntries, занимающим 2 байта, следует массив описаний 10-ти тегов, поскольку NumDirEntries = 10. Описание одного тега занимает всегда 12 байт. Данные о 10 тегах, считанных из файла, приведены в таблице 1.11.

Таблица 1.11 – Содержание тегов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № тега | TagID | DataType | DataCount | DataOffset | Комментарии |
| 0 | 256 | 3 | 1 | 3311 | Ширина изображения  (ImageWidth= 3311 пикселей) |
| 1 | 257 | 3 | 1 | 2241 | Высота изображения  (ImageHeight= 2241 строка) |
| 2 | 258 | 3 | 1 | 16 | Бит на код цвета  (BitPerSample= 16) |
| 3 | 259 | 3 | 1 | 1 | Изображение в файле не сжато  (UnCompressed) |
| 4 | 262 | 3 | 1 | 1 | Признак «BlackIsZero»  (FotometricInterpretation =1) |
| 5 | 273 | 4 | 1 | 8 | Смещение 1-го байта, набора данных, задающих содержимое растра полосы изображения с номером 0. (StripOffset= 8) |
| 6 | 277 | 3 | 1 | 1 | Кодов цвета на пиксель  (SamplesPerPixel) |
| 7 | 278 | 3 | 1 | 2241 | Строк изображения в полосе  (RowsPerStrip= 2241) |
| 8 | 279 | 4 | 1 | 14 839 902 | Байт данных, описывающих полосу изображения: StripByteCounts |
| 9 | 284 | 3 | 1 | 1 | Признак кодирования растровых данных по кодам пикселей  (PlanarConfiguration=1) |

Логическая структура рассматриваемого файла дана в таблице 1.12.

Таблица 1.12 – Логическая структура файла в адресном пространстве

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Смещение  от 1-го байта | Значение | Имя поля данных | | Комментарий 1 | Коммен­тарий 2 |
| 0 | 4949h | Identifier | | Заголовок файла  (IFH) |  |
| 2 | 42 (2Ah) | Version | |
| 4 | ***14 839 910*** | IFDOffset | |
| 6 |
| 8 | XXXXh | Код пикселя с координатами (0, 0) | | | 0-я  строка  растра |
| 10 | XXXXh | Код пикселя с координатами (1, 0) | | |
| … | … | … | | |
| 6628 | XXXXh | Код пикселя с координатами (3310, 0) | | |
| … |  | … | | |  |
| 14 833 288 | XXXXh | Код пикселя с координатами (0, 2240) | | | 2240-я  строка  растра |
| … | … | … | | |
| 14 839 908 | XXXXh | Код пикселя с координатами (3310, 2240) | | |
| **14 839 910** | 10 | NumDirEntries | |  | 1-я дирек­тория файла  (IFD) |
| 14 839 912 | 256 | TagID | Тег № 0 | Ширина  изображения:  ImageWidth = 3311 |
| 14 839 914 | 3 | DataType |
| 14 839 916 | 1 | DataCount |
| 14 839 918 |
| 14 839 920 | ***3311*** | DataOffset |
| 14 839 922 |
| 14 839 924 | 257 | TagID | Тег № 1 | Высота  изображения:  ImageHeight = 2241 |
| 14 839 926 | 3 | DataType |
| 14 839 928 | 1 | DataCount |
| 14 839 930 |
| 14 839 932 | ***2241*** | DataOffset |
| 14 839 934 |
| .. | .. | .. |  |  |
| 14 839 936 | 273 | TagID | Тег № 5 | Смещение данных о 0-й полосе растра:  StripOffset = 8 |
| 14 839 938 | 4 | DataType |
| 14 839 940 | 1 | DataCount |
| 14 839 942 |
| 14 839 944 | ***8*** | DataOffset |
| 14 839 946 |

Окончание таблицы 1.12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Смещение  от 1-го байта | Значение | Имя поля данных | | Комментарий 1 | Коммен­тарий 2 |
| … | … | … | | … |  |
| 14 840 032 | 284 | TagID | Тег № 9 | Признак кодирования растровых данных по кодам пикселей:  PlanarConfiguration=1 |
| 14 840 034 | 3 | DataType |
| 14 840 036 | 1 | DataCount |
| 14 840 038 |
| 14 840 040 | ***1*** | DataOffset |
| 14 840 042 |
| 14 840 044 | ***0*** | NextIFDOffset | | Следующей  директории нет |
| 14 840 046 |

В таблице 1.12 непосредственно после массива описаний тегов следует 4-байтовое поле, задающее положение в файле следующей директории. В данном случае содержимое этого поля равно 0 (NextIFDOffset=0) , что указывает на отсутствие в файле второго изображения.

Когда количество элементов данных тега равно 1, то значение поля DataOffset задает непосредственно значение параметра, за который «отвечает» этот тег. Поэтому в таблице 1.11 в полях DataOffset отражены значения соответствующих параметров изображения.

# *1.1.2.5 Влияние сдвига кодов пикселей на отображение содержимого файлов*

Здесь демонстрируются результаты визуализации содержимого TIFF файлов при сдвигах кодов пикселей вправо на разное количество разрядов.

На рисунках 1.5 – 1.7 показаны результаты визуализации фрагмента изображения, упакованного в tiff файле в формате 10 значащих бит кода пиксела, при сдвигах эти кодов в процессе визуализации содержимого файла вправо на 0, 1 и 2 разряда соответственно.

На рисунках 1.8 – 1.15 показан фрагмент изображения, упакованного в tiff файле в формате 16 бит/пиксел, в котором значащими являются старшие разряды 16 битного кода. При визуализации фрагмента были использованы 8 вариантов сдвига кодов пикселов.

|  |  |
| --- | --- |
|  |  |
| Рисунок 1.5- Отображение без  сдвига кодов пикселов | Рисунок 1.6- Отображение со сдвигом кодов пикселов вправо на 1 разряд |



Рисунок 1.7- Отображение со сдвигом   
кодов пикселов вправо на 2 разряда

|  |  |
| --- | --- |
|  |  |
| Рисунок 1.8- Отображение со сдвигом кодов пикселов вправо на 1 разряд | Рисунок 1.9- Отображение со сдвигом кодов пикселов вправо на 2 разряда |

|  |  |
| --- | --- |
|  |  |
| Рисунок 1.10- Отображение со сдвигом кодов пикселов вправо на 3 разряда | Рисунок 1.11- Отображение со сдвигом кодов пикселов вправо на 4 разряда |

При сдвиге кодов пикселей вправо на 3 разряда на изображении появились контуры береговой линии, очертания которой при сдвиге на 4 разряда стали четче.

|  |  |
| --- | --- |
|  |  |
| Рисунок 1.12- Отображение со сдвигом кодов пикселов вправо на 5 разрядов | Рисунок 1.13- Отображение со сдвигом кодов пикселов вправо на 6 разрядов |

|  |  |
| --- | --- |
|  |  |
| Рисунок 1.14- Отображение со сдвигом кодов пикселов вправо на 7 разрядов | Рисунок 1.15- Отображение со сдвигом кодов пикселов вправо на 8 разрядов |

Приведенные рисунки 1.5 – 1.7 показывают, что в некоторых файлах изображения ДЗЗ упакованы так, что значащими разрядами 16-ти битного коды пиксела являются младшие 10. Такую упаковку называют, упаковкой с выравниванием по младшему разряду 16-битной разрядной сетки (РС). В других файлах коды пикселей упакованы в 16-битную РС с выравниванием по старшему разряду РС (рисунки 1.8 – 1.15).

В рамках лабораторного практикума будем использовать только изображения с упаковкой кодов пикселов с выравниванием по младшему разряду 16-битной РС.

# *1.1.2.6 Особенность структуры много матричных оптико-электронных преобразователей КА ДЗЗ*

Изобразительные системы КА ДЗЗ имеют в своей структуре сканирующие многоматричные оптико-электронные преобразователи (МОЭП), обеспечивающие получение широкоформатных снимков Земной поверхности. Примером МОЭП служит аппаратура типа «Сангур» разработки НПП «ОПТЭКС» [2]. В ней фоточувствительные матрицы, составляющие МОЭП, построены на основе матриц приборов с зарядовой связью (ПЗС матрицы) и работают в режиме задержки накопления зарядов (ВЗН). Как источник информации, каждая матрица эквивалентна ПЗС линейке, с которой одномоментно снимается только одна строка видеоинформации. Кадровая развертка получается механическим сканированием подстилающей поверхности (Земли). При этом каждая ПЗС матрица порождает полосу изображения шириной равной количеству ПЗС ячеек в строке матрицы. За счет такой организации процесса съемки имеется возможность получения кадров, длина которых (количество строк изображения) ограничивается только объемом буферной памяти съемочной аппаратуры и технологическими режимными ограничениями.

МОЭП аппаратуры типа «Сангур» может содержать несколько десятков матриц. При этом очередные 6 матриц логически объединены в блок, называемый зоной компенсации (ЗК). В отдельный файл упаковываются полосы изображений, сформированные ПЗС матрицами, составляющими одну ЗК. В результате в каждом файле представлены 6 полос изображений. При упаковки в файл эти полосы склеиваются друг с другом по боковым границам.

Взаимное расположение ПЗС матриц одной ЗК на картинной плоскости изобразительной системы представлено на рисунке 1.16.

Матрица 1

Матрица 2

Матрица 3

Матрица 4

Матрица 5

Матрица 6

Рисунок 1.16 – Ступенчатое расположение матриц ФПЗС в ОЭП

Каждая ПЗС матрица КА «Ресурс ДК1» порождает полосу изображения шириной в 1012 пикселей. В результате 6 склеенных полос порождают изображение шириной 6072 пиксела. В каждый tiff файл помимо собственно изображения (кодов яркостей пикселов) упаковывается разнообразная служебная информация, для размещения которой в файле каждая строка изображения шириной 6072 пиксела дополняется слева 32 пикселами (то есть 64 байтами), которые содержат не коды пикселов, а эту служебную информацию. В результате, упакованное в файле изображение имеет ширину 6104 пиксела, из которых первые (левые) 32 пиксела изображением не являются и при визуализации изображений, упакованных в файле, эти 32 пиксела (64 байта) в каждой строке должны пропускаться.

# 1.2 Задание

Разработать средудля визуализации изображения ДЗЗ упакованного в файле типа tiff.

### *Интерфейс разрабатываемой среды*

Пример главного окна с основными интерфейсными средствами представлен на рисунке 1.17.

На рисунке 1.18 показаны фрагменты интерфейсных средств в увеличенном масштабе.

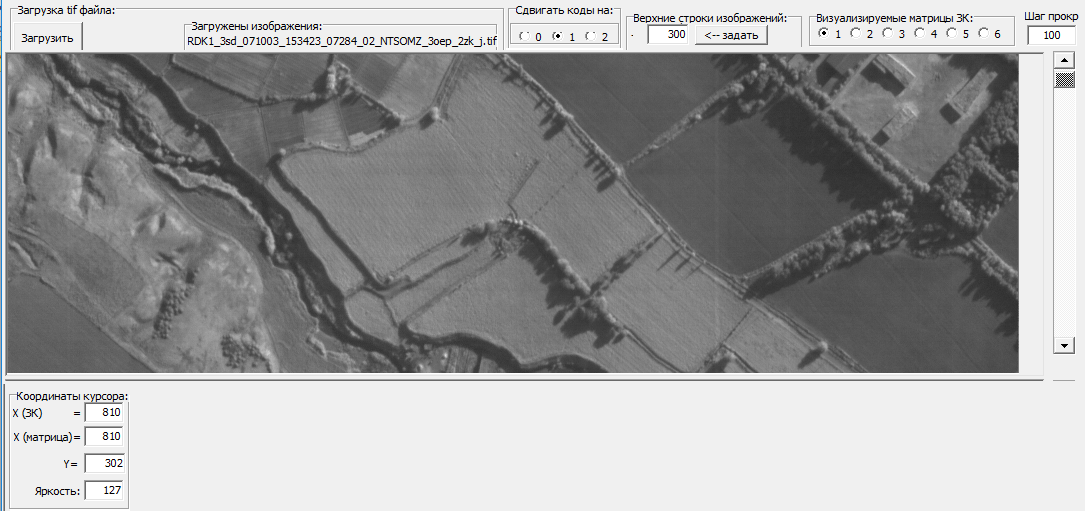


Рисунок 1.17 – Пример интерфейсных средств разрабатываемой среды

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | |  |
|  | |

Рисунок 1.18 – Фрагменты интерфейсных средств в увеличенном масштабе

Интерфейсные средства представлены:

- кнопкой «Загрузить»;

- полем для отображения имени загруженного файла;

- полем «Сдвигать коды на:» для задания величины сдвига кодов яркостей пикселов вправо при отображении на форму;

- полем «Верхние строки изображений:» предназначенным для точного выбора отображаемого фрагмента путем задания строки изображения, с которой начинается визуализируемый фрагмент;

- полем для выбора одной из 6-ти полос изображений, представленных в файле;

- полем для выбора шага прокрутки изображения по вертикали при нажатии на кнопки «вверх», «вниз» полосы прокрутки;

- полем «Координаты курсора:» для отображения текущих координат и яркости пиксела, на который указываем курсор мыши.

Среду разрабатывать на основе следующих концептуальных положений.

1. Операционная система (ОС): Windows 7 и выше.

2. Среда разработки по выбору автора.

3. Предполагаемый максимальный размер изображения: 6104 × 100000 пикселей.

4. Поддерживаемые методы сжатия изображения в файле TIFF: без сжатия.

5. Цветность изображения: изображение монохромное.

6. Отображение изображения на форме: в виде полутонового изображения в палитре серых цветов от черного до белого.

7. Разрешение изображения по глубине в файле: 16 бит/пиксел, из которых информативными являются 10 младших бит.

8. Обеспечить возможность прокрутки изображения при масштабе отображения 1:1 как шагами, так и с помощью движка полосы прокрутки.

9. Предусмотреть интерфейсные средства управления величиной сдвига кодов пикселей вправо перед их отображением на экране.

10. Отображать размер изображения.

11. Отображать текущие координаты положения курсора мыши как относительно всего изображения, так и относительно первой (верхней) строки визуализированного фрагмента.

# 1.3 Методика выполнения работы

1. Изучить теоретическую часть этих методических указаний.

2. Создать интерфейсную часть разрабатываемой среды. Для этого:

- выбрать и расположить на форме необходимые визуальные компоненты;

- все имена компонент, назначаемые автоматически, переименовать на логически содержательные имена, отражающие назначение соответствующей компоненты;

- продумать какие события, связанные с использованием интерфейсных средств, необходимо предусмотреть в программе;

- создать заготовки для обработчиков выбранных событий;

- все имена обработчиков событий, назначенные средой разработки автоматически, переименовать на логически содержательные имена, отражающие задачи, решаемые каждым обработчиком.

3. Приступить к программированию обработчиков событий, начиная с разработки алгоритма загрузки изображения из tiff файла. При этом можно руководствоваться прилагаемым алгоритмом распаковки tiff файлов (рисунок 1.19).

В программе распаковки tiff файлов необходимо выполнить рутинное программирование извлечения информации из тегов файла. В качестве помощи при программировании этого этапа распаковки файла ниже приведен пример процедуры ImagePropertesExtract (извлечение свойств изображения), написанной на языке Delphi. Синтаксис языка Delphi очень простой и понятный, поэтому при программировании на любом другом языке приведенный текст можно легко адаптировать под синтаксис этого языка.

При разработке структуры данных, связанной с хранением данных растра, необходимо учесть, что все изображение может оказаться слишком большим для размещения в оперативной памяти целиком. Поэтому следует продумать вопрос организации загрузки изображения необходимыми в данный момент фрагментами.

При загрузке изображения фрагментами следует позаботиться об оперативности (быстродействия) доступа к содержимому файла с заданным фрагментом. Необходимо учесть, что фрагмент может начинаться с любой строки изображения. Поэтому имеет смысл создать массив, задающий для каждой строки изображения смещение ее первого пиксела относительно начала файла.



Рисунок 1.19 - Алгоритм распаковки TIFF файлов

Строка изображения, как указывалось выше, будет состоять из 6104 пикселей, из которых первые 32 следует пропускать, то есть не отображать на форме. Оставшиеся 6072 пиксела принадлежат 6-ти смежным полосам изображения шириной 1012 пикселей каждая. Каждая полоса порождается отдельной ПЗС матрицей в составе МОЭП.

*Для упрощения задачи отображения содержимого файла* примем следующее ограничение на сложность задачи: в данный момент отображать на форме фрагмент только одной из 6-ти полос, но любой из этих полос, имея возможность оперативно переключаться от текущей полосы к любой другой.

### *Процедура «Извлечение свойств изображения»*

procedure ImagePropertesExtract( var FP: file );

var

TagNum: word;

OffsetsForSeek: dword;

ForShowMessage: string;

Begin

ForShowMessage:= '';

// Освободить память, выделенную под динамический массивы:

if StripCountWasSet

then SetLength( StripOffsetsArr, 0 );

StripCount:= 0;

StripCountWasSet:= false;

RowsPerStripWasSet:= false;

for TagNum:= 0 to TiffIFD.NumDirEntries - 1 do

// Анализ идентификатора тега:

case TiffTagArr[ TagNum ].TagID of

254: // NewSubFileType - ???

if TiffTagArr[ TagNum ].DataCount = 1

then NewSubFileType:= TiffTagArr[ TagNum ].DataOffset;

256: // ImageWidth

begin

ImageWidth:= TiffTagArr[ TagNum ].DataOffset;

MatrixNum:= ImgW div MatrixW;

ImgWInByte := MatrixW2 \* MatrixNum + 64;

end; // тег 256

257: // ImageHeight

begin

ImageHeight:= TiffTagArr[ TagNum ].DataOffset;

if ImgRowsOffsetsArrWasCreated

then

begin

SetLength( ImgRowsOffsetsArr, 0 );

ImgRowsOffsetsArrWasCreated:= false

end;

SetLength( ImgRowsOffsetsArr, ImgH );

end; // тег 257

258: // BitsPerSample - бит на каждый код цвета (яркости)

if TiffTagArr[ TagNum ].DataOffset = 16

then

BitsPerSample:= 16

else

begin

ShowMessage('Параметр BitsPerSample <> 16'); +

HALT; // - завершение работы программы

end; // тег 258

259: // Compression

begin

Compression:= TiffTagArr[ TagNum ].DataOffset;

if Compression <> 1

then

begin

ShowMessage('Программа работает только' + #13 +

'с изображениями без сжатия.' + #13 +

'Программа завершает работу!' );

HALT // - завершение работы программы

end

end; // тег 259

273: // StripOffsets - тег массива смещений данных

// о полосах

begin

StripCount:= TiffTagArr[ TagNum ].DataCount;

StripCountWasSet:= true;

SetLength( StripOffsetsArr, StripCount );

StripOffsets:= TiffTagArr[ TagNum].DataOffset;

if StripCount = 1

then StripOffsetsArr[ 0 ]:= StripOffsets

else // Количество Полос > 1

begin // Одним чтением заполнить весь массив

// смещений полос:

OffsetsForSeek:= StripOffsets shr 1;

Seek( FP, OffsetsForSeek );

BlockRead( FP, StripOffsetsArr,

StripCount shl 1 );

end;

end; // тег 273

277: // SamplesPerPixel - кодов (цветов) на пиксел

begin

SamplesPerPixel:= TiffTagArr[ TagNum ].DataOffset;

if SamplesPerPixel <> 1

then

begin

ShowMessage('Это цветное изображение!' + #13 +

'Программа завершает работу!' );

HALT // - завершение работы программы

end

end; // тег 277

278: // RowsPerStrip

begin

RowsPerStrip:= TiffTagArr[ TagNum ].DataOffset;

RowsPerStripWasSet:= true;

end; // тег 278

// 279: StripByteCounts- количество байт в каждой из полос.

// Параметр не востребован!!!

280: begin

MinSampleValue:= TiffTagArr[ TagNum ].DataOffset;

ForShowMessage:= ForShowMessage +

' Min яркость= ' +

IntToStr( MinSampleValue )

end; // тег 280

281: begin

MaxSampleValue:= TiffTagArr[ TagNum ].DataOffset;

ForShowMessage:= ForShowMessage +

' Max яркость= ' +

IntToStr( MaxSampleValue )

end;

end; // case TagID

if not StripCountWasSet

then

begin

ShowMessage('В файле отсутствует необходимый тэг ' + #13 +

'с номером 273.' + #13 +

'Программа завершает работу.' );

HALT // - завершение работы программы

end;

if not RowsPerStripWasSet

then if StripCount < 2

then RowsPerStrip:= ImageHeight

else

begin

ShowMessage(

'В файле ' + IntToStr( StripCount ) +

'полос(ы) и отсутствует тег 278, задающий’+ #13 +

'количество строк растра в каждой из полос.' + #13 +

'Значение RowsPerStrip будет вычислено по формуле:'

+ #13 +

'RowsPerStrip = Количество\_Строк / Количество\_Полос'

+ #13 + 'Вычисленное значение может быть неверным.');

RowsPerStrip:= ImageHeight div StripCount;

end;

ShowMessage( ForShowMessage + #13 +

'Количество матриц =' + IntToStr(MatrixNum) + #13 +

'Ширина изображения =' + IntToStr(ImageWidth) + #13 +

'Высота изображения =' + IntToStr(ImageHeight) + #13 +

'Бит на код цвета =' + IntToStr(BitsPerSample) + #13 +

'Полос изображения =' + IntToStr(StripCount) + #13 +

'Смещение первой полосы =' + IntToStr(StripOffsetsArr[0])

+ #13 +

'Строк растра в полосе =' + IntToStr(RowsPerStrip));

End; // ImagePropertesExtract( FP: file )

Процесс разработки среды может сопровождаться консультациями с преподавателем.

# 1.4 Контрольные вопросы

1. Перечислите основные отличительные особенности изображений ДЗЗ.

2. Опишите структуру пиксела в формате True Color. Какие другие форматы пикселов Вам известны?

3. Как следует формировать код пиксела формата True Color для отображения на форме монохромного изображения в цветах серой палитры?

4. Какие проблемы возникают при отображении на форме изображения с разрешением по глубине 10 бит/пиксел? Как решить эти проблемы?

5. Что понимается под принципами «младший в младшем» и «старший в младшем» относительно расположения структур данных в оперативной памяти компьютера?

6. Какой принцип расположения структур данных в оперативной памяти компьютера поддерживается процессорами фирмы Intel?

7. Как директория файла (IFD) может располагаться в файле относительно данных растра?

8. Какая информация хранится в тегах tiff файла?

9. Какую информацию содержат теги RowsPerStrip, StripOffsets и SripByteCounts, SamplesPerPixel?

10. В чем особенность оптико-электронных преобразователей (ОЭП), обеспечивающих получение широкоформатных снимков Земной поверхности?

11. Почему при визуализации изображения ДЗЗ высокого разрешения следует пропускать по 32 первых пиксела в каждой строке изображения?

# Лабораторная работа №2 Масштабирование растровых изображений

**Цель работы:** приобретения навыков программирования базовых средств для работы с файлами и изображениями ДЗЗ.

# 2.1 Инструмент «Лупа»

# 2.1.1 Применение лупы и варианты реализации

В лабораторной работе №2 требуется расширить функциональность среды, разработанной при выполнении лабораторной работы №1, дополнив эту среду следующими средствами:

- «лупой» переменного увеличения;

- окном для построения обзорного изображения;

- локатором-навигатором по обзорному изображению для выбора и указания текущего фрагмента изображения на обзорном изображении.

Инструмент «Лупа» позволяет разглядывать изображения в увеличенном геометрическом масштабе и в расширенном яркостном диапазоне (рисунок 2.1). Этот инструмент особенно полезен для дешифровщиков изображений.

На рисунке 2.1 *а*) приведен фрагмент изображения в том яркостном диапазоне и масштабе, в котором он задан в исходном файле. На рисунке 2.1 *б*) показан фрагмент изображения с рисунка *а*) в увеличенном масштабе и с нормированием яркостного диапазона пикселей. В данном случае инструмент «Лупа» позволил увидеть на изображении объект «трактор с прицепом», который на исходном изображении практически не виден! Поиск объектов на изображениях составляет одну из задач дешифрования изображений.

На рисунке 2.2 показаны два примера увеличения фрагмента изображения.



|  |  |
| --- | --- |
| *а*) | *б*) |

Рисунок 2.1 – Пример увеличения фрагмента

|  |  |
| --- | --- |
|  |  |
| *а*) – фрагмент увеличенного изображения с исходным яркостным диапазоном | *б*) – фрагмент увеличенного изображения с яркостным диапазоном, приведенным к диапазону [0..255] |

Рисунок 2.2 – Два варианта увеличения фрагмента изображения

На рисунке 2.3 в левом верхнем углу показан фрагмент изображения в исходном масштабе и с исходными яркостями пикселов. Здесь под исходным масштабом понимается такое отображение содержимого файла, при котором каждому коду пиксела в файле соответствует один пиксел изображения на форме. Под исходными яркостями пикселов понимаются коды пикселов, извлеченные из файла, с поправкой на необходимость сдвигать эти коды в процессе отображения на форму вправо на 0 – 2 разряда.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Без нормирования  яркости | С нормированием  яркости |  |
| Метод  ближайшего соседа |  |  | Метод ближайшего соседа порождает «*пикселизацию*» изображения – неприятный визуальный эффект |
| Билинейная интерполяция |  |  |

Рисунок 2.3 – Четыре варианта увеличения фрагмента изображения

Остальные 4 изображения на рисунке 2.3 демонстрируют увеличенный в 5 раз фрагмент при различных настройках процедуры «Лупа». Здесь под *нормированием яркости* понимается приведения кодов яркости пикселов к диапазону 0..255 единиц яркости. Сброшенный флаг «Интерполировать» означает, что при увеличении изображения яркости новых пикселов выбирались по методу *ближайшего соседа*. При установленном флаге «Интерполировать» яркости новых пикселов вычислялись методом *билинейной интерполяции*. Рассмотрим эти два простейших метода увеличения размеров изображения.

# 2.1.2 Лупа на базе метода ближайшего соседа

Будем считать, что увеличить изображение в *m* раз означает, что следует увеличить в *m* раз линейный размер изображения и это увеличение одинакового по координатным осям. При этом увеличению фрагмента изображения (или всего изображения) в *m* раз соответствует увеличение количества пикселов увеличиваемого фрагмента в *m*2 раз.

Будем считать, что число *m* всегда целое. Это ограничение не уменьшает полезность получаемой «лупы» и одновременно несколько упрощает программную реализацию процесса увеличения изображения.

На рисунке 2.4 показано, как некоторые 4 смежные пикселы исходного изображения должны располагаться среди пикселов изображения увеличенного в 5 раз.

На рисунке 2.5 показано как закрашиваются пикселы увеличенного изображения при использовании метода ближайшего соседа: для каждого нового пиксела находится ближайший пиксел исходного изображения и его яркость назначается новому пикселу.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 1 |  |  |  |  | 2 |  |  |
| 1 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | **3** |  |  |  |  | 4 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Рисунок 2.4 – Увеличение изображения   
в масштабе *m* = 5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 1 |  |  |  |  | 2 |  |  |
| 1 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | **3** |  |  |  |  | 4 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Рисунок 2.5 – Увеличение изображения   
в масштабе *m* = 5 методом ближайшего соседа

Метод ближайшего соседа при целочисленном значении масштаба *m* эквивалентен простому тиражированию яркости каждого исходного пиксела на *m*2 пикселов увеличенного изображения.

Внимательный и наблюдательный читатель может заметить, что пикселы исходного изображения на рисунке 2.4 оказались в центре квадратов размером 5×5 пикселов только потому, что 5 число нечетное. Поэтому естественно поставить вопрос: как эти исходные пикселы будут расположены среди пикселов увеличенного изображения при четном значении масштаба? Например, при *m* = 4? Вопрос иллюстрируется рисунком 2.6 *а*).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | 1? | 1? |  |  | 2? | 2? |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | 1? | 1? |  |  | 2? | 2? |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | **3?** | **3?** |  |  | 4? | 4? |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | **3?** | **3?** |  |  | 4? | 4? |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | *а*) | | | | | | | |  |  |  | *б*) | | | | | | | |

Рисунок 2.6 – Увеличение изображения   
в масштабе *m* = 4

При реализации метода ближайшего соседа простым размножением пикселов исходного изображения следует просто каждый исходный пиксел повторить *m*2 раз: при *m*= 4 повторить 16 раз (рисунок 2.6 *б*). При этом, понятно, не имеет значения какое положение в квадрате 4×4 пиксела занимает исходный пиксел!

# 2.1.3 Лупа на базе билинейной субпиксельной интерполяции

Достоинства метода билинейной субпиксельной интерполяции:

- не порождает пикселизацию увеличенного изображения;

- обеспечивает плавность изменения яркостей пикселов добавленных в увеличенное изображение;

- является простейшим среди методов, обладающих выше перечисленными достоинствами.

*Под субпиксельной интерполяцией понимается* вычисление яркости пиксела (отсутствующего на исходном изображении) смещенного относительно пиксельной сетки исходного изображения.

На рисунке 2.7 изображены 4 смежных пиксела исходного изображения и положение пиксела, изображенного тонкими линиями, яркость которого требуется вычислить.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | 1 |  |  |  |  | 2 |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | **3** |  |  |  |  | 4 |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

При формировании изображения яркость каждого пиксела формируется под действием лучистой энергии отраженной от некоторого участка поверхности фотографируемой сцены. Понятно, что существует бесконечное множество вариантов распределения этой лучистой энергии по поверхности, каждый из которых создаст данную яркость пиксела. Поэтому понятно, что задача субпиксельной интерполяции не корректна по сути: она не может быть решена точно! То есть, любой метод субпиксельной интерполяции может дать только приближенное в той или иной степени решение.

Рисунок 2.7 – К задаче   
субпиксельной   
интерпо­ляции

Все методы интерполяции строятся на основе некоторой гипотезы о законе изменения яркости пиксела при его смещении относительно имеющейся пиксельной сетки. Метод билинейной интерполяции строится на основе предположения об указанном законе как о линейном. То есть предполагается, что если пиксел, например 1-й на рисунке 2.7, смещать непрерывно из своего положения в положение пиксела 2 по кратчайшему пути (горизонтально), то в промежуточных положениях его яркость будет плавно по линейному закону от величины смещения изменяться от исходной яркости до яркости 2-го пиксела.

Пиксел – это не делимый элемент изображения: он не имеет структуры и яркость всех его точек одинакова. Исходя из такого представления и из рисунка 2.7 ясно, что при вычислении яркости смещенного пиксела следует принимать во внимание яркости 4-х смежных пикселей, фрагменты которых накрывает этот смещенный пиксел. Причем, яркости исходных пикселей должны вносить вклад в яркость нового пикселе тем меньшую, чем меньший фрагмент исходного пиксела входит в смещенный пиксел.

Если яркости исходных пикселей обозначить как *I*1, …, *I*4, а площади их частей, вошедших в смещенный пиксел, обозначить как *S*1, …, *S*4 соответственно, то искомую яркость *I* смещенного пиксела можно вычислить по формуле

*I* = *I*1 *S*1 + *I*2 *S*2 + *I*3 *S*3 + *I*4 *S*4. (2.1)

В методе билинейной интерполяции используется иная формула, дающая эквивалентный результат.

По координатам центра смещенного пиксела определяются координаты 4-х пикселов, между которыми находится смещенный пиксел (рисунок 2.8). В центре левого верхнего пиксела из 4-х найденных задается начало локальной системы координат (СК) *OX*л*Y*л. В этой СК центры 4-х пикселей имеют координаты, указанные на рисунке. Центр смещенного пиксела расположен в точке (*x*л, *y*л) и (0 ≤ *x*л ≤ 1) Λ (0 ≤ *y*л ≤ 1).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  | *I*1  (1, 0)  (1, 0) | |  |  |  | *I*2 |  | |  |
| (0, 0) | | |  |  |  |  |  | | *X*л |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | (*x*л, *y*л) |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| (0, 1) | | |  |  |  |  |  |  |  |
|  | *I*3 |  |  |  |  | *I*4 |  |  |  |
|  |  | *Y*л |  |  |  |  |  |  |  |

Введем обозначение:

*I*(*x*, *y*) – яркость пиксела с локаль­ными координатами (*x*, *y*).

Тогда:

*I*(0, 0)= *I*1; *I*(1, 0)= *I*2; *I*(0, 1)= *I*3; *I*(1, 1)= *I*4.

В методе билинейной интерполяции искомая яркость *I*(*x*л, *y*л) смещенного пиксела вычисляется по следующей формуле:

*I*(*x*л, *y*л) = *a x*л + *b y*л + *c x*л *y*л – *d*. (2.2) (2.2)

Рисунок 2.8 – Билинейная   
субпиксельная интерпо­ляции

Формулы для вычисления значений коэффициентов a, b, c и d получим, последовательно подставляя в (2.2) вместо *x*л и *y*л координаты 4-х найденных пикселов:

*I*1 *= I*(0, 0) = *a⋅* 0 + *b⋅* 0 + *c⋅* 0*⋅* 0 + *d* = *d*;

*I*2 *= I*(1, 0) = *a⋅* 1 + *b⋅* 0 + *c⋅* 1*⋅* 0 + *d* = *a* + *d*;

*I*3 *= I*(0, 1) = *a⋅* 0 + *b⋅* 1 + *c⋅* 0*⋅* 1 + *d* = *b* + *d*; (2.3)

*I*4 *= I*(1, 1) = *a⋅* 1 + *b⋅* 1 + *c⋅* 1*⋅* 1 + *d* = *a* + *b* + *c* + *d*.

Из (2.3) находим:

*d = I*1;

*a* = *I*2 - *d*;

*b* = *I*3 - *d*; (2.4)

*c* = *I*4 - *a - b - d*.

Рисунок 2.9 иллюстрирует происхождение термина «билинейная интерполяция»: точки поверхности, образованной интерполированными значениями яркости для всех точек в пределах (0 ≤ xл ≤ 1) Λ (0 ≤ yл ≤ 1), задают линейчатую поверхность: поверхность, созданную движением прямой линии. Действительно, если в (2.2) значение одной из координат xл, yл зафиксировать, то выражение для I(xл, yл) преобразуется в линейную функцию другой координаты.

*X*л

*Y*л

*Z*л

*I* (0,0)

*I* (1,0)

*I* (0,1)

*I* (1,1)

(0,0)

(0,1)

(1,0)

(1,1)

*I*(*х*л, *у*л)

(*х*л,*у*л)

Рисунок 2.9 – Линейчатая поверхность   
яркостей смещенных пикселов, вычисляемых   
по билинейному закону интерполяции

# 2.1.4 Нормирование яркости пикселей изображения

Под *нормированием яркости* понимается приведения кодов яркости пикселов к некоторому диапазону. В рассмотренных выше примерах нормирование яркости пикселей увеличенных фрагментов изображений осуществлялось приведением этих яркостей к диапазону [0, 255] единиц. Именно в таком диапазоне изменяются коды каждого из трех основных цветов в формате True Color.

Процедура нормирования яркостей поясняется рисунком 2.10.

0

255

0

*min*

*max*

1023

Рисунок 2.10 – Нормирование яркостей пикселей

На рисунке 2.10:

- [0, 1023] – диапазон значений 10-битного кода;

- [0, 255] – диапазон значений байта;

- [*min*, *max*] – диапазон значений кодов яркостей пикселей исходного изображения;

Пунктирные стрелки показывают, что все пикселы исходного изображения, имеющие яркость равную *min*, при отображении в поле лупы должны иметь яркость равную 0; все пикселы исходного изображения, с яркостью *max* в поле лупы должны иметь яркость равную 255.

Вывод формул и разработку алгоритмов, реализующих описанное нормирование яркостей пикселов поля «Лупа», обучающиеся выполняют самостоятельно.

# 2.2 Обзорное изображение

Обзорное изображение (ОИ) – это изображение, уменьшенное до размеров, позволяющих видеть на экране по возможности сразу все изображение, записанное в файле. Пример такого ОИ дан на рисунке 2.11.



Рисунок 2.11 –Обзорное изображение в пакете ENVI

ОИ позволяет видеть «картину» целиком. Оно особенно актуально именно в системах обработки изображений ДЗЗ, поскольку такие изображения отличаются огромными размерами.

В системах обработки изображений ДЗЗ ОИ используются для навигации по всему изображению. Щелчок на поле ОИ приводит к отображению в окне изображения высокого разрешения соответствующего фрагмента. И наоборот, прокрутка изображения высокого изображения сопровождается перемещением указателя-локатора на поле ОИ.

Часто обзорные изображения подготавливаются заранее и представляются отдельным файлом. В пакете ENVI, упомянутом на рисунке 2.11, обзорное изображение можно пересчитывать с желаемым коэффициентом масштабирования в процессе работы с исходным изображением высокого разрешения.

ОИ могут строиться как минимум двумя способами.

Способ 1 отличается алгоритмической простотой, высоким быстродействием и одновременно сравнительно низким качеством получаемого изображения. Способ состоит в прореживании пикселов исходного изображения: выбирается только каждая *m*-я строка, а в ней каждый *m*-й пиксел.

Способ 2 состоит в вычислении простейших сверток изображения блоками размером *m× m* пикселов. Отдельная свертка состоит в вычислении среднего значения яркости по всем пикселам блока. Это среднее значение и отображается на поле ОИ.

# 2.3 Задание

Расширить функциональные возможности разработанной ранее среды, дополнив ее следующими инструментами:

- «лупой» переменного увеличения;

- окном для построения обзорного изображения.

Задание на разработку локатора-навигатора по обзорному изображению для выбора и указания текущего фрагмента изображения можно давать отдельным активным обучающимся в качестве дополнительного задания.

# 2.4 Методика выполнения работы

1. Изучить теоретическую часть этих методических указаний.

2. Дополнить интерфейсную часть разрабатываемой среды. Для этого:

- выбрать и расположить на форме необходимые дополнительные визуальные компоненты;

- все имена компонент, назначаемые автоматически, переименовать на логически содержательные имена, отражающие назначение соответствующей компоненты;

- продумать какие события, связанные с использованием интерфейсных средств, необходимо предусмотреть в программе;

- создать заготовки для обработчиков выбранных событий;

- все имена обработчиков событий, назначенные средой разработки автоматически, переименовать на логически содержательные имена, отражающие задачи, решаемые каждым обработчиком.

3. Приступить к программированию обработчиков событий, начиная с разработки алгоритма отображения фрагмента в поле «Лупа».

4. Для «размножения» пикселов в процессе увеличения изображения на основе билинейной интерполяции необходимо:

- начало локальной СК (рисунок 2.8) последовательно позиционировать в центр каждого пиксела увеличиваемого фрагмента изображения;

- в каждом положении СК последовательно задавать все возможные точки с координатами (*x*л, *y*л), вычисляемыми по формулам

*x*л = *i/m; y*л = *j/m*, (2.5)

где *i*, *j* – целые числа интервала [0, *m*-1].

- для каждой заданной точки вычислить значение интерполиро­ванной яркости по формулам (2.3), (2.4).

Полезно заметить, что при переходе от точки с координатами (*x*л, *y*л) к следующей точке с координатами, например, (*x*л + 1/ *m*, *y*л) интерполированное значение яркости возрастет на величину *a/m*:

*I*(*x*л + 1/ *m*, *y*л) = *I*(*x*л, *y*л) + *a/m*. (2.6)

Использование указанной зависимости позволит сократить вычислительную трудоемкость процесса билинейной интерполяции.

# 2.5 Контрольные вопросы

1. Что понимается под инструментом «Лупа»?

2. Какой вариант нормирования яркости пикселей рассмотрен в данной лабораторной работе?

3. Напишите формулу, нормирующую яркость пиксела при его отображении в поле лупы.

4. Что понимается под интерполяцией изображения?

5. Объясните суть метода ближайшего соседа.

6. Что такое пикселизация изображения при его увеличении?

7. Приведите два варианта формул субпиксельной интерполяции изображения.

8. Какую поверхность интерполированных яркостей пикселей порождает билинейная интерполяция?

9. Объясните суть двух способов получения обзорного изображения.

10. По каким формулам рассчитываются координаты точек локальной системы координат при увеличении изображения в *m* раз?

11. На сколько отличаются интерполированные яркости двух ближайших в локальной системе координат точек при увеличении изображения в *m* раз?

# Лабораторная работа №3 Гистограмма яркостей пикселей

**Цель работы:** приобретения навыков программирования базовых средств для работы с изображениями ДЗЗ на примере формирования и коррекции гистограммы яркостей пикселей.

# 3.1 Пример гистограммы

Гистограмма – это столбчатая диаграмма, которая для каждой возможной яркости пиксела указывает количество пикселов данной яркости. То есть, гистограмма строится в системе координат *яркость пиксела* (по оси абсцисс), *количество пикселей* (по оси ординат) (рисунок 3.1).

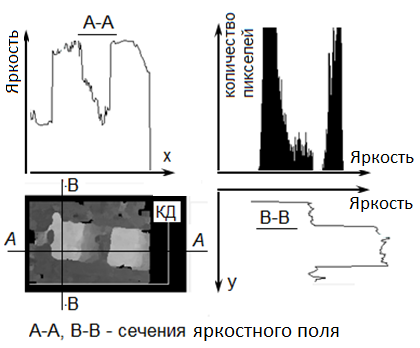
****

Рисунок 3.1 –Яркостное поле   
и его гистограмма яркостей

Гистограмма яркостей изображения содержит в интегральном виде важную информацию об изображении и поэтому средства ее построения имеются в каждой системе обработки изображений ДЗЗ.

# 3.2 Инструменты для коррекции гистограмм

Поле для отображения гистограмм обычно содержит дополнительные инструменты, с помощью которых, управляя гистограммой, корректируют само изображение с целью изменения яркостного диапазона пикселей, изменения контраста изображения, выполнения как линейных, так и нелинейных преобразований яркостей пикселей.

# 3.3 Задание

Расширить функциональные возможности разработанной ранее среды, дополнив ее следующими инструментами:

- полем «Гистограмма»;

- на поле «Гистограмма» создать два движка (рисунок 3.2) для отсечения и выделения яркостных диапазонов, манипулирование которыми сопровождается динамическим пересчетом соответствующего изображения или его фрагмента.

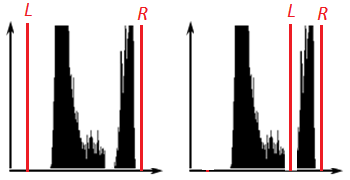


Рисунок 3.2 – Движки *L* и *R*   
управления гистограммой

Математические модели, сопровождающие разрабатываемые средства управления гистограммой, обучающиеся разрабатывают самостоятельно, консультируясь, при необходимости, с преподавателем.

На рисунке 3.2 показаны два движка обозначенные буквами *L* (left - левый) и *R* (right - правый). Это должны быть перемещаемые движки, положение которых на шкале яркостей диапазона [0, 255] должно отображаться как визуально на поле гистограммы, так и в числовом виде в соответствующих полях. Захват и перемещение движков должно выполняться с помощью курсора мыши. При этом разумно предусмотреть вокруг движков зону в 2-3 пиксела (2-3 единицы яркости) по обе стороны от движков, при попадании курсора в которую движок может быть «захвачен».

Движок *R* предназначен для отсечения яркостей, значения которых расположены правее этого движка. Движок *L* отсекает яркости левее своего положения. При этом отсечение яркостей надо осуществлять в нескольких вариантах. Все яркости правее движка *R* (левее движка *L*) заменять по следующим правилам:

- считать равными *R* (равными *L*);

- считать равными максимальному значению 255 (минимальному значению 0);

- диапазон [*L*, *R*] отображать в диапазон яркостей пикселей изображения равный [0, 255].

Принятие решения об использовании одного из указанных вариантов отсечения осуществлять для одного движка независимо от другого.

Предусмотреть вариант фиксации взаимного положения движков, при котором захват одного из них означает одновременный захват другого. В таком режиме удобно препарировать изображение заданным яркостным диапазоном.

В качестве дополнительного задания предлагается реализовать два поля для отображения горизонтального и вертикального сечений изображения в виде графиков яркостей пикселей в соответствующих сечениях. Примеры таких сечений приведены на рисунке 3.1. Можно предусмотреть режим построения гистограмм яркостей в этих сечениях.

# 3.4 Методика выполнения работы

Первые шаги выполнения лабораторной работы, связанные с разработкой интерфейса и заготовок обработчиков событий, описаны в методиках к предыдущим работам.

Приступая к программированию обработчиков событий, необходимо разработать соответствующие математические модели и алгоритмы из реализации.

# 3.5 Контрольные вопросы

1. Что такое гистограмма яркостей пикселей?

2. Покажите в программе структуру данных, поддерживающую операции с гистограммой.

3. Какие основные методы были разработаны для работы с гистограммами?

4. Перечислите варианты отсечения яркостей с использованием движков на гистограмме.

5. Приведите формулы для реализации отображения диапазона [*L*, *R*] яркостей пикселей исходного изображения в диапазон яркостей [0, 255].

6. Какие программные средства (методы, приемы) использованы для улучшения интерактивности программы?

# Лабораторная работа №4 Свертки и фильтрация изображений

**Цель работы:** приобретения навыков программирования типовых сверток изображений и изучение их свойств.

# 4.1 Шумы и фильтры изображений

# 4.1.1 Цифровые маски-фильтры

Под шумами изображений понимаются разнообразные дефекты и ошибки в изображениях.

Любое изображение содержит в себе шумы. Например, современные изображения ДЗЗ создаются с помощью оптико-электронных преобразователей на базе ПЗС матриц. В составе ПЗС матрицы имеются десятки и сотни тысячи фоточувствительных ячеек. Каждая ячейка характеризуется собственной передаточной характеристикой энергии входящего (освещающего) светового потока в электрический сигнал. Различия в этой характеристике приводят к формированию разными ячейками при одинаковой облученности разных накопленных внутри ячейки зарядов. Это уже причина появления шумов.

С целью подавления шумов изображения обрабатывают с помощью различных цифровых фильтров. Такая обработка называется фильтра­цией.

Любая фильтрация основана на использовании некоторых цифровых масок. Маска представляет собой двумерный числовой массив с нечетными размерами. Но находят применение в качестве масок и одномерные массивы. Числовое значение элемента такого массива называется весом элемента маски. Нечетное количество строк и столбцов в массиве позволяет выделить в маске центральный элемент.

Использование маски состоит в том, что ее центр последовательно совмещают с каждым пикселом фильтруемого изображения. При этом считается, что соседние элементы маски накрывают соседние пикселы. На каждом совмещении пикселу исходного изображения сопоставляется новое значение яркости, вычисленное путем суммирования произведения яркостей всех пикселов, накрытых маской, на веса элементов маски, накрывающих пиксел. Вычисленная сумма произведений обычно делится на сумму весов всех элементов маски. В частных случаях маска может быть задана массивом дробных чисел, сумма которых равна 1. В таком случае операция деления не выполняется. Но предпочтительно в отношении простоты и скорости обработки изображения использовать маски с целочисленными весами ее элементов.

# 4.1.2 Примеры цифровых масок-фильтов

Простейшими являются усредняющие (арифметические) фильтры.

На рисунке 4.1 приведены варианты фильтров, задаваемых масками размером 3×3 элемента.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 1 | 1 |  |  |  | 1 | 2 | 1 |  |  |  | -1 | -1 | -1 |  |  |  | 0 | 1 | 0 |  |
| 1/9 | × | 1 | 1 | 1 |  | 1/16 | × | 2 | 4 | 2 |  |  |  | -1 | 9 | -1 |  |  |  | 1 | -4 | 1 |  |
|  |  | 1 | 1 | 1 |  |  |  | 1 | 2 | 1 |  |  |  | -1 | -1 | -1 |  |  |  | 0 | 1 | 0 |  |
|  |  |  | *а*) |  |  |  |  |  | *б*) |  |  |  |  |  | *в*) |  |  |  |  |  | *г*) |  |  |

а) – арифметический фильтр;

б) – фильтр Гаусса;

в) – контрастоповышающий фильтр;

г) - дифференциальный оператор Лапласа (лапласиан)

Рисунок 4.1 – Примеры масок-фильтров

На рисунке 4.2 приведены варианты фильтров, задаваемых масками размером 5×5 элементов. Маска фильтра *а*) не требует дополнительного множителя, поскольку сумма ее элементов равна 1.

Фильтры Гаусса, подобные приведенному на рисунке 4.2, строятся на основе целочисленной аппроксимации функции Гаусса (4.1):

, (4.1)

где под *Ouv* понимается локальная система координат маски, начало которой совмещено с центральным элементом маски;

σ - параметр, определяющий крутизну поверхности, построенной в системе координат *Ouvh*: *h* – высота точки поверхности над плоскостью *Ouv*.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.00954 | 0.01527 | 0.01908 | 0.01527 | 0.00954 |  |  |  | 2 | 4 | 5 | 4 | 2 |
| 0.01527 | 0.03817 | 0.07634 | 0.03817 | 0.01527 |  |  |  | 4 | 9 | 12 | 9 | 4 |
| 0.01908 | 0.07634 | 0.30534 | 0.07634 | 0.01908 |  | 1/159 | × | 5 | 12 | 15 | 12 | 5 |
| 0.01527 | 0.03817 | 0.07634 | 0.03817 | 0.01527 |  |  |  | 4 | 9 | 12 | 9 | 4 |
| 0.00954 | 0.01527 | 0.01908 | 0.01527 | 0.00954 |  |  |  | 2 | 4 | 5 | 4 | 2 |
|  |  | *а*) |  |  |  |  |  |  |  | *б*) |  |  |

Рисунок 4.2 – Два примера фильтров Гаусса в виде масок 5×5 элементов

Два фильтра на рисунке 4.2 построены для разных значений параметра σ, поэтому соотношения между парой элементов, одинаково расположенных в этих масках, различны.

Интересным вариантом фильтрации является использование медианного фильтра. При своей арифметической простоте он порождает нелинейную фильтрацию и одновременно с этим, в отличие от ранее приведенных фильтров, не требует применения какой-либо числовой заранее синтезированной маски.

Медианный фильтр заменяет яркость пиксела под центральным пикселом окна, накладываемого на изображение как маска, на медиану значений яркостей всех пикселов, попавших в окно. Работа медианного фильтра при использовании окна размером 3×3 элемента поясняется рисунком 4.3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 18 | 22 | 25 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 16 | **26** | 24 | 🡪 сортировка 🡪 | | | | | | 12 | 15 | 16 | 18 | **20** | 22 | 24 | 25 | 26 | 🡪20 |  |  |
|  | 12 | 15 | 20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Рисунок 4.3 – Медианный фильтр

Пусть под окном размером 3×3 элемента находятся пикселы с яркостями, показанными на рисунке 4.3. Яркости этих пикселов сортируются, например, по возрастанию. Затем в качестве нового значения яркости пиксела под центральным элементом окна выбирается яркость центрального (в данном случае пятого) элемента отсортирован­ного массива яркостей. То есть центральный пиксел окна, имевший яркость 26 единиц будет иметь яркость 20 единиц.

# 4.2 Задание

Расширить функциональные возможности среды, разработанной при выполнении предыдущих работ, добавлением инструментов для фильтрации изображений с помощью рассмотренных выше типов фильтров.

Относительно фильтров Гаусса и лапласиана требуется предусмотреть возможность задания параметров, входящих в математическую модель соответствующего фильтра.

Предусмотреть возможность зашумления изображения шумами типа «соль», «перец» и другими по заданию преподавателя.

# 4.3 Методика выполнения работы

Первые шаги выполнения лабораторной работы, связанные с разработкой интерфейса и заготовок обработчиков событий, описаны в методиках к предыдущим работам.

Приступая к программированию обработчиков событий, необходимо разработать соответствующие математические модели и алгоритмы из реализации.

Понятно, что вычисленное только что новое значение яркости какого-то пиксела не должно участвовать в вычислении яркостей других пикселов.

# 4.4 Контрольные вопросы

1. С какой целью применяется фильтрация изображений?

2. Приведите пример источника возникновения шумов изображения, дополнительно к указанному выше.

3. Что такое маска и как она используется при фильтрации.

4. Приведите примеры не менее четырех типов фильтров.

5. Как работает медианный фильтр?

6. Как реализуется дифференцирование изображения?

7. Что такое оператор Собеля?

8. Что понимается под градиентом яркости?

9. Какой тип фильтра лучше отфильтровывал искусственно привнесенные в изображение шумы?

# 5 Содержание отчетов по лабораторным работам

По каждой выполненной лабораторной работе представляется отчет, содержащий:

- разработанное программное обеспечение (сдается преподавателю в электронном виде со всеми исходными и исполняемыми файлами);

- пояснительную записку в формате docx как в электронном, так и в машинописном видах.

В исходных текстах программ обязательно должны присутствовать комментарии, достаточные для последующего сопровождения программы другим программистом.

Содержание пояснительной записки.

1. Титульный лист с указанием названий министерства, университета, кафедры, номера лабораторной работы, ее темы и названия дисциплины. Указываются также автор работы, преподаватель ее проверивший, поле для оценки, год выполнения работы.
2. Формулируются тема, цель и задание на разработку.
3. Описываются основные проектные решения:

- выбранные методы и математические модели, положенные в основу программного продукта;

- разработанные структуры данных и основные программные методы.

4. Приводится листинг основных разработанных методов. Методы, описанные в предыдущих отчетах, повторно описывать не следует.

5. Примеры экранных форм с результатами работы программы.

# Библиографический Список

1 Д. Мюррей, У. Ван Райпер. Энциклопедия форматов графических файлов: пер. с англ. – К.: Издательская группа BHV, 1997. 672 с.

2 Кузнецов П.К. Методика высокоточной сшивки изображений, получаемых при съемке обстановки многомат­ричным сканирующим оптико-электронным преобразователем / Кузнецов П.К., Мартемьянов Б.В., Скирмунт В.К., Семавин В.И. // Вестник Самар. гос. техн. ун-та. Серия «Технические науки». - №3 (31). – Самара, 2011. – с.69-81.

3 Шапиро Лю Компьютерное зрение / Л. Шапиро, Дж. Стокман; Пер. с англ.; - М.: БИНОМ. Лаборатория знаний, 2006. – 752 с. ISBN 5-94774-384-1.

**Содержание**

[Введение 3](#_Toc433728997)

[Лабораторная работа №1 Визуализация изображений ДЗЗ 4](#_Toc433728998)

[1.1 Изображения ДЗЗ высокого разрешения 4](#_Toc433728999)

[1.1.1 Особенности изображений ДЗЗ высокого разрешения 4](#_Toc433729000)

[1.1.2 Структура tiff файлов 7](#_Toc433729001)

[1.2 Задание 36](#_Toc433729015)

[1.3 Методика выполнения работы 38](#_Toc433729017)

[1.4 Контрольные вопросы 45](#_Toc433729019)

[Лабораторная работа №2 Масштабирование растровых изображений 46](#_Toc433729020)

[2.1 Инструмент «Лупа» 46](#_Toc433729021)

[2.1.1 Применение лупы и варианты реализации 46](#_Toc433729022)

[2.1.2 Лупа на базе метода ближайшего соседа 49](#_Toc433729023)

[2.1.3 Лупа на базе билинейной субпиксельной интерполяции 51](#_Toc433729024)

[2.1.4 Нормирование яркости пикселей изображения 54](#_Toc433729025)

[2.2 Обзорное изображение 55](#_Toc433729026)

[2.3 Задание 56](#_Toc433729027)

[2.4 Методика выполнения работы 56](#_Toc433729028)

[2.5 Контрольные вопросы 58](#_Toc433729029)

[Лабораторная работа №3 Гистограмма яркостей пикселей 59](#_Toc433729030)

[3.1 Пример гистограммы 59](#_Toc433729031)

[3.2 Инструменты для коррекции гистограмм 60](#_Toc433729032)

[3.3 Задание 60](#_Toc433729033)

[3.4 Методика выполнения работы 62](#_Toc433729034)

[3.5 Контрольные вопросы 62](#_Toc433729035)

[Лабораторная работа №4 Свертки и фильтрация изображений 63](#_Toc433729036)

[4.1 Шумы и фильтры изображений 63](#_Toc433729037)

[4.1.1 Цифровые маски-фильтры 63](#_Toc433729038)

[4.1.2 Примеры цифровых масок-фильтов 64](#_Toc433729039)

[4.2 Задание 66](#_Toc433729040)

[4.3 Методика выполнения работы 66](#_Toc433729041)

[4.4 Контрольные вопросы 67](#_Toc433729042)

[5 Содержание отчетов по лабораторным работам 68](#_Toc433729043)

[Библиографический Список 69](#_Toc433729044)

*Учебное издание*

*МАРТЕМЬЯНОВ Борис Викторович*

**Системы анализа данных космического зондирования**

Методические указания

В авторской редакции

Подп. в печать \_\_\_\_\_\_\_\_

Формат \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Усл. п.л. 4,12. Уч.-изд. л. 4

Тираж \_\_\_\_\_\_\_\_. Рег. № \_\_\_\_\_\_

Федеральное государственное бюджетное

образовательное учреждение

высшего профессионального образования

«Самарский государственный технический университет»

443100, г. Самара, ул. Молодогвардейская, 244. Главный корпус