МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Название направления»

Профиль подготовки: «Название профиля подготовки бакалавра»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**Тема:**

**«Кодирование данных дистанционного зондирования земли по методу главных компонент»**

еВыполнил: студент группы

381507-3

Кошелев Д.И.

ученая. степень, ученое \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель:

к.т.н. Васин Д.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рецензент:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2019

Оглавление

[Оглавление 2](#_Toc9856811)

[Введение 4](#_Toc9856812)

[Актуальность темы 4](#_Toc9856813)

[1 Теоретическая часть 6](#_Toc9856814)

[1.1 Обзор существующих методов решения данной задачи 6](#_Toc9856815)

[1.1.1 Статистические методы сжатия 6](#_Toc9856816)

[1.1.2 Словарные методы сжатия 9](#_Toc9856817)

[1.1.3 Фрактальный алгоритм 10](#_Toc9856818)

[1.1.4 Алгоритм JPEG 11](#_Toc9856819)

[1.2 Форматы для сжатия изображений 13](#_Toc9856820)

[1.2.1 WinRar 13](#_Toc9856821)

[1.2.2 PNG 13](#_Toc9856822)

[1.2.3 TIFF 14](#_Toc9856823)

[1.3 Математическая модель 14](#_Toc9856824)

[1.4 Постановка задачи 15](#_Toc9856825)

[1.4.1 Входная информация 15](#_Toc9856826)

[1.4.2 Выходная информация 15](#_Toc9856827)

[2 Практическая часть 16](#_Toc9856828)

[2.1 Описание разработанных и используемых алгоритмов 16](#_Toc9856829)

[2.1.1 Алгоритм ColumnEze 16](#_Toc9856830)

[2.1.2 Алгоритм StringEze 17](#_Toc9856831)

[2.1.3 Алгоритм ColumnZmey 17](#_Toc9856832)

[2.1.4 Алгоритм StringZmey 19](#_Toc9856833)

[2.1.5 Алгоритм GetCovarMatrix 20](#_Toc9856834)

[2.1.6 Алгоритм RotationMethod //скелет 20](#_Toc9856835)

[2.1.7 Алгоритм Multiplication 21](#_Toc9856836)

[2.1.8 Алгоритм FromImageToTxt 21](#_Toc9856837)

[2.1.9 Алгоритм FromTxtToTB 21](#_Toc9856838)

[3 Список литературы 22](#_Toc9856839)

Введение

## Актуальность темы

Дистанционное зондирование - это способ получения информации об объекте на расстоянии без вступления с ним в прямой контакт, К методам дистанционного зондирования относятся все методы неконтактного получения информации, такие как сейсморазведка, гравиразведка ит.д. Среди них особое место занимают методы дистанционного зондирования Земли (ДЗЗ) из космоса.[1]

Под дистанционным зондированием поверхности Земли понимается наблюдение и измерение энергетических и поляризационных характеристик излучения объектов в различных диапазонах электромагнитного (ЭМ) спектра с целью определения местоположения, вида, свойств и временной изменчивости объектов окружающей среды без непосредственного контакта с ним измерительного прибора [1].

Гиперспектральная съемка является приоритетным направлением развития систем Д33[2]. Она предполагает одновременное получение сотен и тысяч изображений одной и той же сцены, зафиксированных в узких соприкасающихся диапазонах спектра. В результате формируется многомерное гиперспектральное изображение (ГСИ), называемое гиперкубом. Каждой точке земной поверхности гиперкуб ставит в соответствие спектральную характеристику (СХ), которая описывает распределение энергии излучения по длине волны. Знание СХ для множества точек вполне определенных объектов позволяет автоматизировать процесс сегментации объектов и установить их физико-химические свойства[3].

Поэтому материалы гиперспектральной съемки находят широкое применение во многих сферах человеческой деятельности: экологии, сельском и лесном хозяйствах, анализе атмосферы, поиске полезных ископаемых, военной сфере и др.

Актуальность решения проблем кодирования и сжатия ГСИ обусловлена увеличивающимся потоком значительных объёмов данных, поступающих со спутников и иных источников, требующих единовременной обработки для решения поставленных задач.

На данный момент существует множество алгоритмов и методов кодирования и сжатия ГСИ, однако, в большинстве из них имеются те или иные недостатки, которые не позволяют выделить какой-либо алгоритм кодирования и последующего сжатия данных как универсальный.

**Целью данной работы** является разработка и реализация программного обеспечения, способного кодировать и сокращать размерность гиперспектральных данных ДЗЗ.

Для достижения данной цели необходимо решить следующий **ряд задач**:

1. Изучить существующие форматы представления графической информации.
2. Изучить базовые способы обработки растровых изображений ДЗЗ.
3. Рассмотреть основные подходы в сжатии растровых данных.
4. Разработать информационную систему, принимающую на вход набор монохромных растровых изображений и возвращающую их закодированное и сжатое представление, в виде файла формата bin.

**Объектом исследования** являются гиперспектральные растровые изображения, представленные в цифровом виде, полученные как результат выполнения гиперспектрального сканирования земной поверхности.

1. Теоретическая часть
   1. Обзор существующих методов решения данной задачи
      1. Статистические методы сжатия

Статистические методы компрессии используют статистические свойства сжимаемых данных и присваивают всем символам коды с переменной длиной в зависимости от частоты появления символа во входном потоке данных. Основным примером статистических алгоритмов сжатия является известный алгоритм Хаффмана, который основывается на предположении, что разные символы встречаются в тексте с разной вероятностью, и присваивает символам с большей вероятностью коды наименьшей длины. Алгоритм Хаффмана имеет большое значение в теории алгоритмов сжатия и используется во многих программах сжатия данных [4,5].

Арифметическое сжатие превалирует над методом Хаффмана, поскольку именно данный алгоритм сжатия с различными модификациями используется для непосредственного кодирования данных. К модификациям алгоритма арифметического кодирования можно отнести алгоритм интервального кодирования [6]. В модификациях чаще всего увеличение скорости кодирования обуславливается уменьшением коэффициента сжатия.

При арифметическом кодировании данные представляются в виде определённого интервала.

Арифметический кодер используется для отображения накопленного значения счетчика кодируемого символа в определенный битовый код. При этом в качестве счетчика кодируемого символа чаще всего используется значение счетчика, означающего сколько раз данный символ встречался во входном потоке данных, либо в данном контексте.

В классическом варианте алгоритма для первого символа в качестве рабочего интервала [*l0,h0*) берется интервал [0, 1), где *li* и *hi* левая и правая границы рабочего интервала при кодировании *i*-го символа. В качестве следующего рабочего интервала берется диапазон внутри текущего интервала в распределении накопленных вероятностей кодируемого алфавита, соответствующий текущему кодируемому символу. Начало *li* каждого нового интервала сообщения определяется путем прибавления к началу предыдущего интервала произведения его ширины на накопленную вероятность в распределении частот для значения символа, предшествующего текущему *li=li-1+b[si-1]\*(hi-1-li-1*+1), где *b[si*-1] накопленная вероятность появления символа со значением *si*-1 в распределении накопленных вероятностей *P{0<S<n},* где *siєS*. Конец *hi* каждого нового интервала сообщения определяется путем прибавления к началу предыдущего интервала произведения его ширины на накопленную вероятность в распределении частот для значения текущего символа, *hi=li-1+b[si]\*(hi-1-li-1*+1), где *b[si*-1] вероятность появления символа со значением *si-*1 в распределении накопленных вероятностей *P{0<S<n*}, где *siєS*. При этом длина каждого нового рабочего интервала пропорциональна вероятности появления этого символа в потоке. На каждом шаге длина рабочего интервала уменьшается пропорционально вероятности текущего символа, а точка начала сдвигается вправо пропорционально началу диапазона для этого символа в распределении накопленных вероятностей. Новый построенный диапазон берется в качестве рабочего и т. д.

После кодирования последнего символа входной последовательности в качестве итогового значения берется любое значение входящее в последний полученный рабочий интервал. Полученное значение однозначно кодирует входную последовательность. Стоит отметить, что более вероятные символы сужают интервал в меньшей степени, чем менее вероятные, и, следовательно, добавляют меньшее количество разрядов к результату.

Классический алгоритм может сжимать только достаточно короткие цепочки из-за ограничений разрядности всех переменных. Чтобы избежать этих ограничений, реальный алгоритм работает с целыми числами и оперирует с дробями, числитель и знаменатель которых являются целыми числами (например, знаменатель равен 65536), и в качестве начального диапазона для первого кодируемого символа берется [0, 65535). При этом с целью минимизации потерь точности, отслеживается сближение *li* и *hi*, с последующим умножением их значений на какое-то число (чаще всего на 2) в случае сужения интервала до размеров меньших половины исходного интервала [0,65535). С целью борьбы с переполнением сверху, записываются старшие биты значений *li* и *hi* в файл тогда, когда они перестают меняться (т. е. не участвуют в дальнейшем уточнении интервала).

Минимизация потерь поточности достигается благодаря тому, что длина целочисленного интервала всегда не менее половины всего интервала. Когда *li* и *hi* одновременно находятся в верхней или нижней половине интервала, то записываются их одинаковые верхние биты в выходной поток, вдвое увеличивая интервал. Если *li* и *hi* приближаются к середине интервала, оставаясь по разные стороны от его середины, то также вдвое увеличивается интервал, биты при этом записываются «условно». «Условно» означает, что эти биты выводятся в выходной файл позднее, когда становится известно их значение. Процедура изменения значений *li* и *hi* называется нормализацией*,* а вывод соответствующих битов - переносом*.*

* + 1. Словарные методы сжатия

Входную последовательность символов можно рассматривать как последовательность строк, содержащих произвольное количество символов. Идея словарных методов состоит в замене строк символов на такие коды, что их можно трактовать как индексы строк некоторого словаря. Образующие словарь строки обычно называют фразами. При декодировании осуществляется обратная замена индекса на соответствующую ему фразу словаря. Словарные методы преобразуют исходную последовательность путем ее представления таким образом, что «буквы» в словарных методах являются фразами словаря, состоящими, в общем случае, из произвольного количества символов входной последовательности. Словарь - это набор таких фраз, которые, как предполагается, будут встречаться в обрабатываемой последовательности. Индексы фраз должны быть построены таким образом, чтобы в среднем их представление занимало меньше места, чем требуют замещаемые строки. За счет этого и происходит сжатие. LZ77 и LZ78 – алгоритмы сжатия без потерь, опубликованные в статьях Абрахама Лемпеля и Якоба Зива в 1977 и 1978 годах. Эти алгоритмы – самые известные варианты в семье LZ\*, которая также включает LZW, LZSS, LZMA и другие алгоритмы. Оба алгоритма – алгоритмы со словарным подходом [7]. LZ77 использует уже просмотренную часть сообщения как словарь. Алгоритм LZ77 использует «скользящее окно», которое эквивалентно неявному использованию словарного подхода, сначала предложенного в LZ78. «Скользящее окно» разделено на две неравных части. Первая часть, большая по размеру, включает уже просмотренную часть сообщения. Вторая, намного меньшая, – буфер, содержащий еще не закодированные символы. Как правило, размер окна – несколько килобайт. Буфер намного меньше, обычно не более, чем сто байт.

Алгоритм пытается найти фрагмент в словаре, который совпадает с содержанием буфера и закодировать указатель на этот фрагмент. Словарные методы хорошо себя зарекомендовали при сжатии текстов и в качестве алгоритмов, используемых в универсальных архиваторах, таких как 7z и WinRar.

* + 1. Фрактальный алгоритм

Фрактальная архивация основана на представлении изображения в более компактной форме — с помощью коэффициентов системы итерируемых функций (Iterated Function System — далее как IFS).

IFS представляет собой набор трехмерных аффинных преобразований, переводящих одно изображение в другое. Преобразованию подвергаются точки в трехмерном пространстве (х\_координата, у\_координата, яркость).

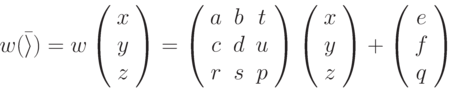
Фрактальный алгоритм бащируется на следующих определениях и теоремах:

Определение 1. Преобразование w: R2 🡪 R2, представимое в виде

w(\bar \rangle ) = w 
\left(
  \begin{array}{c}
    x \\
    y \\
  \end{array}
\right)
=
\left(
  \begin{array}{cc}
    a & b \\
    c & b \\
  \end{array}
\right)
\left(
  \begin{array}{c}
    x \\
    y \\
  \end{array}
\right)
+
\left(
  \begin{array}{c}
    e \\
    f \\
  \end{array}
\right)


где a, b, c, d, e, f действительные числа и (xy) ∈ R2 называется двухмерным аффинным преобразованием.

Определение 2. Преобразование w: R3 🡪 R3, представимое в виде



где a, b, c, d, e, f, p, q, r, s, t, u действительные числа и (xyz) ∈ R3 называется трехмерным аффинным преобразованием.

* + 1. Алгоритм JPEG

JPEG — один из самых новых и достаточно мощных алгоритмов. Практически он является стандартом для полноцветных изображений [7]. Оперирует алгоритм областями 8х8, на которых яркость и цвет меняются сравнительно плавно. Вследствие этого, при разложении матрицы такой области в двойной ряд по косинусам (см. формулы ниже) значимыми оказываются только первые коэффициенты. Таким образом, сжатие в JPEG осуществляется за счет плавности изменения цветов в изображении.

Алгоритм разработан группой экспертов в области фотографии специально для сжатия 24-битных изображений. JPEG — Joint Photographic Expert Group — подразделение в рамках ISO — Международной организации по стандартизации. В целом алгоритм основан на дискретном косинусоидальном преобразовании (в дальнейшем ДКП), применяемом к матрице изображения для получения некоторой новой матрицы коэффициентов. Для получения исходного изображения применяется обратное преобразование.

ДКП раскладывает изображение по амплитудам некоторых частот. Таким образом, при преобразовании мы получаем матрицу, в которой многие коэффициенты либо близки, либо равны нулю. Кроме того, благодаря несовершенству человеческого зрения, можно аппроксимировать коэффициенты более грубо без заметной потери качества изображения.

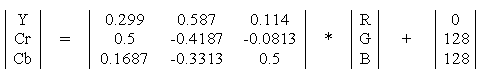
Для этого используется квантование коэффициентов. В самом простом случае — это арифметический побитовый сдвиг вправо. При этом преобразовании теряется часть информации, но могут достигаться большие коэффициенты сжатия.

Работа алгоритма:

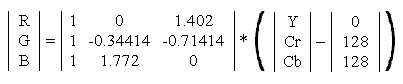
Пусть мы сжимаем 24-битное изображение.

Шаг 1. Перевод изображения из цветового пространства RGB, с компонентами, отвечающими за красную, зеленую и синюю составляющие цвета точки, в цветовое пространство YCrCb. В нем Y – яркостная составляющая, а Cr, Cb — компоненты, отвечающие за цвет (хроматический красный и хроматический синий). За счет того, что человеческий глаз менее чувствителен к цвету, чем к яркости, появляется возможность архивировать массивы для Cr и Cb компонент с большими потерями и, соответственно, большими степенями сжатия. Подобное преобразование уже давно используется в телевидении. На сигналы, отвечающие за цвет, там выделяется более узкая полоса частот.

Упрощенно перевод из цветового пространства RGB в цветовое пространство YCrCb можно представить с помощью матрицы перехода:

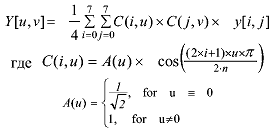


Обратное преобразование осуществляется умножением вектора YUV на обратную матрицу:



Шаг 2. Исходное изображение разбивается на матрицы 8x8. Формируется из каждой три рабочие матрицы ДКП – по 8 бит отдельно для каждой компоненты. Изображение делится по компоненте Y – как и в первом случае, а для компонент Cr и Cb матрицы набираются через строчку и через столбец. Т.е. из исходной матрицы размером 16x16 получается только одна рабочая матрица ДКП. При этом, как нетрудно заметить, мы теряем 3/4 полезной информации о цветовых составляющих изображения и получаем сразу сжатие в два раза. Мы можем поступать так благодаря работе в пространстве YCrCb.

Шаг 3. При n = 8 ДКП можно представить следующим образом:



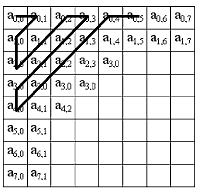
ДКП применяется к каждой рабочей матрице. При этом получается матрица, в которой коэффициенты в левом верхнем углу соответствуют низкочастотной составляющей изображения, а в правом нижнем – высокочастотной. Понятие частоты следует из рассмотрения изображения как двумерного сигнала (аналогично рассмотрению звука как сигнала). Плавное изменение цвета соответствует низкочастотной составляющей, а резкие скачки – высокочастотной.

Шаг 4. Производится квантование. Для каждой компоненты (Y, U и V), в общем случае, задается своя матрица квантования q[u,v] (далее МК).



На этом шаге осуществляется управление степенью сжатия, и происходят самые большие потери.

Шаг 5. Матрица 8x8 переводится в 64-элементный вектор следующим образом:



Таким образом, в начале вектора получаются коэффициенты матрицы, соответствующие низким частотам, а в конце — высоким.

Шаг 6. Вектор свертывается с помощью алгоритма группового кодирования. При этом получаются пары типа (Q, V), где “Q” является счетчиком пропускаемых нулей, а “V” – значение, которое необходимо поставить в следующую ячейку. Так, вектор 42 3000-2 00001... будет свернут в пары (0,42) (0,3) (3,-2) (4,1)...

Шаг 7. Получившиеся пары свертываем кодированием по Хаффману с фиксированной таблицей.

Процесс восстановления изображения в этом алгоритме полностью симметричен. Метод позволяет сжимать некоторые изображения в 10-15 раз без серьезных потерь.

Конвейер операций, используемый в алгоритме JPEG:



Подводя итог всему вышесказанному, можно составить следующие таблицы

|  |  |
| --- | --- |
| Алгоритм | Особенности изображения, за счет которых происходит сжатие |
| RLE | Подряд идущие одинаковые цвета: 2 2 2 2 2 2 15 15 15 |
| LZW | Одинаковые подцепочки: 2 3 15 40 2 3 15 40 |
| Хаффмана | Разная частота появления цвета: 2 2 3 2 2 4 3 2 2 2 4 |
| CCITT-3 | Преобладание белого цвета в изображении, большие области, заполненные одним цветом |
| Рекурсивный | Плавные переходы цветов и отсутствие резких границ |
| JPEG | Отсутствие резких границ |
| Фрактальный | Подобие между элементами изображения |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Алгоритм | К-ты сжатия | Симметричность по времени | На что  ориентирован | Потери | Размерность |
| RLE | 32, 2, 0.5 | 1 | 3,4-х битные | Нет | 1D |
| LZW | 1000, 4, 5/7 | 1.2-3 | 1-8 битные | Нет | 1D |
| Хаффмана | 8, 1.5, 1 | 1-1.5 | 8 битные | Нет | 1D |
| CCITT-3 | 213(3), 5, 0.25 | ~1 | 1-битные | Нет | 1D |
| JBIG | 2-30 раз | ~1 | 1-битные | Нет | 2D |
| Lossless JPEG | 2 раза | ~1 | 24-битные, серые | Нет | 2D |
| JPEG | 2-20 раз | ~1 | 24-битные, серые | Да | 2D |
| Рекурсивное сжатие | 2-200 раз | 1.5 | 24-битные, серые | Да | 2D |

* 1. Форматы для сжатия изображений
     1. WinRar

WinRAR–архиватор файлов в форматы RAR и ZIP для 32-и 64-разрядных операционных систем Windows (также существуют версии этого архиватора дляAndroid, Linux, FreeBSD, MacOSX, MS-DOS, WindowsMobile). Считается одним из лучших архиваторов по соотношению степени сжатия к скорости работы. RAR использует собственный алгоритм, основанный на словарном методе LZSS. Данный формат является универсальным для любых видов данных и не является специализированным для сжатия изображений, но,в виду своей широкой распространенности, очень часто используется для сжатий изображений в том числе.

* + 1. PNG

Переносимый сетевой графический формат (Portable Network Graphics, PNG) разрабатывался как более эффективная, гибкая и свободная от патентов замена GIF-формату. PNG был задуман для хранения отдельных растровых изображений для дальнейшего распространения по компьютерным сетям. PNG был создан в 1995 в ответ на давление со стороны Unisys и их патента на алгоритм LZW-сжатия, используемый в GIF. PNG использует алгоритм deflate-сжатия обычно с 32Кб скользящим (sliding) окном. Deflateявляется улучшенной версией алгоритма сжатия Lempel-Ziv(LZ77), который используется в ZIP-и GZIP-файлах (3, 4). Созданный Phil Katzдля второй версии PKZip, deflate совмещает LZ77 с кодированием Huffmanи является от 10% до 30% более эффективным, чем LZWпри сжатии без потери информации. Изображения, закодированные с помощью алгоритма PNG, обычно от 10% до 30% меньше, чем GIF

/\*найди и распиши один-два алгоритма из представленных выше, чтоб добавить к предыдущей главу \*/

* + 1. TIFF

TIFF (Tagged Image File Format) –формат хранения растровых графических изображений. TIFF стал популярным форматом для хранения изображений с большой глубиной цвета. Он используется при сканировании, распознавании текста, в полиграфии, широко поддерживается графическими приложениями. Изначально формат поддерживал только сжатие без потерь, впоследствии формат был дополнен для поддержки сжатия с потерями в формате JPEG.Файлы формата TIFF, как правило, имеют расширение .tiff или .tif. Имеется возможность сохранять изображение в файле формата TIFF со сжатием и без сжатия. Степени сжатия зависят от особенностей самого сохраняемого изображения, а также от используемого алгоритма. Формат TIFF позволяет использовать следующие алгоритмы сжатия: PackBits(RLE), Lempel-Ziv-Welch(LZW), LZ77, ZIP, JBIG, JPEG, CCITT Group 3, CCITT Group 4. При этом,JPEG является просто инкапсуляцией формата JPEG в формат TIFF. Формат TIFF позволяет хранить изображения, сжатые по стандарту JPEG.Алгоритмы CCITT Group 3 и 4, используемые в Tiff,предназначены для кодирования бинарных растровых изображений. Первоначально они были разработаны для сетей факсимильной связи (поэтому иногда их называют Fax 3, Fax 4). В настоящий момент они также используются в полиграфии, системах цифровой картографии и географических информационных системах.

* 1. Математическая модель

//не знаю

* 1. Постановка задачи
     1. Входная информация

На вход программе подаётся N монохромных растровых изображений с одинаковым разрешением.

N принимает значение от 2 до …

* + 1. Выходная информация

На выходе программа создаёт N+3 файла:

Result.txt – файл, содержащий полученную ковариационную матрицу

Vectors.txt – файл, содержащий собственные вектора ковариационной матрицы, полученные после применения метода Якоби

Values.txt – файл, содержащий собственные числа ковариационной матрицы, полученные после применения метода Якоби

1. Практическая часть
   1. Описание разработанных и используемых алгоритмов
      1. Алгоритм ColumnEze

Данный алгоритм получает на вход монохромное растровое изображение, представленное последовательностью пикселов, и обрабатывает его следующим образом:

1. Для каждого i-ого пиксела с координатами [X,Y] происходит сравнение значений цветности, т.е. сравнивается значение красного цвета с зелёным, а так же зелёного с синим. В случае если сравниваемые значения одинаковы – алгоритм продолжает свою работу. В противном случае, будет создано новое диалоговое окно со следующим сообщением: “Изображение не монохромно, целью были моего создания была работа с монохромными изображениями”. После чего, обработка изображений будет прервана.
2. После того, как пройдена проверка на монохромность i-ого пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
3. Далее берётся пиксел с координатами [X+1,Y] и алгоритм продолжает свою работу с пункта 1, до тех пора, пока X координата не примет значение Height, где Height – высота изображения в пикселах, а Y координата принимает значение 0.
4. Как только X координата достигнет значения Height, обработка продолжится с пункта 1, в котором будет взят пиксел, имеющий координаты [X,Y+1], где X примет значение 0.
5. При достижении координатой X значения Height, а координатой Y значения Width алгоритм записывает полученную последовательность байт в файл “ColumnEzeN.bin”.

Графически данный алгоритм можно представить следующим образом:

|  |  |  |  |
| --- | --- | --- | --- |
| a00 | a01 | … | a0W |
| a10 | a11 | … | a1W |
| … | … | … | … |
| aH0 | aH1 | … | aHW |

Данная матрица А с пикселями aij, где i = 0.. Height, j = 0... Width.

* + 1. Алгоритм StringEze

Данный алгоритм получает на вход монохромное растровое изображение, представленное последовательностью пикселов, и обрабатывает его следующим образом:

1. Для каждого i-ого пиксела с координатами [X,Y] происходит сравнение значений цветности, т.е. сравнивается значение красного цвета с зелёным, а так же зелёного с синим. В случае если сравниваемые значения одинаковы – алгоритм продолжает свою работу. В противном случае, будет создано новое диалоговое окно со следующим сообщением: “Изображение не монохромно, целью были моего создания была работа с монохромными изображениями”. После чего, обработка изображений будет прервана.
2. После того, как пройдена проверка на монохромность i-ого пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
3. Далее берётся пиксел с координатами [X,Y+1] и алгоритм продолжает свою работу с пункта 1, до тех пора, пока Y координата не примет значение Width, где Width – ширина изображения в пикселах.
4. Как только Y координата достигнет значения Width, обработка продолжится с пункта 1, в котором будет взят пиксел, имеющий координаты [X+1,Y], где Y принимает значение 0.
5. При достижении координатой X значения Height, а координатой Y значения Width, алгоритм записывает полученную последовательность байт в файл “StringEzeN.bin
   * 1. Алгоритм ColumnZmey

Данный алгоритм получает на вход монохромное растровое изображение, представленное последовательностью пикселов, и обрабатывает его следующим образом:

1. Для каждого i-ого пиксела с координатами [X,Y] происходит сравнение значений цветности, т.е. сравнивается значение красного цвета с зелёным, а так же зелёного с синим. В случае если сравниваемые значения одинаковы – алгоритм продолжает свою работу. В противном случае, будет создано новое диалоговое окно со следующим сообщением: “Изображение не монохромно, целью были моего создания была работа с монохромными изображениями”. После чего, обработка изображений будет прервана.
2. После того, как пройдена проверка на монохромность i-ого пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
3. Далее берётся пиксел с координатами [X+1,Y] и алгоритм продолжает свою работу с пункта 1, до тех пора, пока X координата не примет значение Height, где Height – высота изображения в пикселах.
4. Когда координата X принимает значение Height, алгоритм продолжает свою работу с пикселом, имеющим координаты [X,Y+1].
5. Для выбранного пиксела с новыми координатами [X,Y] происходит сравнение значений цветности, т.е. сравнивается значение красного цвета с зелёным, а так же зелёного с синим. В случае если сравниваемые значения одинаковы – алгоритм продолжает свою работу. В противном случае, будет создано новое диалоговое окно со следующим сообщением: “Изображение не монохромно, целью были моего создания была работа с монохромными изображениями”. После чего, обработка изображений будет прервана.
6. После того, как пройдена проверка на монохромность пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
7. Далее берётся пиксел с координатами [X-1,Y] и алгоритм продолжает свою работу с пункта 5, до тех пора, пока X координата не примет значение 0.
8. После того, как X координата примет значение 0, алгоритм продолжит свою работу с пункта 1, при этом координатами нового пиксела будут [X,Y+1].
9. При достижении координатой X значения Height, а координатой Y значения Width, алгоритм записывает полученную последовательность байт в файл “ ColumnZmey.bin”
   * 1. Алгоритм StringZmey

Данный алгоритм получает на вход монохромное растровое изображение, представленное последовательностью пикселов, и обрабатывает его следующим образом:

1. Для каждого i-ого пиксела с координатами [X,Y происходит сравнение значений цветности, т.е. сравнивается значение красного цвета с зелёным, а так же зелёного с синим. В случае если сравниваемые значения одинаковы – алгоритм продолжает свою работу. В противном случае, будет создано новое диалоговое окно со следующим сообщением: “Изображение не монохромно, целью были моего создания была работа с монохромными изображениями”. После чего, обработка изображений будет прервана.
2. После того, как пройдена проверка на монохромность i-ого пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
3. Далее берётся пиксел с координатами [X,Y+1] и алгоритм продолжает свою работу с пункта 1, до тех пора, пока Y координата не примет значение Width, где Width –ширина изображения в пикселах.
4. Когда координата Y принимает значение Width, алгоритм продолжает свою работу с пикселом, имеющим координаты [X+1,Y].
5. Для выбранного пиксела с новыми координатами [X,Y] происходит сравнение значений цветности, т.е. сравнивается значение красного цвета с зелёным, а так же зелёного с синим. В случае если сравниваемые значения одинаковы – алгоритм продолжает свою работу. В противном случае, будет создано новое диалоговое окно со следующим сообщением: “Изображение не монохромно, целью были моего создания была работа с монохромными изображениями”. После чего, обработка изображений будет прервана.
6. После того, как пройдена проверка на монохромность пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
7. Далее берётся пиксел с координатами [X,Y-1] и алгоритм продолжает свою работу с пункта 5, до тех пора, пока Y координата не примет значение 0.
8. После того, как Y координата примет значение 0, алгоритм продолжит свою работу с пункта 1, при этом координатами нового пиксела будут [X+1,Y].
9. При достижении координатой X значения Height, а координатой Y значения Width, алгоритм записывает полученную последовательность байт в файл “ StringZmey.bin”
   * 1. Алгоритм GetCovarMatrix

Алгоритм получает на вход

* + 1. Алгоритм RotationMethod //скелет

Положить k=0, A(0)=A и задать ε>0

Выделить в верхней треугольной наддиагональной части матрицы A(k) максимальный по модулю элемент a(k)ij, i<j.

Если |a(k)ij|⩽ε для всех i≠j, процесс завершить. Собственные значения определяются по формуле λi(A(k))=a(k)ii, i=1,…,n

Собственные векторы Xi находятся как i-e столбцы матрицы, получающейся в результате перемножения:

νk=H(0)\*H(1)\*H(2)\*…\*H(k−1) = (X1,X2,X3,…,Xn).

Если ∣∣a(k)ij∣∣>ε процесс продолжается.

Найти угол поворота по формуле φ(k)= \* arctg.

Составить матрицу вращения H(k)

Вычислить очередное приближение A(k+1) = (H(k))T \* A(k)\* H(k).Положить k=k+1 и перейти к пункту 2.

* + 1. Алгоритм Multiplication
    2. Алгоритм FromImageToTxt
    3. Алгоритм FromTxtToTB

1. Информационное обеспечение системы
2. Условие разработки и функционирования ПО

Среда разработки - Microsoft Visual Studio 2017.

Язык разработки – C#.

Требования к квалификации пользователя - оператор ЭВМ.

Минимальные технические требования для работы с программным продуктом:

1. ПК типа Asus K53SV
2. ОС – не ниже Windows 7 SP1
3. Видеоадаптер Nvida 560m
4. Разрешение экрана 1366x768
5. Процессор - не ниже 2.70GHz
6. Оперативная память - не ниже 4 Гб;
7. Место на жестком диске (HDD) - не ниже 4 Гб;
8. Периферийные устройства ввода информации: клавиатура, мышь или тачпад

Дополнительные требования: платформа.NET Framework 4.5 или более новые версии.

Объем программного кода – 1000 строк

Разработано классов - 3

Примерное время разработки - 5 месяцев.

1. Детальное описание разрабатываемого ПО
   1. Назначение
   2. Входные и выходные данные

Входные параметры:

На вход программе подаются монохромные гиперспектральные растровые изображения.

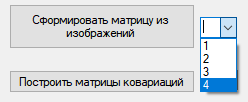
Варьируемые параметры:

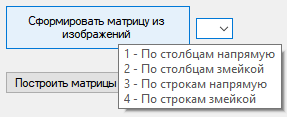
* Eps – погрешность
* Способ формирования вектора из пиеселей изображения

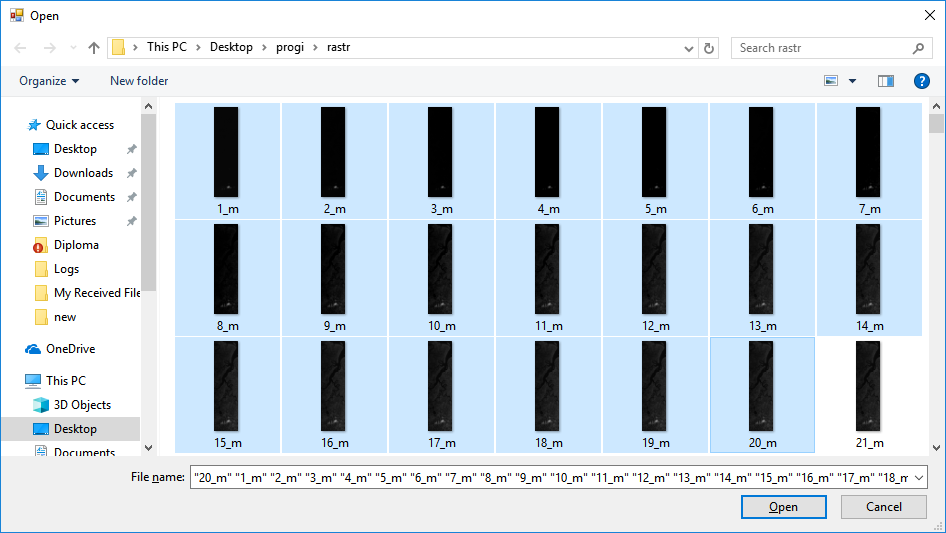
Ограничения:

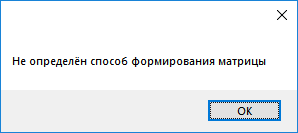
* countImages <= 1000, где countImages – количество обрабатываемых изображений
* Images[i].Width = Images[i+1].Width, где Images – массив изображений, i – номер изображения, i = 1.. countImages., Width – ширина изображения
* Images[i].Height = Images[i+1].Height, где Height – высота изображения
  1. Руководство оператора
* Для запуска программы дважды нажмите левой кнопкой мыши(ЛКМ) на её ярлык. Программа выведет главное окно

/\*скрин\*/

* Определите способ формирования матрицы из изображений, выбрав соответствующую цифру в списке:
* 
* В случае затруднения, обратитесь к подсказке, наведя курсор мыши на кнопку . Вам будет показана следующая подсказка:



* Определив способ формирования матрицы из изображений, нажминте на кнопку  и выберите изображения для обработки:
* 
* В случае, если не был определён способ формирования матрицы, будет выведено соответствующее диалоговое окно:

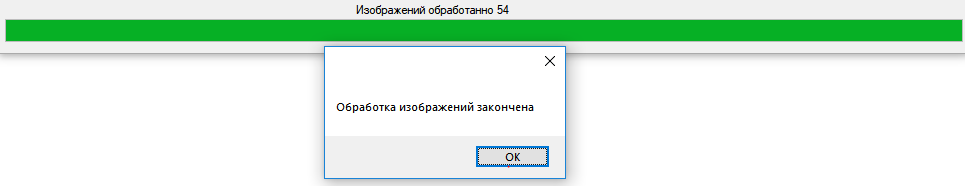


* После успешного выбора изображений и нажатия кнопки “Open”,начнётся процесс формирования матрицы из выбранных изображений, который будет сопровождаться визуальной индикацией хода процесса:





* По завершении процесса формирования матрицы, будет показано диалоговое окно, а на главном окне отобразится количество обработанных изображений:



* Обработанные изображения будут помещены в корневую директорию данного программного обеспечения, матрица, сформированная из выбранных ранее изображений будет содержаться в файле All.bin

1. Список литературы
2. А.В. Вицентий «Применение дистанционного зондирования земли и космических технологий для развития арктических и субарктических территорий Российской Федерации»
3. Демидова Л.А., Еремеев В.В., Мятов Г.Н., Тишкин Р.В., Юдаков А.А.«Сегментация объектов по данным гиперспектральной съемки земли с использованием методов искусственного интеллекта»
4. ЮдаковА.А, «Алгоритмы сегментации объектов земной поверхности по данным гиперспектральной съемки»
5. Катермоул, К.В. Принципы импульсно-кодовой модуляции/ К.В.Катермоул; пер. с англ. под ред. В.В. Маркова. – М.: Связь, 1974.
6. Ватолин, Д. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео/ Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юркин. – М.: ДИАЛОГ – МИФИ, 2003.
7. Колесник, В.Д. Курс теории информации/ В.Д. Колесник, Г.Ш. Полтырев. – М.: Наука, 1982.
8. Васин, Ю.Г. Оптимизация описания исходных данных в диалоговых системахрешения задач классификации/ Ю.Г. Васин // В кн. "Современное состояние теории исследования операций". – М.: Наука, 1979. –С.424-450.
9. Le Gall D. J. The MPEG Video Compression Algorithm // Signal Processing: Image Communication. 1992. Vol. 4, № 2. P. 129 140.