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:

доцент кафедры ИАНИ, к.т.н. Васин Д.Ю.

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

Рецензент:

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

Нижний Новгород

2019

# Оглавление

[Оглавление 2](#_Toc9074101)

[Введение 3](#_Toc9074102)

[Актуальность темы 3](#_Toc9074103)

[Цель работы 3](#_Toc9074104)

[Объект исследования 3](#_Toc9074105)

[Обзор задач, которые предстоит решить 4](#_Toc9074106)

[Обзор существующих методов решения 5](#_Toc9074107)

[Статистические методы сжатия 5](#_Toc9074108)

[Словарные методы сжатия 6](#_Toc9074109)

[Фрактальный алгоритм 7](#_Toc9074110)

[Алгоритм JPEG 7](#_Toc9074111)

[Список литературы 8](#_Toc9074112)

# Введение

## Актуальность темы

Дистанционное зондирование - это способы 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 и т. 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, краткий обзор которых будет представлен ниже в данной работе.

## Цель работы

Целью данной работы является создание информационной системы, способной кодировать и сжимать гиперспектральные данные ДЗЗ при помощи метода главных компонент.

## Объект исследования

Объектом исследования являются гиперспектральные растровые изображения.

## Обзор задач, которые предстоит решить

1. Обзор существующих методов достижения данной цели.
2. Сравнение метода главных компонент с иными существующими метода кодирования гиперспектральных растровых изображений.

# Обзор существующих методов решения

## 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.

/\* переработай нижнее \*/

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*.*

## 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 7zи WinRar.

## Фрактальный алгоритм

## Алгоритм JPEG

# Форматы для сжатия изображений

## 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.

## 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 \*/

## 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.

*/\* переработай верхнее \*/*

# Математическая модель

//не знаю

# Постановка задачи

## Входная информация

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

N принимает значение от 2 до …

## 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

# Описание разработанных и используемых алгоритмов

## Алгоритм ColumnEze

Данный алгоритм получает на вход монохромные растровые изображения, представленные последовательностью пикселов, и обрабатывает их следующим образом:

1. Для каждого i-ого пиксела с координатами [X,Y] в K-ом изображении 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.

1. После того, как пройдена проверка на монохромность i-ого пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
2. 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.
3. 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.
4. При достижении координатой X значения Height, координатой Y значения Width для K-ого изображения из входной последовательности пикселов, алгоритм записывает полученную последовательность байт в файл “ColumnEzeN.txt” и продолжает свою работу с пункта 1, где вместо K-ого изображения берётся K+1-ое изображение, где Width – ширина изображения в пикселах,N в имени файла – номер обработанного изображения.
5. По достижении K значения Count, где Count – количество изображений в полученных входных данных, алгоритм завершает свою работу, создав новое диалоговое окно с сообщением “DONE”.

## Алгоритм StringEze

Данный алгоритм получает на вход монохромные растровые изображения, представленные последовательностью пикселов, и обрабатывает их следующим образом:

1. Для каждого i-ого пиксела с координатами [X,Y] в K-ом изображении 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.

1. После того, как пройдена проверка на монохромность i-ого пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
2. 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.
3. 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.
4. При достижении координатой X значения Height, координатой Y значения Width для K-ого изображения из входной последовательности пикселов, алгоритм записывает полученную последовательность байт в файл “StringEzeN.txt” и продолжает свою работу с пункта 1, где вместо K-ого изображения берётся K+1-ое изображение, где Width – ширина изображения в пикселах,N в имени файла – номер обработанного изображения.
5. По достижении K значения Count, где Count – количество изображений в полученных входных данных алгоритм завершает свою работу, создавая новое диалоговое окно с сообщением “DONE”.

## Алгоритм ColumnZmey

Данный алгоритм получает на вход монохромные растровые изображения, представленные последовательностью пикселов, и обрабатывает их следующим образом:

1. Для каждого i-ого пиксела с координатами [X,Y] в K-ом изображении 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.

1. После того, как пройдена проверка на монохромность i-ого пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
2. 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.
3. Когда координата X принимает значение Height, алгоритм продолжает свою работу с пикселом, имеющим координаты [X,Y+1].
4. 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.

1. После того, как пройдена проверка на монохромность пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
2. 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.
3. После того, как X координата примет значение 0, алгоритм продолжит свою работу с пункта 1, при этом координатами нового пиксела будут [X,Y+1].
4. По достижении K значения Count, где Count – количество изображений в полученных входных данных, алгоритм завершает свою работу, создав новое диалоговое окно с сообщением “DONE”.

## Алгоритм StringZmey

Данный алгоритм получает на вход монохромные растровые изображения, представленные последовательностью пикселов, и обрабатывает их следующим образом:

1. Для каждого i-ого пиксела с координатами [X,Y] в K-ом изображении 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.

1. После того, как пройдена проверка на монохромность i-ого пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
2. 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.
3. Когда координата Y принимает значение Width, алгоритм продолжает свою работу с пикселом, имеющим координаты [X+1,Y].
4. 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.

1. После того, как пройдена проверка на монохромность пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
2. 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.
3. После того, как Y координата примет значение 0, алгоритм продолжит свою работу с пункта 1, при этом координатами нового пиксела будут [X+1,Y].
4. По достижении K значения Count, где Count – количество изображений в полученных входных данных, алгоритм завершает свою работу, создав новое диалоговое окно с сообщением “DONE”.

## Алгоритм GetCovarMatrix

## Алгоритм RotationMethod //скелет

1. Положить k=0, A(0)=A и задать ε>0
2. 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.

1. Найти угол поворота по формуле φ(k)= \* arctg.
2. Составить матрицу вращения H(k)
3. Вычислить очередное приближение A(k+1) = (H(k))T \* A(k)\* H(k).Положить k=k+1 и перейти к пункту 2.

## Алгоритм Multiplication

## Алгоритм FromImageToTxt

## Алгоритм FromTxtToTB

# Список литературы

1. А.В. Вицентий «Применение дистанционного зондирования земли и космических технологий для развития арктических и субарктических территорий Российской Федерации»
2. 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»
3. ЮдаковА.А, «Алгоритмы сегментации объектов земной поверхности по данным гиперспектральной съемки»
4. Катермоул, К.В. Принципы импульсно-кодовой модуляции/ К.В.Катермоул; пер. с англ. под ред. В.В. Маркова. – М.: Связь, 1974.
5. Ватолин, Д. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео/ Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юркин. – М.: ДИАЛОГ – МИФИ, 2003.
6. Колесник, В.Д. Курс теории информации/ В.Д. Колесник, Г.Ш. Полтырев. – М.: Наука, 1982.
7. 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.