МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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**

**Центр прикладных информационных технологий**

Направление подготовки: «Прикладная информатика»

Профиль подготовки: «Прикладная информатика в информационной сфере»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**Тема:**

**«Кодирование гиперспектральных 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](#_Toc10471860)

[Введение 4](#_Toc10471861)

[Актуальность темы 4](#_Toc10471862)

[1 Теоретическая часть 9](#_Toc10471863)

[1.1 Гиперспектральные данные ДЗЗ 9](#_Toc10471864)

[1.1.1 Основные пробемы обработки ГСИ 10](#_Toc10471865)

[1.2 Обзор существующих методов решения данной задачи 12](#_Toc10471866)

[1.2.1 Статистические методы сжатия 12](#_Toc10471867)

[1.2.2 Лексические методы сжатия 14](#_Toc10471868)

[1.2.3 Алгоритм RLE 15](#_Toc10471869)

[1.2.4 Алгоритм LZW 15](#_Toc10471870)

[1.2.5 Фрактальный алгоритм 16](#_Toc10471871)

[1.3 Метод главных компонент 18](#_Toc10471872)

[2 Постановка задачи 20](#_Toc10471873)

[3 Практическая часть 21](#_Toc10471874)

[3.1 Описание разработанных алгоритмов 21](#_Toc10471875)

[3.1.1 Алгоритм формирования матрицы изображений №1 21](#_Toc10471876)

[3.1.2 Алгоритм формирования матрицы изображений №2 22](#_Toc10471877)

[3.1.3 Алгоритм формирования матрицы изображений №3 22](#_Toc10471878)

[3.1.4 Алгоритм формирования матрицы изображений №4 24](#_Toc10471879)

[3.1.5 Алгоритм метода вращения 25](#_Toc10471880)

[3.1.6 Алгоритм умножения матриц 25](#_Toc10471881)

[3.1.7 Алгоритм нахождения скалярного произведения векторов 26](#_Toc10471882)

[3.2 Условие разработки и функционирования ПО 26](#_Toc10471883)

[4 Описание разработанного СПО 27](#_Toc10471884)

[4.1 Класс Form 27](#_Toc10471885)

[4.1.1 Назначение класса Form1 27](#_Toc10471886)

[4.1.2 Описание полей и методов 27](#_Toc10471887)

[4.2 Класс Rotation 29](#_Toc10471888)

[4.2.1 Назначение класса Rotation 29](#_Toc10471889)

[4.2.2 Описание полей и методов 29](#_Toc10471890)

[4.3 Класс ColumgAndString 31](#_Toc10471891)

[4.3.1 Назначение класса 31](#_Toc10471892)

[4.3.2 Описание полей и методов 32](#_Toc10471893)

[4.4 Описание разработанного модуля 36](#_Toc10471894)

[4.5 Руководство оператора 36](#_Toc10471895)

[4.6 Результаты практических экспериментов 39](#_Toc10471896)

[5 Выводы 40](#_Toc10471897)

[6 Список литературы 41](#_Toc10471898)

[7 Приложение 43](#_Toc10471899)

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 и т.д. Среди них особое место занимают методы дистанционного зондирования Земли (ДЗЗ) из космоса.[1]

Географы в своих исследованиях используют не только методоы отраслевых географических наук но и общегеографические методы — описательный, математический, картографический, аэрокосмический.

Под аэрокосмическими методами принято понимать совокупность методов исследований атмосферы, земной поверхности, океанов, верхнего слоя земной коры с воздушных и космических носителей путем дистанционной регистрации и последующего анализа идущего от Земли электромагнитного излучения. Аэрокосмические методы обеспечивают определение точного географического положения изучаемых объектов или явлений, получение их качественных и количественных биогеофизических характеристик и их изменений во времени. В географических исследованиях главную роль играют методы, основанные на регистрации параметров излучения в виде двумерного изображения, снимка. Использование аэрокосмических снимков не только упрощает изучение труднодоступных территорий, но и обеспечивает географа дистанционной геопространственной информацией, которую другими способами получить не удается.[11]

Снимки с самолетов — одиночные фотографии небольших участков местности — появились в начале XX в., перед Первой мировой войной. Сейчас аэрокосмические снимки можно получать днем и ночью, используя для этого не только световое, но и невидимое излучение инфракрасного и радиодиапазона. Работают спутниковые системы разного назначения, выполняя глобальные съемки и передавая в считанные дни и часы тысячи снимков.

В связи с динамичным развитием мирового космического сообщества, ожидается, что уже в первые десятилетия XXI в. мировая группировка превысит тысячу космических аппаратов, среди которых число спутников, предназначенных для аэрокосмического зондирования Земли, приблизится к 20%. Возрастающей популярности аэрокосмических снимков способствует появление в Интернете поисковой системы Google Earth (http:maps. google. com).[11]

В результате обработки первичных аэрокосмических снимков с широким применением компьютерных технологий создаются разнообразные геоинформационные продукты — топографические и тематические карты, фотокарты, цифровые модели местности и др. Важное значение придается аэрокосмической видеоинформации при создании национальной инфраструктуры пространственных данных.[11]

Аэрокосмический снимок — это прежде всего информационная модель изучаемого объекта или явления. Снимки, имеющие множество разновидностей, несут разнообразную информацию о географических объектах, об их пространственном распределении, взаимосвязях, состоянии, а так же изменении во времени. Для результативного использования таких снимков исследователь должен знать их свойства и владеть специальными способами и приемами эффективного извлечения из снимков необходимой информации. Знание информационных особенностей снимков в совокупности с правельным выбором компьютерных программ для их обработки представляют собой залог успешного получения требуемой географической информации аэрокосмическими методами. В современном облике аэрокосмического зондирования как самостоятельной дисциплины отчетливо проявляются следующие тенденции его поступательного развития:

• Космические снимки, оперативно размещаемые в Интернете, становятся наиболее востребованной видеоинформацией о местности не только для специалистов-профессионалов;

• Разрешение и метрические свойства космических снимков открытого доступа быстро повышаются. Получают распространение орбитальные снимки сверхвысокого разрешения — метрового и дециметрового;

• Аналоговые фотографические снимки и традиционные технологии их обработки утрачивают свое прежнее монопольное значение. Основным обрабатывающим прибором для исполнителя стал компьютер, оснащенный специализированным программным обеспечением и периферией;

• Прогрессивное развитие всепогодной радиолокации превращает ее в мощный метод получения метрически точной пространственной геоинформации, начинающий эффективно комплексироваться с оптическими технологиями аэрокосмического зондирования.

• Быстро формируется рынок разнообразной продукции аэрокосмического зондирования Земли. Число зарубежных коммерческих космических аппаратов, функционирующих на орбитах, неуклонно увеличивается, а получаемые ими снимки активно распространяются по всему миру.

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].

В настоящее время сложно найти передовую отрасль или направление деятельности, где не применялись бы технологии ДЗЗ. Ниже приведены основные области применения данных ДЗЗ.

*Сельское, лесное и охотничье хозяйство*. В данной области ДЗЗ применяют для различения типов вегетации и их состояния, оценки площадей посевов, лесных и охотничьих угодий по типам культур, определяют состояние почв и площади выгоревших участков.

*Геология*. Это одна из первых областей, при изучении которой активно использовалась съемка с воздушных шаров, самолетов и, впоследствии, с космических платформ. Наиболее часто данные ДЗЗ используют в этой области для различения типов пород, картирования больших геологических образований, обновления геологических карт и поиска указаний на определенные минералы.

*Картография и землепользование*. При решении различных задач землепользования с использованием данных ДЗЗ важнейшими являются классификация, картографирование и обновление карт, категоризация земель, разделение урбанизированных и сельских районов, региональное планирование, картирование транспортных сетей, картирование границ вода‑суша.

*Океанография и морские ресурсы*. При решении задач в этой области актуальными являются обнаружение живых морских организмов, исследование течений, картирование береговой линии, картирование отмелей и мелей, картирование льдов для целей судовождения, а также исследование морских волн.

*Водные ресурсы*. При исследовании водных ресурсов с использованием данных ДЗЗ чаще всего специалисты определяют границы водных объектов, их площади и объемы, исследуют мутность и турбулентность, проводят картирование областей затопления и границ снежного покрова, динамику их изменения.

*Окружающая среда*. Одна из наиболее актуальных для использования данных ДЗЗ является именно эта область. Вопросы безопасности и мониторинга окружающей среды стоят перед современным человечеством наиболее остро. Данные ДЗЗ активно используются для мониторинга разработок полезных ископаемых, картирования и мониторинга загрязнения поверхностных вод, обнаружения атмосферного загрязнения, определения последствий стихийных бедствий и чрезвычайных ситуаций, а также мониторинга воздействия человеческой активности на окружающую среду в целом.

Гиперспектральная съемка является новейшим и приоритетным направлением развития систем Д33[2]. Она предполагает одновременное получение сотен и тысяч изображений одной и той же сцены, зафиксированных в узких диапазонах спектра, которые соприкасаются друг с другом. 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 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.

К главным особенностям ГСИ относятся высокое спектральное разрешение (порядка нескольких нанометров) и большое количество спектральных каналов, что вызывает в свою очередь проблему размерности, из-за которой многие традиционные классификаторы становятся непригодными. Кроме того, попиксельная классификация ГСИ высокого пространственного разрешения часто приводит к получению шумовых картосхем, которые слабо поддаются интерпретации [13].

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.

Существующие математические методы обработки данных ДЗЗ не в полной мере соответствуют требованиям эффективности (по критериям качества и/или скорости) обработки и не покрывают весь спектр актуальных прикладных задач, решаемых на основе космической информации.[12]

Гигантский объём получаемых слабоструктурированных данных ДЗЗ (около 500 терабайт в год только на территории России) требует внедрения новых высокопроизводительных систем обработки информации на основе распределённых вычислений с применением суперкомпьютеров и облачных технологий, средств работы со сверхбольшими объёмами данных (big data), высокоскоростной среды обмена такими данными.[12]

Многочисленные станции приёма данных ДЗЗ не связаны в единую инфотелекоммуникационную и вычислительную сеть и слабо взаимодействуют, так же отсутствует эффективный обмен данными. Это является препятствием для научных исследований и прикладных инновационных проектов, требующих интеграции разнородной информации по большим территориям.[12]

**Целью данной работы** является разработка алгоритмического и программного обеспечения, кодирования гиперспектральных данных ДЗЗ по методу Главных компонент.

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

1. Рассмотреть основные подходы к сжатию растровых данных а также существующие методы сжатия без ошибки и с ошибкой
2. Изучить метод главных компонент;
3. Разработать алгоритмическое и программное обеспечение нахождения собственных векторов матрицы ковариации, полученной по набору ГСИ ДЗЗ

**Объектом исследования** являются гиперспектральные растровые изображения, полученные с гиперспектрометра Hyperion представленные в форме растровых изображений формата bmp.

1. Теоретическая часть
   1. Гиперспектральные данные ДЗЗ

Аэрокосмические снимки — результат аэрокосмических съемок, для выполнения которых используют разнообразные авиационные и космические носители. Аэрокосмические съемки делят на пассивные, которые предусматривают регистрацию отраженного солнечного или собственного излучения Земли, и активные, при которых выполняют регистрацию отраженного искусственного излучения.[16]

Аэрокосмический снимок — представляет собой двумерное изображение реальных объектов, которое получено по определенным геометрическим и радиометрическим (фотометрическим) законам путем дистанционной регистрации яркости объектов и предназначено для исследования видимых и скрытых объектов, явлений и процессов окружающего мира, а также для определения их пространственного расположения.[16]

Диапазон масштабов современных аэрокосмических снимков огромен: он может меняться от 1:1000 до 1:100000000, т.е. в сто тысяч раз. При этом наиболее распространенные масштабы аэрофотоснимков находятся в пределах 1:10000 — 1:50000, а космических — 1:200000 — 1:10000000. Все аэрокосмические снимки принято делить на аналоговые (обычно фотографические) и цифровые (электронные). Изображение цифровых снимков состоит из отдельных одинаковых пикселов, яркость каждого пиксела характеризуется одним числом. Аэрокосмический снимок состоит из миллионов пикселов. При выполнении практических работ приходится отличать исходные (первичные) снимки, которые получены непосредственно в результате съемки, от их копий и преобразованных снимков, поступающих к потребителям. Так, аналоговые снимки можно преобразовать в цифровые и наоборот. Наиболее распространено цифрование с помощью специального высокоточного сканера оригинальных негативов аэрофотографической съемки, предназначенных для компьютерной обработки.[16]

Аэрокосмические снимки как информационные модели местности характеризуются рядом следующим рядом свойств,:

Изобразительные - арактеризуют способность снимков воспроизводить мелкие детали, цвета и тоновые градации объектов;

Радиометрические (фотометрические) - свидетельствуют о точности количественной регистрации снимком яркостей объектов;

Геометрические - характеризуют возможность определения по снимкам размеров, длин и площадей объектов и их взаимного положения;

Свойства снимков, получаемых в разных диапазонах и различной съемочной аппаратурой, существенно различаются. Примером могут послужить следующие снимки одной и той же сцены, но в разных диапозонах (рис 1,2).



Рис 1

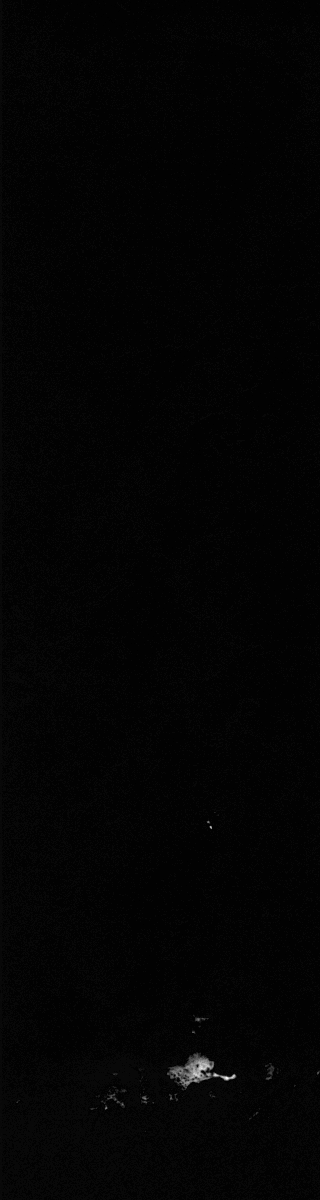


Рис 2

Кадровые снимки, отличающиеся наивысшей геометрической точностью, наиболее пригодны для точных измерений. Снимки, получаемые оптикоэлектронными съемочными системами, могут регистрировать больше энергетических уровней излучения, чем фотографические, и обладают наиболее высокой радиометрической точностью. Радиолокационные снимки можно получать в любую погоду, даже когда земная поверхность закрыта сплошным облачным покровом.

Важными показателями снимка служат охват и пространственное разрешение. Для географических исследований требуются снимки большого охвата и высокого пространственного разрешения. Но удовлетворить эти противоречивые требования в одном снимке не удается. Обычно чем больше охват получаемых снимков, тем ниже их пространственное разрешение. Поэтому при разработке съемочной аппаратуры приходится идти на компромиссные решения либо выполнять одновременно съемку несколькими системами с различными параметрами.[16]

* + 1. Основные пробемы обработки ГСИ

Смешанный пиксел

Гиперспектральная аппаратура работает в смежных узких спектральных каналах в пределах окон прозрачности атмосферы, что позволяет строить практически непрерывную кривую спектрального отражения для элементов подстилающей поверхности.[15]

Увеличение числа каналов даёт больший объём информации о сцене, но визуальное дешифрирование не позволяет извлечь все сведения, которые содержит куб данных. В связи с этим возникает необходимость автоматизированного спектрального анализа, основными методами которого являются классификация и обнаружение конкретных объектов.

Проблему спектральной селекции можно было бы решить проще, будь все пикселы изображения «чистыми», т.е. в случае, если каждый пиксел изоборажения содержал в себе только один объект или фон. Но на деле, во-первых, естественные поверхности редко состоят из однородного материала. Во-вторых, суммарное излучение от всех объектов в пределах элемента пространственного разрешения фиксируется аппаратурой как одиночный пиксел изображения. Таким образом возникает проблема смешанного пиксела.

Динамику смеси двух и более субстанций в пределах пиксела можно описать линейной и нелинейной моделью, которые представленны в [14].

Смешанные пикселы оказывают значительное влияние на дешифрирование изображений. Целочисленные методы классификации смешанных пикселов приводят к ошибкам в определении площадей, поскольку каждый пиксел может быть отнесён лишь к одному конкретному классу, несмотря на то, что при низком пространственном разрешении фактически он содержит два и более класса.

Кроме того, смешанный пиксел может быть не согласованным ни с одним из известных эталонов или же наоборот - иметь сходство со спектральной сигнатурой другого эталона, который в действительности отсутствует в данном пикселе.[15]

Проблему смешанного пиксела нельзя решить улучшением пространственного разрешения, т.к. при этом появляются дополнительные детали изображения, не регистрируемые ранее, т.е. вводятся новые спектральные классы. Примером может служить изображение лесопосадки, которое однородно при низкой разрешающей способности, но при повышении разрешения начинают выделяться отдельные деревья разных видов и разнообразные открытые площадки.

Влияние атмосферы

Нельзя игнорировать тот факт, что при прохождении солнечного излучения через атмосферу происходит изменение его интенсивности, вызванное влиянием присутствующих в атмосфере газов и аэрозолей.[15]

В частности, водяные пары вызывают наиболее слабое поглощение на длинах волн 0,6 и 0,66 мкм, чуть сильнее – на 0,73, 0,82 и 0,91 мкм, наибольшее поглощение - на 0,94 и 1,14 мкм. Поглощение в окрестностях длин волн 1,375, 1,9 и 2,5 мкм практически не позволяет восстановить отражательные характеристики объектов.

Таким образом, ГСИ ДЗЗ содержат информацию не только о состоянии и составе земной поверхности, но и о состоянии атмосферы.

Устранение этого искажающего фактора и перевод изображения из единиц спектральной яркости в единицы спектрального коэффициента отражения проводится с помощью процедуры атмосферной коррекции.[15]

Таким образом, разработка новых методов и алгоритмов обработки и компрессии ГСИ ДЗЗ является важнейшим аспектом развития таких областей, как военная разведка, океанография, картография, геология, разведка водных ресурсов, оценка последствий различных чрезвучайных ситуаций антропогенного, техногенного, природного, экологического, комбинированного характера и других.

* 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].

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 *si* ∈ *S*.

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 *si* ∈ *S*. 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. [6]

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-либо 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 участвовать 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*.*[6]

* + 1. Лексические методы сжатия

Входная последовательность символов в данных методах рассматривается как последовательность строк, содержащих произвольное количество символов. Суть лексических методов состоит в замене строк символов на коды, которые можно интерпретировать как индексы строк некоторого словаря. Образующие словарь строки - фразамы. При декодировании осуществляется обратная замена индекса на соответствующую ему фразу словаря. Лесические методы преобразуют исходную последовательность путем ее представления таким образом, что «буквы» в лексических методах являются фразами словаря, состоящими 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. Благодаря подобному построению происходит сжатие. LZ77 и LZ78 – алгоритмы сжатия без потерь, опубликованные в статьях Абрахама Лемпеля и Якоба Зива в 1977 и 1978 годах. Данные алгоритмы – наиболее известные варианты алгоритмов LZ\*, котораые также включают LZW, LZSS, LZMA и другие алгоритмы. Оба алгоритма – алгоритмы с лексическим подходом [7]. LZ77 использует уже просмотренную часть сообщения в качестве словарь. Алгоритм LZ77 использует «скользящее окно», которое эквивалентно неявному использованию лексического подхода, изначально предложенного в LZ78. «Скользящее окно» разделено на две неравных части. Первая часть, большая по размеру, включает уже просмотренную часть сообщения. Вторая, много меньшая, – буфер, содержащий еще не закодированные символы. Как правило, размер окна – несколько килобайт. Буфер много меньше, обычно не более ста байт.

Алгоритм пытается найти фрагмент в словаре, который совпадает с содержанием буфера и закодировать указатель на этот фрагмент. Лексические методы хорошо себя зарекомендовали при сжатии текстов и в качестве алгоритмов, используемых в универсальных архиваторах, таких как 7z и WinRar.[6]

* + 1. Алгоритм RLE

Первый вариант алгоритма

Групповое кодирование — от английского Run Length Encoding (RLE) — один из самых простых алгоритмов архивации графики. Изображение в нем вытягивается в цепочку байт по строкам растра. Сама компрессия в RLE происходит за счет того, что в исходном изображении встречаются последовательности одинаковых байт. Замена их на пары <счетчик повторений, значение> уменьшает избыточность данных.

В данном алгоритме признаком счетчика служат единицы в двух верхних битах считанного файла:

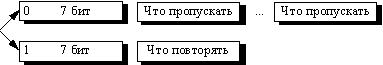
http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image13.gif

Соответственно оставшиеся 6 бит расходуются на счетчик, который может принимать значения от 1 до 64. Строка из 64 повторяющихся байтов преобразуется в два байта, т.е. сжимается в 32 раза.[6]

Второй вариант алгоритма

Второй вариант этого алгоритма имеет больший максимальный коэффициент архивации и меньше увеличивает в размерах исходный файл.

Признаком повтора в данном алгоритме является единица в старшем разряде соответствующего байта:



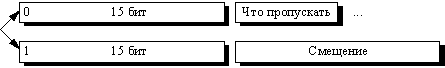
В лучшем случае данный алгоритм сжимает исходный файл в 64 раза, а худшем увеличивает на 1/128. Средние показатели степени компрессии данного алгоритма находятся на уровне показателей первого варианта.[6]

* + 1. Алгоритм LZW

Сжатие в данном алгоритме осуществляется за счет одинаковых цепочек байт.

Существует большое семейство LZ-подобных алгоритмов, различающихся, например, методом поиска повторяющихся цепочек. Один из достаточно простых вариантов этого алгоритма, например, предполагает, что во входном потоке идет либо пара <счетчик, смещение относительно текущей позиции>, либо просто <счетчик> “пропускаемых” байт и сами значения байтов (как во втором варианте алгоритма RLE). При декомпрессии для пары <счетчик, смещение> копируются <счетчик> байт из выходного массива, полученного в результате декомпрессии, на <смещение> байт раньше, а <счетчик> (т.е. число равное счетчику) значений “пропускаемых” байт просто копируются в выходной массив из входного потока.

Данный алгоритм не является симметричным по времени, поскольку требует полного перебора буфера при поиске одинаковых подстрок. Соответственно задание большого буфера повлечёт за собой существенное возрастание времени на процесс компрессии. Однако потенциально построение алгоритма, в котором на <счетчик> и на <смещение> будет выделено по 2 байта (старший бит старшего байта счетчика — признак повтора строки / копирования потока), даёт возможность сжимать все повторяющиеся подстроки размером до 32Кб в буфере размером 64Кб.



При этом получается увеличение размера файла в худшем случае на 32770/32768 (в двух байтах записано, что нужно переписать в выходной поток следующие 215 байт). Максимальный коэффициент компрессии в пределе 8192 раза. В пределе, поскольку максимальное сжатие получится, прикомпрессии 32Кб буфера в 4 байта, а буфер такого размера накапливается не сразу.

Однако, минимальная подстрока, для которой выгодно проводить сжатие, должна состоять в общем случае минимум из 5 байт, что и определяет малую ценность данного алгоритма. К достоинствам LZ можно отнести чрезвычайную простоту алгоритма декомпрессии.

* + 1. Фрактальный алгоритм

Фрактальная компрессия основана на представлении изображения в более компактной форме с помощью коэффициентов системы итерируемых функций (IFS).

IFS представляет собой набор трехмерных аффинных преобразований, преобразующих одно изображение в другое. Преобразованию подвергаются точки в трехмерном пространстве (х\_координата, у\_координата, яркость). Наглядное изложение данного алгоритма приведено в [10].

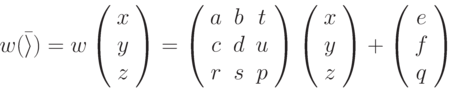
Фрактальная компрессия базируется на следующих определениях и теоремах:

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.

Определение 3. Пусть f:X 🡪 X — преобразование в пространстве Х. Точка xf ∈ X такая, что f(xf) = xf называется неподвижной точкой (аттрактором) преобразования.

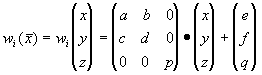
Определение 4. Пусть f:X 🡪 X — преобразованиев метрическом пространстве (Х,d) называется сжимающим, если существует число s: 0 ≤ s < 1, такое, что

d(f(x),f(y)) ≤ s\*d(x,y), ∀ x,y ∈ X

Теорема 1. (О сжимающем преобразовании). Пусть f:X 🡪 X в полном метрическом пространстве (Х, d). Тогда существует в точности одна неподвижная точка xf ∈ X этого преобразования, и для любой точки последовательность {fn(x): n =0,1,2..} сходится к xf.

Определение 5***.*** Изображением называется функция S, определенная на единичном квадрате и принимающая значения от 0 до 1 или http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image59.gif

Пусть трехмерное аффинное преобразование http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image60.gif, записано в следующем виде



Также, пусть определено на компактном подмножестве http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image62.gif декартового квадрата [0..1]x[0..1]. Тогда оно переведет часть поверхности *S* в область http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image63.gif, расположенную со сдвигом *(e,f)* и поворотом, заданным матрицей



При этом, если интерпретировать значение *S* как яркость соответствующих точек, она уменьшится в *p* раз (преобразование обязано быть сжимающим) и изменится на сдвиг *q*.

Определение 6**.** Конечная совокупность *W* сжимающих трехмерных аффинных преобразований http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image65.gif, определенных на областях http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image62.gif, таких, что http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image66.gif и http://lib.ru/TECHBOOKS/ALGO/VATOLIN/algcomp/image67.gif, называется системой итерируемых функций.

IFS однозначно сопоставляется неподвижная точка — изображение. Таким образом, процесс компрессии заключается в поиске коэффициентов системы, а процесс декомпрессии — в проведении итераций системы до стабилизации полученного изображения (неподвижной точки IFS).

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 |
| 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 |

* 1. Метод главных компонент

Метод главных компонент является одним из простейших методов уменьшения размерности данных[17]. Идея метода заключается в том, чтобы в исходном пространстве найти гиперплоскости (ГП) заданной размерности с последующим проектированием выборки на данную ГП.

Выбор ГП обуславливается ошибкой проектирования данных на ГП– она должна быть минимальной. Под ошибкой подразумевается сумма квадратов отклонений.

Рассмотрим произвольный ортонормированный базис в пространстве RD: w1,..., wD, \* wj = δij , где δij = [i = j] – символ Кронекера. Не ограничивая общности, можно считать, что первые d векторов этого базиса w1,...,wd образуют базис искомой ГП. Тогда точки ГП определяются следющим образом:

x = w1t1 +…+ wdtd + µ = W\*t + µ,

где t1,...,td – координаты точки x в базисе ГП, W = (w1|...|wd) ∈ RD×d – матрица, столбцы которой представляют собой базисные вектора w1,...,wd, µ ∈ RD – вектор сдвига. Поиск ГП, обеспечивающей минимальную квадратичную ошибку проектирования, может быть записан следующим образом

J = →

Далее требуется ввод следующих дополнительных обозначений:

X = – выборочное среднее,

S = – выборочная матрица ковариаций.

Рассмотрим собственные вектора и собственные значения матрицы S: S = QΛQT , где Λ = diag(λ1, ..., λD), Q – ортогональная матрица (QTQ = I), в столбцах которой стоят собственные вектора. Предположим, без ограничения общности, что собственные значения λ1 ≥ λ2 ≥ · · · ≥ λD, т.е. отсортированы по убыванию. Можно показать [17], что задача оптимизации имеет следующее аналитическое решение: w1,..., wd – собственные вектора матрицы S, отвечающие d наибольшим собственным значениям λ1 ≥ λ2 ≥ · · · ≥ λd, µ = , tn = WT (xn − ). При этом значение критерия J в точке минимума равно . Таким образом, величина ошибки проектирования для оптимальной ГП составляет сумму дисперсий данных по отбрасываемым размерностям, определяемым собственными векторами wd+1,..., wD.

Итак, метод главных компонент предполагает переход от исходного базиса к базису из собственных векторов матрицы ковариации S с дальнейшим отбрасыванием проекций выборки на собственные вектора, отвечающие D − d наименьшим собственным значениям. В базисе из собственных векторов матрица ковариации S имеет диагональный вид Λ = diag(λ1, ..., λD).

Таким образом, признаки, получаемые с помощью метода главных компонент, являются некоррелированными. Переход к некоррелированным признакам часто является разумным методом предварительной обработки исходных данных. Поэтому метод главных компонент применяется и в случае d = D.

1. Постановка задачи

Задан набор из 360 растровых ГСИ ДЗЗ (рис.3), полученных со спектрометра Hyperion, обладающие следующими характеристиками:

Геометрические размеры ~320x1200 пикселей по обеим осям;

Глубина цвета 16 бит/пиксель;

Режим: монохромный;

Разработать алгоритмическое и программное обеспечение решения задачи кодирования ГСИ ДЗЗ по методу главных компонент

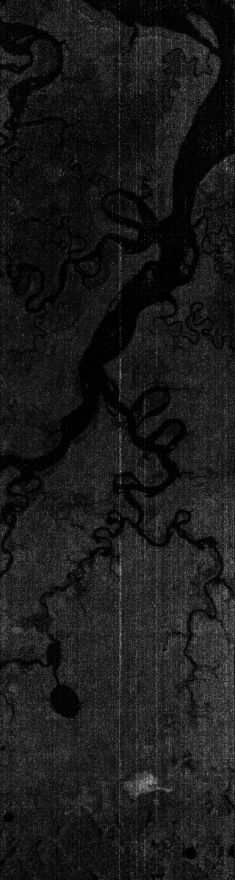
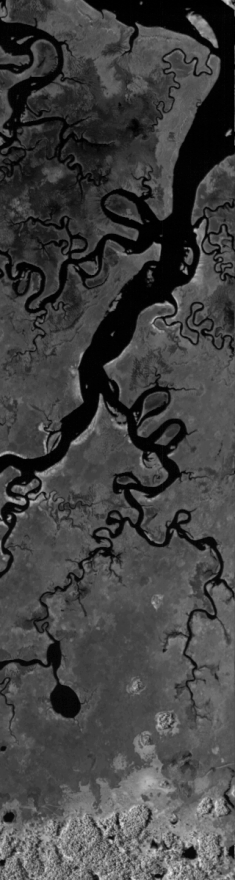


Рис. 3

1. Практическая часть
   1. Описание разработанных алгоритмов
      1. Алгоритм формирования матрицы изображений №1

*Назначение:* алгоритм предназначен для построчного формирования матрицы изображений, каждая из строк которой является векторным представлением отдельного изображения. При этом, элементами получаемого вектора являются значения яркости пикселов.

*Входные параметры:*

bitmap - изображение, представленное последовательностью значений яркости его пикселей.

С – номер обрабатываемого изображения

*Шаги алгоритма:*

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: “Изображние не монохромно”. 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”.
   * 1. Алгоритм формирования матрицы изображений №2

*Назначение:* алгоритм предназначен для построчного формирования матрицы изображений, каждая из строк которой является векторным представлением отдельного изображения. При этом, элементами получаемого вектора являются значения яркости пикселов.

*Входные параметры:*

bitmap - изображение, представленное последовательностью значений яркости его пикселей.

С – номер обрабатываемого изображения

*Шаги алгоритма:*

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: “Изображние не монохромно”. 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 формирования матрицы изображений №3

*Назначение:* алгоритм предназначен для построчного формирования матрицы изображений, каждая из строк которой является векторным представлением отдельного изображения. При этом, элементами получаемого вектора являются значения яркости пикселов.

*Входные параметры:*

bitmap - изображение, представленное последовательностью значений яркости его пикселей.

С – номер обрабатываемого изображения

*Шаги алгоритма:*

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: “Изображние не монохромно”. 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: “Изображние не монохромно”. После чего, обработка изображений будет прервана.
5. После того, как пройдена проверка на монохромность пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
6. 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.
7. После того, как X координата примет значение 0, алгоритм продолжит свою работу с пункта 1, при этом координатами нового пиксела будут [X,Y+1].
8. При достижении координатой X значения Height, а координатой Y значения Width, алгоритм записывает полученную последовательность байт в файл “ColumnZmey.bin”
   * 1. Алгоритм формирования матрицы изображений №4

*Назначение:* алгоритм предназначен для построчного формирования матрицы изображений, каждая из строк которой является векторным представлением отдельного изображения. При этом, элементами получаемого вектора являются значения яркости пикселов.

*Входные параметры:*

bitmap - изображение, представленное последовательностью значений яркости его пикселей.

С – номер обрабатываемого изображения

*Шаги алгоритма:*

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: “Изображние не монохромно”. 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: “Изображние не монохромно”. После чего, обработка изображений будет прервана.
5. После того, как пройдена проверка на монохромность пиксела, значение содержания в нём красного цвета записывается в лист байтов “fileForColorByte».
6. 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.
7. После того, как Y координата примет значение 0, алгоритм продолжит свою работу с пункта 1, при этом координатами нового пиксела будут [X+1,Y].
8. При достижении координатой X значения Height, а координатой Y значения Width, алгоритм записывает полученную последовательность байт в файл “ StringZmey.bin”
   * 1. Алгоритм метода вращения

*Назначение:* алгоритм предназначен для вычисления собственных векторов и собственных значений ковариационной матрицы

*Входные параметры:*

A – ковариационная матрица, собственные вектора и значения которой будут вычисляться методом Якоби

*Шаги алгоритма:*

1. Положить k=0, A(0)=A и задать ε>0, где k – номер итерации алгоритма
2. Выделить в верхней треугольной наддиагональной части матрицы A(k) максимальный по модулю элемент , i<j. Если ||⩽ε для всех i≠j процесс завершить. Собственные значения определяются по следующей формуле:

λi(A(k)) = , 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 ∣∣∣∣>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.
   * 1. Алгоритм умножения матриц

*Назначение:* алгоритм предназначен для вычисления произведения двух матриц

*Входные параметры:*

А – матрица, представляющая собой первый множитель.

В – матрица, представляющая собой второй множитель.

*Содержательный смысл:* Произведением двух матриц *А* и *В* называется матрица *С*, элемент которой, находящийся на пересечении *i*-й строки и *j*-го столбца, равен сумме произведений элементов *i*-й строки матрицы *А* на соответствующие (по порядку) элементы *j*-го столбца матрицы *В*.

Из данного определения следует формула элемента матрицы *C*:

https://function-x.ru/chapter2/matrixes_clip_image097.gif

Таким образом, находятся очередные элементы матрицы вращения.

* + 1. Алгоритм нахождения скалярного произведения векторов

*Назначение.* Данный алгоритм предназначен для вычисления скалярного произведения векторов, которое потребуется при компрессии и декомпрессии данных.

*Входные данные.*

Vectors – массив векторов, для которых будет вычисляться скалярное произведение.

*Содержательный смысл.* Скалярное произведение будет находиться по следующей формуле

* 1. Условие разработки и функционирования ПО

Среда разработки - Microsoft Visual Studio 2017.

Язык разработки – C#.

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

1. Описание разработанного СПО
   1. Класс Form
      1. Назначение класса Form1

Отображение графического интерфейса СПО для реализации возможности взаимодействия пользователя с продуктом.

* + 1. Описание полей и методов
* Поле CAS.

*Назначение.* Объявление объекта класса ColumnAndString с цель последующего использования его для составления матрицы изоражений.

*Объявление*

private ColumgAndString CAS = new ColumgAndString();

* Поле files.

*Назначение.* Объявление списка изображений, который в дальнейшем будет содержать выбранные пользователем изображения для их последующей компрессии.

*Объявление.*

private List<Image> files = new List<Image>();

* Поле countImages

*Назначение.* Объявление переменной, которая будет хранить количество выбранных пользователем изображений для компрессии.

*Объявление.*

private int countImages

* Метод GetcountImages

*Назначение.* Получение количества изображений для компрессии.

*Прототип.* public int GetcountImages();

* Метод SetcountImages

*Назначение.* Инициализация переменной countImages.

*Входные параметры*

I – целочисленное значение, обозначающее количество изображений, выбранное пользователем для компрессии.

*Прототип*. public void SetcountImages(int i);

* Метод GetCovarMatrix

*Назначение.* Данный метод предназначен для поэлементного расчёта ковариационной матрицы.

*Входные параметры*

nums – пара векторов, участвующих в расчёте элемента ковариационной матрицы,

means – массив значений с плавающей точкой, содержащий средние значения всех векторов исходной матрицы,

jter – целочисленное значение, являющееся номером столбца матрицы, в котором будет располежен расчитываемый элемент,

iter - целочисленное значение, являющееся номером строки матрицы, в которой будет располежен расчитываемый элемент,

h – целочисленное значение, содержащее общее количество пикселей в одном изображении.

*Прототип*. public double GetCovarMatrix(byte[][] nums, double[] means, int jter, int iter, int h).

* Метод Matrix\_Click

*Назначение.* Данный метод представляет собой обработчик нажатия кпопки «Построить матрицы ковариаций». Так же проверяет наличие и корректность введённым пользователем значения погрешности для нахождения собственных значений и собственных векторов матрицы ковариаций методом Якоби.

*Прототип.* private void Matrix\_Click(object sender, EventArgs e);

* Метод BuildMatrix

*Назначение.* Данный метод вычисляет среднее значение для каждого пикселя во всех выбранный пользователем изображениях. Так же вызывает методы другого класса, для нахождения собственных значений и векторов матрицы исходных изображений. После чего реализует механизм записи полученных результатов в файл, т.е. записывает полученные ковариационные матрицы, их собственные вектора и числа в файл.

*Прототип.* public void BuildMatrix();

* 1. Класс Rotation
     1. Назначение класса Rotation

Класс реализовывает метод Якоби для нахождения собственных значений и векторов ковариационных матриц.

* + 1. Описание полей и методов
* Поле eps

*Назначение.* Число с плавающей точкой, для хранения значения погрешности в методе Якоби.

*Объявление. private double eps*

* Поле n

*Назначение.* Целочисленное значение для хранения строк и столбцов расчитываемое матрицы.

*Объявление.* private int n;

* Поле MatrixForOwnVectors

*Назначение.* Матрица, содержащая собственные вектора матрицы ковариаций.

*Объявление. public double[][] MatrixForOwnVectors = new double[n][];*

* Метод Multiplication

*Назначение.* Метод реализовывает алгоритм перемножения матриц.

*Входные параметры*

a – матрица, представляющая собой первый множитель в матричном произведении

b – матрица, представляющая собой второй множитель в матричном произведеии

*Прототип.* public double[][] Multiplication(double[][] a, double[][] b);

*Реализация основного вычислительно блока.*

double[][] r = new double[n][];

for (int i = 0; i < n; i++)

r[i] = new double[n];

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

for (int k = 0; k < n; k++)

r[i][j] += a[i][k] \* b[k][j];

}

}

* Метод RotationMethod

*Назначение.* Данный метод рализовывает алгоритм Якоби для нахождения собственных значений и собственных векторов матрицы ковариаций.

*Входные параметры*

Matrix – матрица ковариаций, представленная двумерным массивом чисел с плаващей точкой.

*Прототип.* public double[][] RotationMethod(double[][] Matrix);

*Реализация основного вычислительно блока*

int iteration = -1;

double max = 0;

int MaxI = 0, MaxJ = 0;

do

{

max = 0;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (j <= i

continue;

if (max < Math.Abs(Matrix[i][j]))

{

max = Matrix[i][j];

MaxI = i;

MaxJ = j;

}

}

}

iteration++;

double angle = 0;

double forAtan;

double Cos, Sin

forAtan = (2 \* max) / (Matrix[MaxI][MaxI] - Matrix[MaxJ][MaxJ]);

angle = 0.5 \* Math.Atan(forAtan);

Cos = Math.Cos(angle);

Sin = Math.Sin(angle);

double[][] RotMat = new double[n][];

for (int i = 0; i < n; i++)

RotMat[i] = new double[n];

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (i == MaxI && j == MaxI)

RotMat[i][j] = Cos;

else if (i == MaxI && j == MaxJ)

RotMat[i][j] = -1 \* Sin;

else if (i == MaxJ && j == MaxI)

RotMat[i][j] = Sin;

else if (i == MaxJ && j == MaxJ)

RotMat[i][j] = Cos;

else if (i == j)

RotMat[i][j] = 1;

else

RotMat[i][j] = 0;

}

}

if (iteration == 0)

{

MatrixForOwnVectors = RotMat;

}

else

{

MatrixForOwnVectors = Multiplication(MatrixForOwnVectors, RotMat);

}

double tmp;

double[][] tmpMat = (double[][])RotMat.Clone();

for (int i = 0; i < n; i++)

{

for (int j = 0; j < i; j++)

{

tmp = tmpMat[i][j];

tmpMat[i][j] = tmpMat[j][i];

tmpMat[j][i] = tmp;

}

}

double[][] SemiIT;

SemiIT = Multiplication(tmpMat, Matrix);

Matrix = Multiplication(SemiIT, RotMat);

}

while (max > eps

return Matrix;

* Метод Scal

*Назначение.* Данный метод рализовывает алгоритм нахождения скалярного произведения, основываясь на формуле, приведённой в главе 3.

*Входные параметры*

Vectors – пара векторов, эементами которых являются числа с плавающей точкой.

*Прототип.* public double Scal ( double[][] Vectors);

*Реализация основного вычислительно блока*

double sum = 0;

for (int i = 0; i < MatrixForOwnVectors.GetLength(1); i++)

{

sum += Vectors[0][i] \* Vectors[1][i];

}

return sum;

* 1. Класс ColumgAndString
     1. Назначение класса

Класс предназначен для формирования вектора из входящего изображения различными способами, описанными в части 3.

* + 1. Описание полей и методов
* Поле fileForColorByte

*Назначение.* Данное поле предназначено для хранения значения цветности пикселов в изображении.

*Объявление.* private List<Byte> fileForColorByte = new List<Byte>();

* ПолеMono*.*

*Назначение.* Данное поле выполняет роль флага, который меняет своё значение на “false” в том случае, если изображение не является монохромным.

*Объявление.* private bool Mono;

* Метод ColumnEze

*Назначение.* Данный метод предназначен для формирования вектора значений яркости пикселов входящего изображения. Так же в его функциональность входит запись полученного вектора в файл, содержащий подобное представление иных обработанных этим же методом изображений, и запись полученного вектора в отдельный файл.

*Входные параметры.*

bitmap – битмат, хранящий изображение, которое будет обрабатывать данный метод

C – целочисленное значение, хранящее номер обрабатываемого изображения

*Прототип.* public void ColumnEze(Bitmap bitmap, int C);

*Реализация основного вычислительно блока*

Color color;

for (int y = 0; y < bitmap.Width; y++)

{

if (Mono == true)

{

for (int x = 0; x < bitmap.Height; x++)

{

color = bitmap.GetPixel(y, x);

Int32 R = color.R; Int32 G = color.G; Int32 B = color.B;

if (R == G && G == B)

{

Byte bright8;

Byte R8 = color.R;

bright8 = R8;

fileForColorByte.Add(bright8);

}

else

{

Mono = false;

MessageBox.Show("Изображение не монохромно");

break;

}

}

}

else

{

MessageBox.Show("Изображение не монохромно");

break;

}

}

string name = C.ToString() + " ColumnEze.bin";

File.WriteAllBytes(name, fileForColorByte.ToArray());

SummaOfBitmaps(fileForColorByte, "AllColumnEze.txt");

fileForColorByte.Clear();

Mono = true;

* Метод ColumnZmey

*Назначение.* Данный метод предназначен для формирования вектора значений яркости пикселов входящего изображения. Так же в его функциональность входит запись полученного вектора в файл, содержащий подобное представление иных обработанных этим же методом изображений, и запись полученного вектора в отдельный файл.

*Входные параметры.*

bitmap – битмат, хранящий изображение, которое будет обрабатывать данный метод

C – целочисленное значение, хранящее номер обрабатываемого изображения

*Прототип.* public void ColumnZmey (Bitmap bitmap, int C);

*Реализация основного вычислительно блока*

Color color;

for (int y = 0; y < bitmap.Width; y++)

{

if (Mono == false)

break;

for (int x = 0; x < bitmap.Height; x++)

{

if (Mono == false)

break;

color = bitmap.GetPixel(y, x);

Int32 R = color.R; Int32 G = color.G; Int32 B = color.B;

if (R == G && G == B)

{

Byte R8 = color.R;

fileForColorByte.Add(R8);

}

else

{

Mono = false;

MessageBox.Show("Изображение не монохромно");

fileForColorByte.Clear();

break;

}

if (x == bitmap.Height - 1)

{

y++;

int xx = x;

for (; xx >= 0; xx--)

{

color = bitmap.GetPixel(y, xx);

R = color.R; G = color.G; B = color.B;

if (R == G && G == B)

{

Byte R8 = color.R;

fileForColorByte.Add(R8);

}

else

{

Mono = false;

MessageBox.Show("Изображение не монохромно");

fileForColorByte.Clear();

break;

}

}

}

}

}

string name = C.ToString() + " ColumnZmey.txt";

File.WriteAllBytes(name, fileForColorByte.ToArray());

SummaOfBitmaps(fileForColorByte, "AllColumnZmey.txt");

fileForColorByte.Clear();

* Метод StringEze

*Назначение.* Данный метод предназначен для формирования вектора значений яркости пикселов входящего изображения. Так же в его функциональность входит запись полученного вектора в файл, содержащий подобное представление иных обработанных этим же методом изображений, и запись полученного вектора в отдельный файл.

*Входные параметры.*

bitmap – битмат, хранящий изображение, которое будет обрабатывать данный метод

C – целочисленное значение, хранящее номер обрабатываемого изображения

*Прототип.* public void StringEze (Bitmap bitmap, int C);

*Реализация основного вычислительно блока*

Color color;

for (int y = 0; y < bitmap.Height; y++)

{

if (Mono == false)

break;

for (int x = 0; x < bitmap.Width; x++)

{

color = bitmap.GetPixel(x, y);

Int32 R = color.R;

Int32 G = color.G;

Int32 B = color.B;

if (R == G && G == B)

{

Byte R8 = color.R;

fileForColorByte.Add(R8);

}

else

{

Mono = false;

MessageBox.Show("Изображение не монохромно");

fileForColorByte.Clear();

break;

}

}

}

string name = C.ToString() + " StringEze.txt";

File.WriteAllBytes(name, fileForColorByte.ToArray());

SummaOfBitmaps(fileForColorByte, "AllStringEze.txt", C);

fileForColorByte.Clear();

Mono = true;

* Метод StringZmey

*Назначение.* Данный метод предназначен для формирования вектора значений яркости пикселов входящего изображения. Так же в его функциональность входит запись полученного вектора в файл, содержащий подобное представление иных обработанных этим же методом изображений, и запись полученного вектора в отдельный файл.

*Входные параметры.*

bitmap – битмат, хранящий изображение, которое будет обрабатывать данный метод

C – целочисленное значение, хранящее номер обрабатываемого изображения

*Прототип.* public void StringZmey (Bitmap bitmap, int C);

*Реализация основного вычислительно блока*

Color color;

for (int y = 0; y < bitmap.Height; y++)

{

if (Mono == false)

break;

for (int x = 0; x < bitmap.Width; x++)

{

color = bitmap.GetPixel(x, y);

Int16 R = color.R;

Int16 G = color.G;

Int16 B = color.B;

if (R == G && G == B

{

Byte R8 = color.R;

fileForColorByte.Add(R8);

if (x == bitmap.Width - 1)

{

y++;

int xx = x;

for (; xx >= 0; xx--)

{

color = bitmap.GetPixel(xx, y);

R = color.R; G = color.G; B = color.B;

if (R == G && G == B)

{

R8 = color.R;

fileForColorByte.Add(R8);

}

else

{

Mono = false;

MessageBox.Show("Изображение не монохромно");

fileForColorByte.Clear();

break;

}

}

}

}

else

{

Mono = false;

MessageBox.Show("Изображение не монохромно");

fileForColorByte.Clear();

break;

}

}

}

string name = C.ToString() + " StringZmey.txt";

File.WriteAllBytes(name, fileForColorByte.ToArray());

SummaOfBitmaps(fileForColorByte, "AllStringZmey.txt");

fileForColorByte.Clear();

Mono = true;

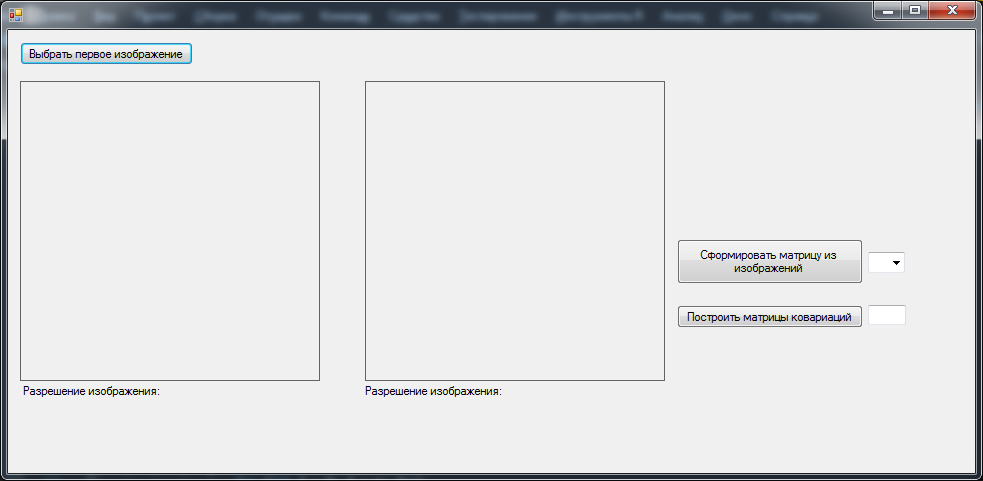
* 1. Описание разработанного модуля

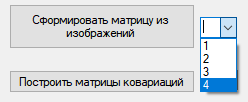
*Название*. HICompression

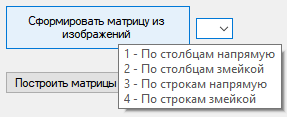
*Назначение*. Данный программный модуль предназначен для компрессии и декомпрессии ГСИ ДЗЗ.

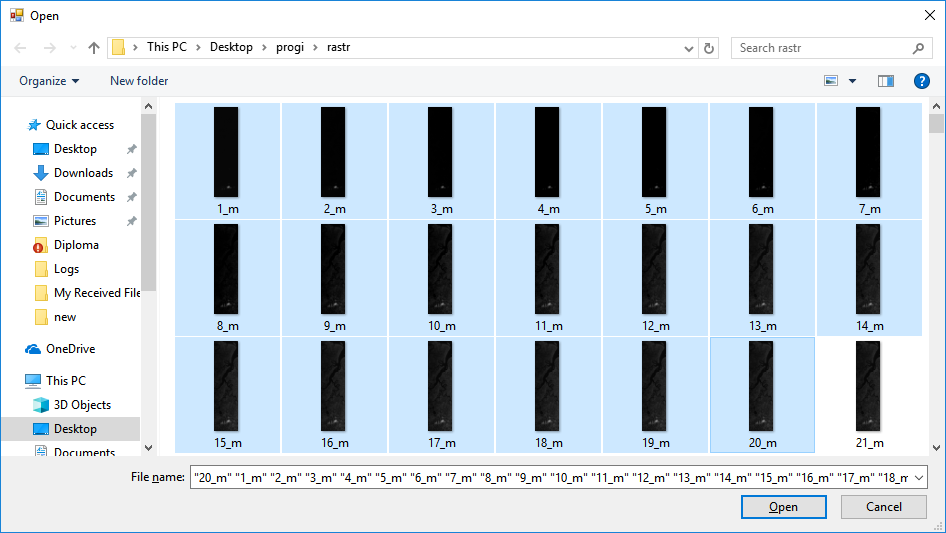
В его основе лежит метод главных компонент, часть нахождения собственных значений и векторов матрицы ковариаций которого реализована методом Якоби. Данный итерационный метод может уступать прямым методам по времененным затратам, но он даёт неоспаримое преимущество в виде возможности вариативного выбора погрешности вычислений, что может повлиять на точность полученных результатов. Например, время его работы можно уменишить, задав большую погрешность, в тех областях, где важна не высокая точность полученных данных, а их объём.

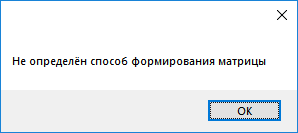
* 1. Руководство оператора
* Для запуска программы дважды нажмите левой кнопкой мыши (ЛКМ) на её ярлык. Программа выведет главное окно



* 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:

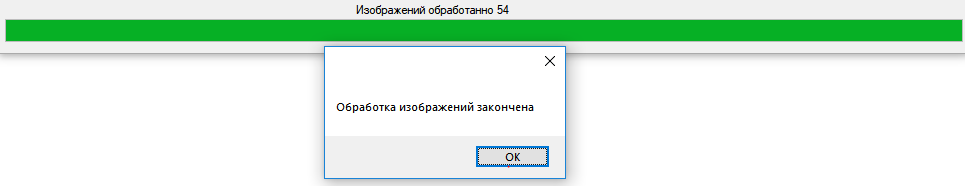


* После успешного выбора изображений и нажатия кнопки “Open”,начнётся процесс формирования матрицы из выбранных изображений, который будет сопровождаться визуальной индикацией хода процесса:





* 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. Результаты практических экспериментов

Ниже представлены исходные ГСИ ДЗЗ (рис.4) и данные, полученые на выходе работы данного программного модуля (рис.5).

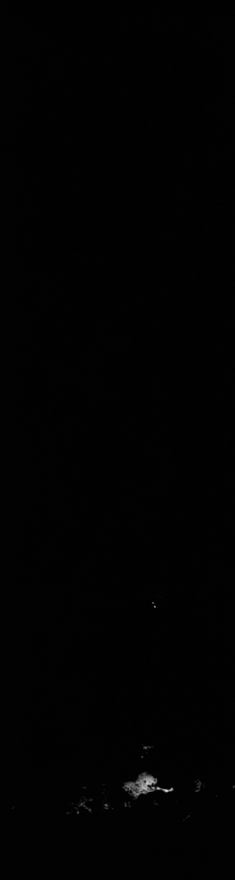


Рис. 4

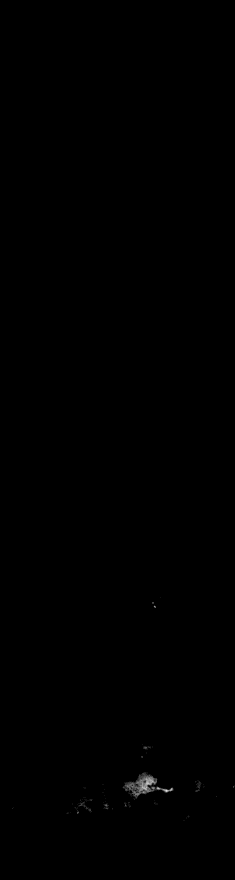
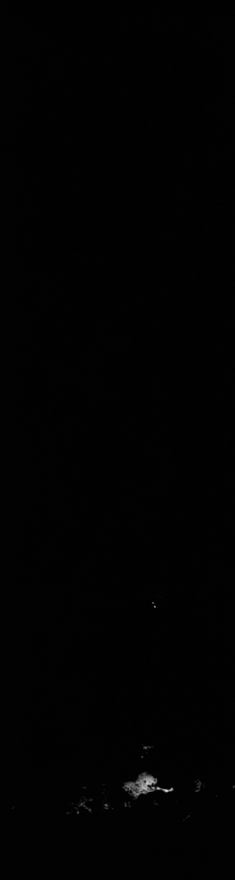


Рис. 5

1. Выводы

В результате анализа литературных источников установлено, что данные ГСД ДЗЗ являются новейшим и актуальнейшим средством получения пространственно распределённых данных об объектах земной поверхности

ГСД ДЗЗ новейший и самый перспективных метод дистанционного зондирования с помощью которого возможно получение наиболее полного описания различных объектов земной поверности, но обладающей громандной информационной избыточностью. Очевидна актуальность задачи использования существующих и разработки новых, перспективных мтодов сжатия ГСИ ДЗЗ.

Установленно, что метод главных компонент является одним из базовых в решении задачи формирования конструктивных систем кодирующих функций ГСИ ДЗЗ

Разработанно алгоритмическое обеспечение нахождения собственных векторГСИ ДЗЗ со ов матрицы ковариаций, получнной по набор из 360 16 битных монохромных растровых изображений ГСИ ДЗЗ, полученных со спектрометра Hyperion.

На базе разработанного алгоритмического обеспечения созданно оригинальное СПО на языке программирования C#.

Практическое тестирование созданного СПО на реальных ГСИ данных выявило его работоспособность и высокую эффективность.

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.
10. Борусяк A.В. Обработка больших объемов графической информации методом статистического кодирования и контекстного моделирования
11. Michael F. Barnsley Fractal Image Compression, 1988
12. Ю.Ф.Книжников, В.И.Кравцова, О.В.Тутубалина. Аэрокосмические методы географических исследований : учебник для студ. учреждений высш. проф. образования / — 2-е изд., перераб. и доп. — М. : Издательский центр «Академия», 2011. — 416 с., [32] с. цв. ил.: ил.
13. Перспективыне информационные технологии дистанционного зондирования Земли:моногр. / под редакцией В.А. Сойфера. – Самара: Новая техника, 2015.-256с.
14. Безуглая Э.Ю. Мониторинг состояния загрязнения атмосферы в городах. Л.: Гидрометеоиздат, 1986. 199 с
15. Keshava, N. Survey of Spectral Unmixing Algorithms // Lincoln Laboratory Journal. - 2003. - V. 14, N 1 - P. 55-78.
16. Журавель Ю.Н., Федосеев А.А. Особенности обработки гиперспектральных данных дистанционного зондирования
17. Фонд знаний Ломоносов. Дистанционное зондирование Земли [Электронный ресурс] /Дистанционное зондирование Земли - Электрон. ст. - Б. м., Б. г. - Режим доступа к ст.: <http://www.lomonosov-fund.ru/enc/ru/encyclopedia:01330:article>
18. J.J. Sylvester. On the reduction of a bilinear quantic of the nth order to the form of a sum of n products by a double orthogonal substitution // Messenger of Mathematics, 19, 1889, pp. 42–46.
19. TIFF 6.0 specification [Электронный ресурс], режим доступа к ст.: <https://www.itu.int/itudoc/itu-t/com16/tiff-fx/docs/tiff6.pdf>
20. Приложение

Основные методы, используемые в программном модуле:

private void BuildMatrix()

{

double eps;

if (Eps.Text == "")

{

MessageBox.Show("Введите погрешность для вычисления собственных значений и векторов матрицы");

return;

}

else

{

try

{

eps = Convert.ToDouble(Eps.Text);

}

catch (FormatException)

{

MessageBox.Show("Введённое число имеет не верный формат");

return;

}

}

Rotation MatrixForRot = new Rotation(countImages, eps int h = CAS.AllPictures.Length; byte[][] CompareVectors = new byte[2][]; for (int i = 0; i < 2; i++)

{

CompareVectors[i] = new byte[h];

}

double[] meanMas = new double[countImages];

double[] dispmatr = new double[countImages];

double[][] MatrixK = new double[countImages][];

for (int i = 0; i < countImages; i++)

MatrixK[i] = new double[countImages];

using (FileStream fsSource = new FileStream("AllColumnEze.txt", FileMode.Open, FileAccess.Read))

{

byte[] bufer = new byte[fsSource.Length];

fsSource.Read(bufer, 0, (int)fsSource.Length);

for (int i = 0; i < countImages; i++)

{

double mean = 0;

for (int k = i \* h; k < h \* (i + 1); k++)

{

mean += bufer[k];

}

meanMas[i] = (mean / h);

}

fsSource.Close();

}

for (int i = 0; i < countImages; i++)

{

using (FileStream fsSource = new FileStream("AllColumnEze.txt", FileMode.Open, FileAccess.Read))

{

byte[] bufer = new byte[fsSource.Length];

fsSource.Read(bufer, 0, (int)fsSource.Length);

for (int j = 0; j < countImages; j++)

{

if (i == j)

{

Array.Copy(bufer, h \* i, CompareVectors[0], 0, h);

Array.Copy(CompareVectors[0], 0, CompareVectors[1], 0, h);

}

else

{

Array.Copy(bufer, h \* i, CompareVectors[0], 0, h);

Array.Copy(bufer, h \* j, CompareVectors[1], 0, h);

}

double cov = GetCovarMatrix(CompareVectors, meanMas, j, i, h);

MatrixK[i][j] = cov;

}

fsSource.Close();

}

}

using (StreamWriter file = new StreamWriter("Result.txt", true, System.Text.Encoding.Default))

{

file.WriteLine("Матрица ковариаций:");

for (int i = 0; i < countImages; i++)

{

for (int j = 0; j < countImages; j++)

file.Write(MatrixK[i][j].ToString("0.000") + " ");

file.WriteLine();

}

double[][] MatrixResult = MatrixForRot.RotationMethod(MatrixK);

file.WriteLine("Собственные значения:");

for (int i = 0; i < countImages; i++)

{

file.Write(MatrixResult[i][i].ToString("0.000") + " ");

file.WriteLine();

}

file.WriteLine("Собственные вектора:");

for (int j = 0; j < countImages; j++)

{

for (int i = 0; i < countImages; i++)

{

file.Write(MatrixForRot.MatrixForOwnVectors[i][j].ToString("0.000") + " ");

}

file.WriteLine();

}

}

MessageBox.Show("Матрица ковариаций посчитана и записана");

}

public double[][] RotationMethod(double[][] Matrix)

{

int iteration = -1;

double max = 0;

int MaxI = 0, MaxJ = 0;

do

{

max = 0;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (j <= i

continue;

if (max < Math.Abs(Matrix[i][j]))

{

max = Matrix[i][j];

MaxI = i;

MaxJ = j;

}

}

}

iteration++;

double angle = 0;

double forAtan;

double Cos, Sin

forAtan = (2 \* max) / (Matrix[MaxI][MaxI] - Matrix[MaxJ][MaxJ]);

angle = 0.5 \* Math.Atan(forAtan);

Cos = Math.Cos(angle);

Sin = Math.Sin(angle);

double[][] RotMat = new double[n][];

for (int i = 0; i < n; i++)

RotMat[i] = new double[n];

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (i == MaxI && j == MaxI)

RotMat[i][j] = Cos;

else if (i == MaxI && j == MaxJ)

RotMat[i][j] = -1 \* Sin;

else if (i == MaxJ && j == MaxI)

RotMat[i][j] = Sin;

else if (i == MaxJ && j == MaxJ)

RotMat[i][j] = Cos;

else if (i == j)

RotMat[i][j] = 1;

else

RotMat[i][j] = 0;

}

}

if (iteration == 0)

{

MatrixForOwnVectors = RotMat;

}

else

{

MatrixForOwnVectors = Multiplication(MatrixForOwnVectors, RotMat);

}

double tmp;

double[][] tmpMat = (double[][])RotMat.Clone();

for (int i = 0; i < n; i++)

{

for (int j = 0; j < i; j++)

{

tmp = tmpMat[i][j];

tmpMat[i][j] = tmpMat[j][i];

tmpMat[j][i] = tmp;

}

}

double[][] SemiIT;

SemiIT = Multiplication(tmpMat, Matrix);

Matrix = Multiplication(SemiIT, RotMat);

}

while (max > eps

return Matrix;

}

public double Scal ( double[][] Vectors)

{

double sum = 0;

for (int i = 0; i < MatrixForOwnVectors.GetLength(1); i++)

{

sum += Vectors[0][i] \* Vectors[1][i];

}

return sum;

}