МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
И КОММУНИКАЦИИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

На правах рукописи

УДК 004.056.5

БУРИЕВ КОМИЛЖОН ХУРРАМОВИЧ

Разработка генетического алгоритма и прикладных программ для фрактального сжатия изображения

5А330601-Программный инжиниринг

Диссертация

на соискание академической степени магистра

Научный руководитель:

Рахманов А.Т.

Ташкент – 2015

Содержание

Введение

Успех деятельности предприятий, да и в целом государства, в силу ограниченности естественных, природных ресурсов, в значительной мере сегодня определяется тем, насколько широко внедряются достижения научно-технического прогресса, наукоемкие технологии, уровнем профессиональной подготовленности кадров. Исторически сложилось так, что на пороге XXI века в Республике Узбекистан сформирован интеллектуальный потенциал, который по своему уровню развития, инновационным открытиям, возможностям превосходит сегодня многие развивающиеся страны мира, а во многом и не уступает экономически развитым странам.

Постановлением Президента от 21.03.2012 г. N ПП-1730 принята Программа дальнейшего внедрения и развития информационно-коммуникационных технологий в Республике Узбекистан на 2012–2014. Разрабатываемая программа обеспечит удобство в предоставлении информации клиентам государственных, а также коммерческих банков.

Актуальность темы исследования. Задача эффективного сжатия данных всегда была одной из самых актуальных в информатике, а с развитием компьютерной, цифровой техники и Интернет она приобрела еще большее значение. Так, с начала 1980-х гг. было разработано множество алгоритмов, направленных на эффективное сжатие цифровых изображений для их передачи и хранения. Развитие теории фракталов и соответствующего математического аппарата в конце XX в. повлекло разработку принципиально нового алгоритма сжатия изображений — фрактального. Идея фрактального метода сжатия изображений была сформулирована Майклом Барнсли в 1990 году и концептуально заключается в следующем. Для заданного изображения по специальным правилам строится система сжимающих отображений, для которых оно является аттрактором. Тогда, в силу теоремы о неподвижной точке, итерационный процесс,

сформированный на базе этих отображений, сходится к своему аттрактору при любом выборе начального изображения. Эффект сжатия достигается за счет того, что вместо исходного изображения для его восстановления необходимо хранить лишь коэффициенты преобразования.

К числу отличительных можно отнести ряд особенностей, обуславливающих интерес к данному методу сжатия на протяжении более двух десятилетий, а именно:

- **Высокий коэффициент сжатия, т.е. отношение объема исходной информации к объему кода.**
- **Независимость качества декодированного изображения от масштаба его просмотра;**
- **Существенная асимметричность, т.е. время кодирования заметно больше времени декодирования.**

Основным недостатком данного метода остается сравнительно большое время кодирования, обусловленное значительным числом перебираемых вариантов. Поэтому разработка модификаций этого алгоритма, направленных на уменьшение времени кодирования, является актуальной.

Степень научной разработанности и нерешенные проблемы. Привлекательность самой идеи фрактального метода сжатия и его достоинства притягивает исследователей к развитию и совершенствованию алгоритмов его реализующих. Предлагаемые модификации касаются практически всех его сторон. В большинстве опубликованных работ по данной теме предлагаются различные методы, направленные на снижение объема перебора доменных блоков и, как следствие, — сокращение времени кодирования. Так, в работе затрагивается способ разбиения доменов и предлагается ограничиться теми, которые окаймляют ранги. В работе используется дискретное косинусоидальное преобразование для классификации всех доменных блоков на некоторое количество классов, в

работе для этого применяются R-деревья, а в работе — самоорганизующаяся нейронная сеть. В работах и авторы используют генетический алгоритм для поиска оптимальных доменов. В работе поиск ограничивается посредством меры информационной энтропии доменов. Другой, менее разработанной областью исследований, является применение нелинейных сжимающих отображений (способов аппроксимации ранговых блоков доменными блоками).

Цели и задачи. Цель работы состоит в проведении сравнительного анализа базовых алгоритмов, установлении особенностей отбора доменных блоков, апробации методов их аппроксимации для повышения эффективности сжатия. Данная цель достигается за счет изменений в методах отбора доменных блоков и способах аппроксимации.

Для достижения указанной цели в магистерской работе поставлены и решены следующие задачи:

Изучение литературных источников и проведение теоретического анализа алгоритмов сжатия изображений с потерями.

Установление особенностей функционирования алгоритмов фрактального сжатия изображений, разработка реализующего их программного обеспечения.

Формирование набора изображений, проведение вычислительных экспериментов по установлению особенностей отбора доменных блоков и их аппроксимации, сравнительный анализ результатов, формулирование выводов. Модификация отдельных этапов алгоритма, формирование его новой версии. Разработка соответствующих программных компонент, проведение экспериментов, сравнительные результаты работы.

Предметом и объектом исследования является — алгоритмы сжатия изображений с потерями.

Методы исследования включают проведение вычислительных экспериментов, сравнительный анализ результатов, численные методы,

математические методы аппроксимации, объектно-ориентированное программирование.

Предполагаемая научная новизна. Научная новизна уже проведенных и планируемых в работе исследований предполагается в следующем:

- Разработано программное обеспечение, реализующее базовые алгоритмы фрактального сжатия и позволяющее проводить сопоставительный анализ.
- Проведены вычислительные эксперименты, сделан сравнительный анализ и выявлены особенности отбора доменных блоков в базовых алгоритмах.
- Анализ применимости нелинейных моделей отображений блоков изображений.

Диссертационная работа состоит из введения, трех глав, заключения. Работа излагается в ___страницах, ___ таблицах и ____рисунков.

I. ГЛАВА. ИЗУЧЕНИЕ И АНАЛИЗ ОБЩИХ ПОЛОЖЕНИЙ АЛГОРИТМОВ СЖАТИЯ ИЗОБРАЖЕНИЙ

1. Представление изображений конечным объемом данных, статистическая избыточность изображений

Компьютерное изображение в его цифровом представлении является набором значений интенсивностей светового потока, распределенных по конечной площади, имеющей обычно прямоугольную форму.

Для простоты рассмотрим сначала монохромные изображения. Тогда интенсивность излучаемой световой энергии с единицы поверхности в точке с координатами (ξ,η) изображения можно представить некоторым числом $B(\xi,\eta)$. Единичный элемент изображения, характеризуемый определенным значением (ξ,η), называется пикселем, а величина $z = f(\xi,\eta)$ - яркостью.

Статистическая и визуальная избыточность изображений

Поток данных об изображении имеет существенное количество излишней информации, которая может быть устранена практически без заметных для глаза искажений.

Существует два типа избыточности:

- **Статистическая избыточность,** связанная с корреляцией и предсказуемостью данных. Эта избыточность может быть устранена без потери информации, исходные данные при этом могут быть полностью восстановлены.
- **Визуальная (субъективная) избыточность,** которую можно устранить с частичной потерей данных, мало влияющих на качество воспроизводимых изображений; это - информация, которую можно изъять из изображения, не нарушая визуально воспринимаемое качество изображений.

Статистическая избыточность изображений. Пусть имеется дискретизированное $M \times N$ пикселей и квантованное с точностью K бит на пиксель монохромное изображение. Следовательно, для хранения этого изображения необходимо $M \times N \times K$ бит информации.

Если предположить, что квантованные значения яркости не равновероятны, то уменьшение информации возможно путем изменения количества бит информации для кодирования пикселей: более вероятные кодируются словами с меньшим количеством бит, менее вероятные - с большим. Этот метод называется кодированием словами переменной длины или энтропийным кодированием.

Пусть квантованный уровень яркости z имеет вероятность $P(z)$ и ему присваивается слово - код длины $L(z)$ бит. Тогда средняя длина кода для всего изображения составит $\hat{L} = \sum_b L(z) \cdot P(z)$ бит на пиксель. Нижняя граница для $\hat{L}$ определяется информационной теоремой и называется энтропией случайной величины:

$$H(f) = -\sum_z P(z) \cdot \log_2 P(z) \le \hat{L}$$

Таким образом, энтропия - это мера количества информации, которую несет случайная величина уровня яркости z.

$H(z) \ge 0$, поскольку $P(z) \in [0,1]$. Из формулы $H(f)$ вытекает, что чем более неравномерно распределение $P(z)$, тем меньше энтропия и тем эффективнее может быть энтропийное кодирование.

Наиболее известные методы эффективного кодирования символов основаны на знании частоты каждого символа присутствующего в сообщении. Зная эти частоты, строят таблицу кодов, обладающую следующими свойствами:

- различные коды могут иметь различное количество бит
- коды символов с большей частотой встречаемости, имеют больше бит, чем коды символов с меньшей частотой
- хотя коды имеют различную битовую длину, они могут быть восстановлены единственным образом.

Этими свойствами обладает известный алгоритм Хаффмана [6].

Визуальная избыточность изображений. Устранение визуальной избыточности изображений является основным резервом сокращения передаваемой информации. Для оптимизации процесса кодирования с точки зрения обеспечения передачи наименьшего объема информации необходимо, с одной стороны, не передавать избыточную информацию, а, с другой, - не допустить чрезмерной потери качества изображения.

До сих пор не существует простой и адекватной модели визуального восприятия изображений, пригодной для оптимизации их кодирования [5].

2. Обзор классов изображений

В течение последних 10 лет в рамках компьютерной графики бурно развивается совершенно новая область - алгоритмы архивации изображений. Появление этой области обусловлено тем, что изображения - это своеобразный тип данных, характеризуемый тремя особенностями:

1. Изображения (как и видео) занимают намного больше места в памяти, чем текст. Так, скромная, не очень качественная иллюстрация на обложке книги размером 500x800 точек, занимает 1.2 Мб - столько же, сколько художественная книга из 400 страниц (60 знаков в строке, 42 строки на странице). В качестве примера можно рассмотреть также, сколько тысяч страниц текста мы сможем поместить на CD-ROM, и как мало там поместится качественных несжатых фотографий. Эта особенность изображений определяет актуальность алгоритмов архивации графики.

2. Второй особенностью изображений является то, что человеческое зрение при анализе изображения оперирует контурами, общим переходом цветов и сравнительно нечувствительно к малым изменениям в изображении. Таким образом, мы можем создать эффективные алгоритмы архивации изображений, в которых декомпрессированное изображение не будет совпадать с оригиналом, однако человек этого не заметит. Данная особенность человеческого зрения позволила создать специальные

алгоритмы сжатия, ориентированные только на изображения. Эти алгоритмы обладают очень высокими характеристиками.

3. Мы можем легко заметить, что изображение, в отличие, например, от текста, обладает избыточностью в 2-х измерениях. Т.е. как правило, соседние точки, как по горизонтали, так и по вертикали, в изображении близки по цвету. Кроме того, мы можем использовать подобие между цветовыми плоскостями R, G и B в наших алгоритмах, что дает возможность создать еще более эффективные алгоритмы. Таким образом, при создании алгоритма компрессии графики мы используем особенности структуры изображения.

Всего на данный момент известно минимум три семейства алгоритмов, которые разработаны исключительно для сжатия изображений, и применяемые в них методы практически невозможно применить к архивации еще каких-либо видов данных.

Для того, чтобы говорить об алгоритмах сжатия изображений, мы должны определиться с несколькими важными вопросами:

1. Какие критерии мы можем предложить для сравнения различных алгоритмов?
2. Какие классы изображений существуют?
3. Какие классы приложений, использующие алгоритмы компрессии графики, существуют, и какие требования они предъявляют к алгоритмам?

Рассмотрим эти вопросы подробнее.

Классы изображений. Статические растровые изображения представляют собой двумерный массив чисел. Элементы этого массива называют пикселами (от английского pixel - picture element). Все изображения можно подразделить на две группы - с палитрой и без нее. У изображений с палитрой в пикселе хранится число - индекс в некотором

одномерном векторе цветов, называемом палитрой. Чаще всего встречаются палитры из 16 и 256 цветов.

Изображения без палитры бывают в какой-либо системе цветопредставления и в градациях серого (grayscale). Для последних значение каждого пиксела интерпретируется как яркость соответствующей точки. Встречаются изображения с 2, 16 и 256 уровнями серого. Одна из интересных практических задач заключается в приведении цветного или черно-белого изображения к двум градациям яркости, например, для печати на лазерном принтере. При использовании некой системы цветопредставления каждый пиксел представляет собой запись (структуру), полями которой являются компоненты цвета. Самой распространенной является система RGB, в которой цвет представлен значениями интенсивности красной (R), зеленой (G) и синей (B) компонент. Существуют и другие системы цветопредставления, такие, как CMYK, CIE XYZccir60-1 и т.п. Ниже мы увидим, как используются цветовые модели при сжатии изображений с потерями.

Для того, чтобы корректнее оценивать степень сжатия, нужно ввести понятие класса изображений. Под классом будет пониматься некая совокупность изображений, применение к которым алгоритма архивации дает качественно одинаковые результаты. Например, для одного класса алгоритм дает очень высокую степень сжатия, для другого - почти не сжимает, для третьего - увеличивает файл в размере. (Известно, что многие алгоритмы в худшем случае увеличивают файл.)

Рассмотрим следующие примеры неформального определения классов изображений:

1. **Класс 1.** Изображения с небольшим количеством цветов (4-16) и большими областями, заполненными одним цветом. Плавные переходы цветов отсутствуют. Примеры: деловая графика - гистограммы, диаграммы, графики и т.п.

2. **Класс 2.** Изображения, с плавными переходами цветов, построенные на компьютере. Примеры: графика презентаций, эскизные модели в САПР, изображения, построенные по методу Гуро.

3. **Класс 3.** Фотореалистичные изображения. Пример: отсканированные фотографии.

4. **Класс 4.** Фотореалистичные изображения с наложением деловой графики. Пример: реклама.

Развивая данную классификацию, в качестве отдельных классов могут быть предложены некачественно отсканированные в 256 градаций серого цвета страницы книг или растровые изображения топографических карт. (Заметим, что этот класс не тождественен классу 4). Формально являясь 8- или 24-битными, они несут даже не растровую, а чисто векторную информацию. Отдельные классы могут образовывать и совсем специфичные изображения: рентгеновские снимки или фотографии в профиль и фас из электронного досье.

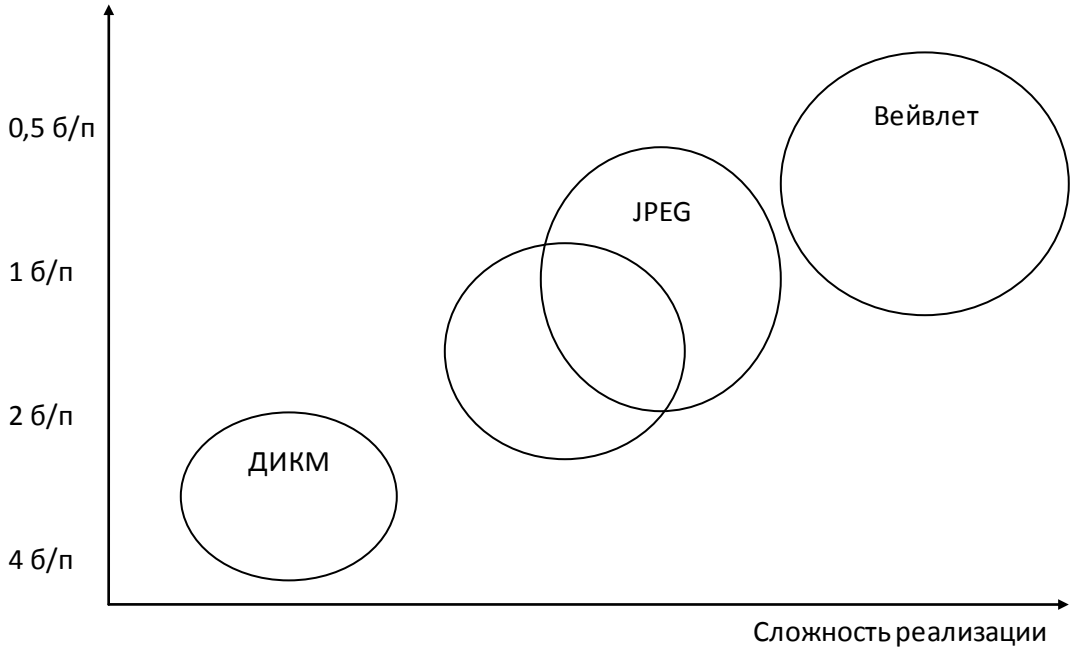
Достаточно сложной и интересной задачей является поиск наилучшего алгоритма для конкретного класса изображений.

Итог: Нет смысла говорить о том, что какой-то алгоритм сжатия лучше другого, если мы не обозначили классы изображений, на которых сравниваются наши алгоритмы.

3. Сравнительный анализ алгоритмов сжатия изображения

В настоящее время в космических системах ДЗЗ отмечается быстрый рост производительности оптико-электронных систем съемки Земли, в то время рост пропускной способности радиолиний передачи данных характеризуется более медленными темпами. Это обстоятельство приводит к возрастанию требований к аппаратуре сжатия видеоинформации. Так, за последние 10 лет характерная степень сжатия алгоритмов, аппаратно реализуемых на борту, КА возросла с 2-3 до 10 и более раз. Такую степень

сжатия невозможно реализовать при использовании алгоритмов сжатия без потерь. Поэтому, несмотря на то, что алгоритмы сжатия без потерь используются в современных космических системах (там, где потери радиометрической точности являются недопустимыми), наиболее активно развиваются и являются востребованными алгоритмы сжатия изображений с потерями. В настоящее время для этих целей, как правило, применяются алгоритмы 4 семейств, отличающихся по эффективности и по сложности реализации: ДИКМ (дифференциально-импульсная кодовая модуляция), ИСИ (иерархическая сеточная интерполяция), JPEG и алгоритмы на основе вейвлет-преобразования.



4. Рисунок. Схема реализации по сложностям алгоритмов сжатие изображение

Алгоритмы семейства ДИКМ отличаются простотой аппаратной реализации на ПЛИС и строгим постоянством выходного информационного потока, однако качество сжатого изображения сохраняется на высоком уровне только при степени сжатия 4 бита на пиксел. При степени сжатия до 2 бита на пиксел деградация изображения становится очень большой, но еще

приемлемой для некоторых применений. Сжатие до 1 бита на пиксел искажает изображение до неприемлемого уровня.

Алгоритм JPEG и его аналоги получили широкое распространение в компьютерной технике. Они отличаются умеренной сложностью аппаратной реализации и позволяют регулировать степень сжатия в широких пределах – до 0,5-1 бита на пиксел, причем при степени сжатия 1 бит на пиксел качество изображения оказывается достаточно высоким. К их недостаткам следует отнести непостоянство выходного информационного потока, которое требует использования буферной памяти.

Алгоритм ИСИ, по замыслу его разработчиков, должен был сочетать степень сжатия JPEG и простоту аппаратной реализации ДИКМ. На практике, однако, оказалось, что сложность реализации этого алгоритма сравнима со сложностью реализации JPEG. Степень сжатия, достижимая при использовании алгоритма ИСИ, близка к таковой для JPEG, однако алгоритм вносит специфические точечные искажения в сжатое изображение, с которыми можно бороться, уменьшая степень сжатия.

Алгоритмы на основе вейвлет – преобразования являются наиболее эффективными алгоритмами, однако и сложность их реализации максимальна. Правда, следует отметить, что они так же, как и ДИКМ, могут обеспечивать постоянство выходного информационного потока.

Критерии качества изображений. При сравнении алгоритмов сжатия изображений с потерями существует проблема, заключающаяся в том, что отсутствует критерий оценки качества сжатого изображения. На практике используются 4 критерия:

- Метод экспертных оценок
- Среднеквадратичное отклонение
- Максимальное отклонение
- Отношение "сигнал/шум".

Пусть x(i, j) – исходное изображение,

y(i, j) -сжатое изображение,

n – число пикселов в изображении.

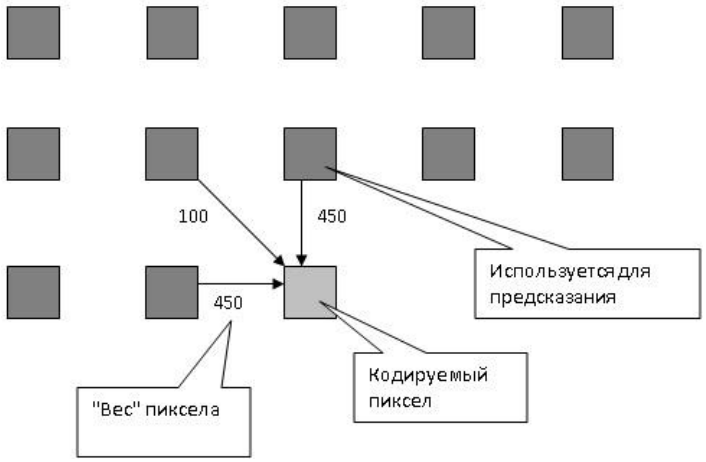
Тогда:

Среднеквадратичное отклонение $STD = \frac{\sqrt{\sum\sum (x(i,j) - y(i,j))^2}}{n}$.

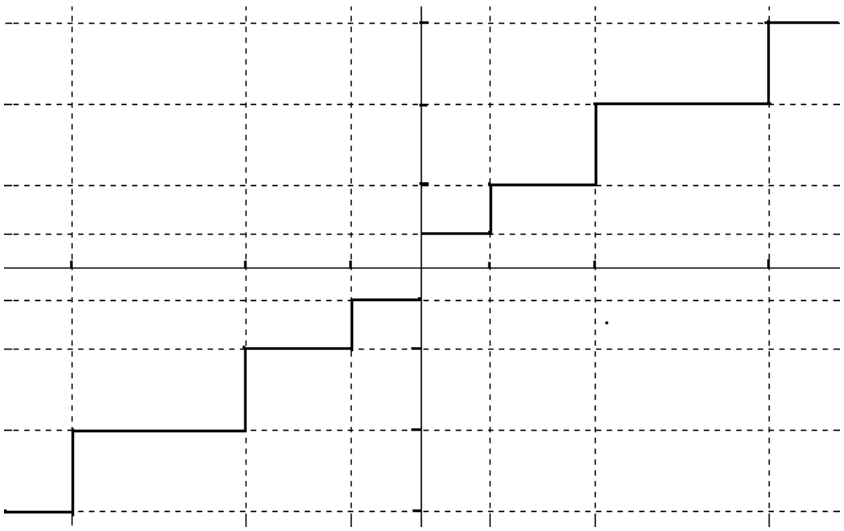
Максимальное отклонение $MAX = \max|x(i,j) - y(i,j)|$.

Отношение "сигнал/шум" $SNR = 20 \cdot \lg\left(\frac{255}{STD}\right)$.

Алгоритм Адаптивная дифференциальная импульсно-кодовая модуляция. Основан на предсказании сигнала в кодируемом пикселе и на использовании одной или нескольких неравномерных шкал квантования.



5. Рисунок. Предсказании сигнала в кодируемом пикселе



6. Рисунок. Пример неравномерной шкалы квантования.

В некоторых модификациях алгоритма используются несколько шкал квантования. В качестве примера рассмотрим 2 шкалы – "грубую" и "тонкую", содержащие одинаковое количество ступеней. В начале используется грубая шкала квантования. Если для кодирования пиксела использовался код "в глубине шкалы" (не крайнее значение), то для кодирования следующего пиксела используется "тонкая" шкала квантования. Если для кодирования очередного пиксела использовался код "на краю" шкалы, но для кодирования следующего пиксела используется "грубая" шкала. Этот подход позволяет заметно увеличить качество сжатого изображения.

Алгоритм JPEG. JPEG - один из новых и достаточно мощных алгоритмов. Практически, он является стандартом де-факто для полноцветных изображений. Оперирует алгоритм областями 8x8, на которых яркость и цвет меняются сравнительно плавно. Вследствие этого при разложении матрицы такой в двойной ряд по косинусам (см. формулы ниже) значимыми оказываются только первые коэффициенты. Таким образом, сжатие в JPEG осуществляется за счет плавности изменения цветов в изображении.

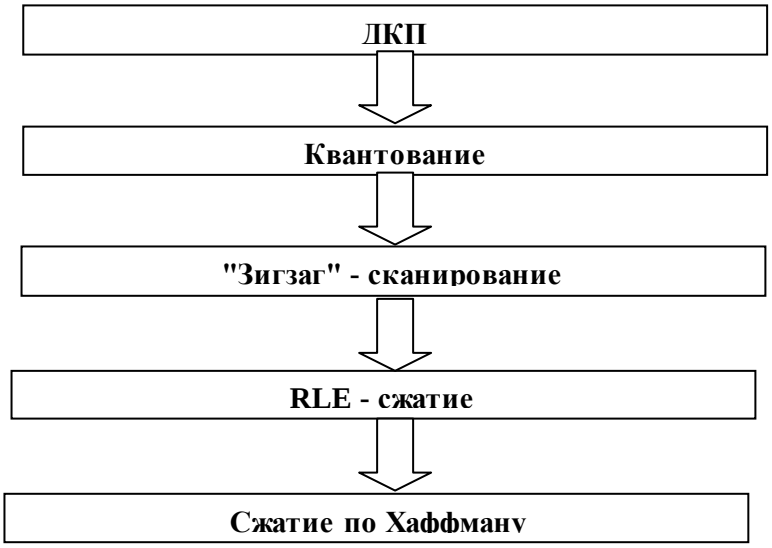
Алгоритм разработан группой экспертов в области фотографии специально для сжатия 24-битовых изображений. **JPEG - Joint Photographic Expert Group** - подразделение в рамках ISO - Международной организации по стандартизации. Название алгоритма читается как [jei'peg]. В целом алгоритм основан на дискретном косинусном преобразовании (в дальнейшем - ДКП), применяемом к матрице изображения для получения некоторой новой матрицы коэффициентов. Для получения исходного изображения применяется обратное преобразование.

ДКП раскладывает изображение по амплитудам некоторых частот. Таким образом, при преобразовании мы получаем матрицу, в которой коэффициенты либо близки, либо равны нулю. Кроме того, благодаря

несовершенству человеческого зрения можно аппроксимировать коэффициенты более грубо без заметной потери качества изображения.

Для этого используется квантование коэффициентов. В самом простом случае это арифметический побитовый сдвиг вправо. При этом преобразовании теряется часть информации, но может достигаться большая степень сжатия.

Как работает алгоритм JPEG. Итак, рассмотрим алгоритм подробнее (рис. 4). Пусть мы сжимаем 8 - битовое изображение.



7. Рисунок. Схема работа алгоритма JPEG

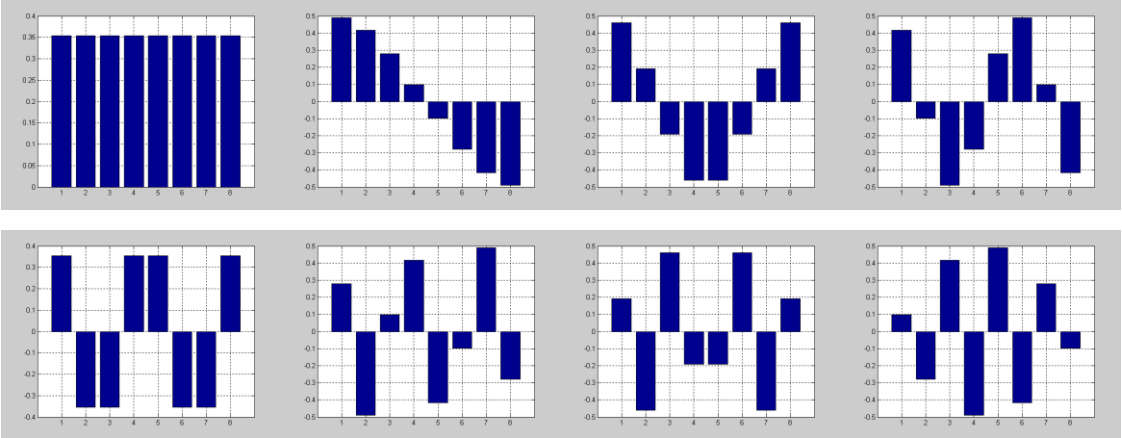
Шаг 1. Разбиваем исходное изображение на матрицы 8x8.

Шаг 2. В упрощенном виде ДКП при n=8 можно представить так:

$$Y(u,v) = \frac{1}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} C(i,u)\cdot C(j,v)\cdot y(i,j) \qquad (11)$$

где $$C(i,u) = A(u)\cdot\cos\left(\frac{\pi\cdot u\cdot(2i+1)}{2\cdot n}\right) \qquad (12)$$

$$A(u) = \begin{cases} \frac{1}{\sqrt{2}} & u = 0 \\ 1 & u \neq 0 \end{cases} \qquad (13)$$



8. Рисунок. Базисные функции одномерного ДКП.

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

Шаг 4. Производим квантование. В принципе это просто деление рабочей матрицы на матрицу квантования поэлементно.

$$Yq(u,v) = \left[\frac{Y(u,v)}{q(u,v)} \right] \qquad (14)$$

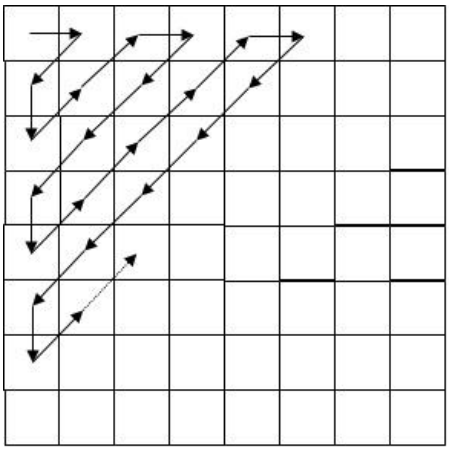
На этом шаге осуществляется управление степенью сжатия, и происходят самые большие потери. Понятно, что, задавая матрицу квантования с большими коэффициентами, мы получим больше нулей и, следовательно, большую степень сжатия.

В стандарт JPEG включены рекомендованные матрицы квантования, построенные опытным путем. Матрицы для большей или меньшей степени сжатия получают путем умножения исходной матрицы на некоторое число A.

С квантованием связаны и специфические эффекты алгоритма. При больших значениях коэффициента A потери в низких частотах мс быть настолько велики, что изображение распадется на квадраты 8x8. Потери в

высоких частотах могут проявиться в так называемом эффекте Гиббса, когда вокруг контуров с резким переходом цвета образуется своеобразный "нимб".

Шаг 5. Переводим матрицу 8x8 в 64-элементный вектор при помощи "зигзаг"-сканирования, т. е. берем элементы с индексами (0,0), (0,1), (1,0), (2,0).



9. Рисунок. "Зигзаг" – сканирование.

Таким образом, в начале вектора мы получаем коэффициенты матрицы, соответствующие низким частотам, а в конце - высоким.

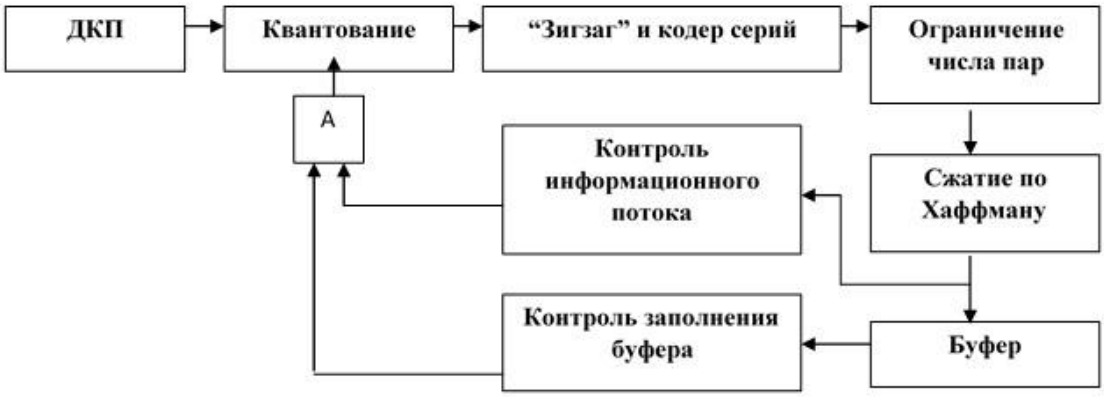
Шаг 6. Свертываем вектор с помощью алгоритма группового кодирования. При этом получаем пары типа <пропустить, число>, где "пропустить" является счетчиком пропускаемых нулей, а "число" - значение, которое необходимо поставить в следующую ячейку. Так, вектор 42 3 0 0 0 2 0 0 0 0 1 ... будет свернут в пары (0,42) (0,3) (3,-2) (4,1).

Шаг 7. Свертываем получившиеся пары кодированием по Хаффману с фиксированной таблицей.

Отрицательными сторонами алгоритма является то, что:

- При повышении степени сжатия изображение распадается на отдельные квадраты (8x8). Это связано с тем, что происходят большие потеря в низких частотах при квантовании и восстановить исходные данные становится невозможно.
- Проявляется эффект Гиббса - ореолы по границам резких переходов цветов.

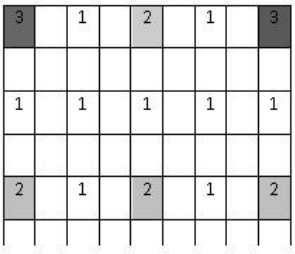
Для обеспечения постоянства выходного информационного потока на выходе блока аппаратного устанавливается специальная буферная память небольшой емкости. Входящая информация разбивается на блоки (обычно по 8 строк), и каждый блок сжимается по отдельности. Объем сжатого блока сравнивается с заданным, и, если полученный объем больше заданного, происходит увеличение матрицы квантования, что приводит к увеличению степени сжатия в А раз. Подобный процесс, в сочетании с контролем степени заполненности буфера (при заполнении буфера на 50-75% степень сжатия увеличивается) приводит к тому, что на выходе буфера получается постоянный информационный поток.



10. Рисунок. «Зигзаг» – сканирование.

Алгоритм ИСИ. Метод иерархической сеточной интерполяции основан на идее сокращения избыточности входных данных за счет использования прореженного изображения для аппроксимации промежуточных отсчетов.

На рисунке показано расположение отсчетов для 4 иерархических уровней для на изображении размера 9x9.



11. Рисунок. Размещение отсчетов иерархических уровней на изображении

Анализируя схему, легко видеть, что r-й уровень представляет собой сетку с шагом 2r, из которой исключены отсчеты, лежащие в узлах сеток с шагами 2r+1, 2r+2 ... 2R-1. Таким образом, данное представление не является избыточным и не увеличивает объем входных данных при сжатии. При этом иерархический уровень с большим шагом используется для аппроксимации уровня с меньшим шагом. Очевидно, что при таком подходе для сжатия каждого уровня представления изображения применяется один и тот же алгоритм, поэтому можно ограничиться описанием последовательности процедур обработки для уровня $X^{(r)}$ произвольным фиксированным номером r.

Вычисление предсказания значения для кодируемого пиксела осуществляется путем интерполяции по 2 или 4 соседним пикселам верхнего уровня иерархии. Квантование разностного сигнала осуществляется при помощи неравномерной шкалы или шкал, различных для каждого уровня.

Метод усеченного блочного кодирования (УБК). Название метода отражает тот факт, что изображение разбивается на небольшие прямоугольные куски одинакового размера, называемые блоками. Этот метод в отличие от большинства других подстраивает параметры кодирования не под некоторую усредненную характеристику всего изображения, а под локальные особенности в пределах каждого блока. Это позволяет сохранить мелкие детали изображений. Метод не приводит к размыванию границ, что характерно для некоторых других алгоритмов. Метод УБК сопоставим с большинством других методов по эффективности сжатия данных и по объему вычислений, требуемых для кодирования, но не имеет конкурентов по простоте декодирования.

Базовый алгоритм УБК строится следующим образом. Изображение, представленное $M \times N$ - матрицей $\|b_{ij}\|$ яркостей пикселей, разбивается на

небольшие прямоугольные блоки $m \times n$ элементов. Каждый такой блок обрабатывается независимо от других, поэтому опишем алгоритм обработки одного блока.

Обработка блока начинается с вычисления порога и двух уровней квантования (описанных ниже), затем проводится квантование блока на два уровня, после чего следует упаковка проквантованного блока. Для определения уровней квантования сначала вычисляются два первых выборочных момента - среднее значение C и средний квадрат E:

$$C = \frac{1}{m \times n} \sum_i \sum_i b_{ij} \quad , \quad E = \frac{1}{m \times n} \sum_i \sum_j b_{ij}^2$$

(где суммируются элементы изображения в пределах блока) и дисперсия

$$\sigma^2 = E - c^2 .$$

Пороговая величина квантователя d полагается равной среднему C. Верхний a и нижний b уровни квантования вычисляются по следующим формулам:

$$a = C - \sigma\sqrt{q/(p-q)} \quad , \quad b = C + \sigma\sqrt{(p-q)/q} \quad ,$$

где $p = m \times n$ - число элементов блока, q - число элементов блока, превышающих порог d.

Квантование проводится по правилу:

$$s_{ij} = \begin{cases} a, если\ b_{ij} < d \\ b, если\ b_{ij} \geq d \end{cases} ,$$

где s_{ij} - элементы изображения после квантования.

После квантования получается блок, содержащий только уровни a и b. Нетрудно показать, что среднее значение и средний квадрат исходного и проквантованного блоков совпадают. Практически для удобства последующей упаковки вместо a записывается нуль, вместо b - единица. Уровни a и b записываются отдельно.

Упаковка состоит в том, что блок, содержащий только нули и единицы, интерпретируется как двоичное число, имеющее $m \times n$ разрядов. Восстановление закодированного изображения также проводится поблочно и состоит в распаковке и обратной подстановке.

Из алгоритма видно, что степень сжатия непосредственно зависит от размеров блока. Наиболее удовлетворительные результаты, как по степени сжатия, так и по качеству восстановленного изображения были получены при использовании блоков размером 4×4 (см. [3]).

Описанный выше способ определения порога и уровней квантования не является единственным. Существует ряд других критериев. Важно, чтобы критерий соответствовал целям последующей обработки изображения и ее конкретным особенностям.

JPEG - один из самых распространенных и достаточно мощных алгоритмов, представляет собой метод сжатия изображений, реализуемый различными способами. Работает он как на черно-белых, так и на полноцветных изображениях.

Коэффициент архивации в JPEG может изменяться в пределах от 2 до 200 раз. Как и у любого другого алгоритма сжатия с потерями, у JPEG свои особенности. Наиболее известны ‘эффект Гиббса’ и дробление изображения на квадраты. Первый проявляется около резких границ предмета, образуя вокруг своеобразный ‘ореол’. Разбиение на квадраты происходит, когда задается слишком большой коэффициент сжатия для данной картинки. Тем не менее, не смотря на эти недостатки, для архивации изображений, предназначенных для просмотра человеком, он на данный момент является лучшим.

Широкое применение JPEG сдерживается, пожалуй, лишь тем, что он оперирует 24-битными изображениями. Поэтому для того, чтобы с приемлемым качеством посмотреть картинку на обычном мониторе в 256-цветной палитре, требуется применение соответствующих алгоритмов и,

следовательно, определенное время. В приложениях, ориентированных на придирчивого пользователя, таких, например, как игры, подобные задержки неприемлемы. Кроме того, если имеющиеся изображения, допустим, в 8-битном формате GIF перевести в 24-битный JPEG, а потом обратно в GIF для просмотра, то потеря качества произойдет дважды при обоих преобразованиях. Тем не менее, выигрыш в размерах архивов зачастую настолько велик (3-20 раз), а потери качества настолько малы, что хранение изображений в JPEG оказывается очень эффективным.

Преобразование цветов RGB в цвета YUV. В сжатии JPEG применяется система цветов YUV. Разделение данных RGB на данные YUV позволяет программе сжатия уделять больше внимание данным о яркости (Y), чем данным о цвете (UV). Этот процесс называется подвыборкой, так как три компоненты выбираются с различной частотой. Так метод подвыборки, который называется YUV411, на каждую выборку цвета делает четыре выборки данных о яркости. Например, если рисунок не подвергался подвыборке, то величины YUV встречаются в нем с одинаковой частотой. При использовании подвыборки YUV411 на шесть значений выборки обработанного файла приходится двенадцать значений выборки исходного файла. Таким образом, подвыборка данных сразу уменьшает размер файла изображения.

Метод сжатия JPEG. Процесс сжатия по схеме JPEG состоит из трех шагов (не считая подвыборку). Первый шаг - это запись изменения значений пикселей в виде изменения частот: как быстро меняются яркость и цвет пикселей. Второй шаг - группировка этих отдельных изменений частот по средним значениям (первый этап сжатия). И третий этап - сжатие этих усредненных данных с помощью модифицированного алгоритма кодирования Хаффмана.

Изменение частоты. JPEG определяет изменение частоты данных с помощью дискретного преобразования Фурье (ДПФ), которое применяется

для каждой пиксельной компоненты в выбранной области пикселей. Например, выбирается группа из 8×8 пикселей, и к ней применяется преобразование ДПФ: сначала к величинам красных компонент во всей группе, затем зеленых, и, наконец, синих.

Вместо действительных значений пикселей величины ДПФ хранят скорость изменения интенсивности от пикселя к пикселю. На самом деле данных о частоте получается больше, чем исходных пиксельных данных, но следующие два шага это устраняют.

Усреднение. После вычисления с помощью ДПФ значений изменения частоты эти величины усредняются в соответствии с плавающей шкалой относительной важности. Это значит, что изменения частоты, которые меньше влияют на общий вид изображения (например, быстрые изменения частоты), усредняются больше других значений. Именно на этой стадии сжатия изображения происходят потери, так как величина применяемого усреднения может регулироваться.

Сжатие методом Хаффмана. Окончательно усредненные данные о частоте сжимаются с помощью модифицированного алгоритма кодирования Хаффмана, который особенно эффективен для этого типа данных, так как строит таблицы кодов, базирующиеся на частоте повторения величин.

4. Изучение различия между форматом и алгоритмом

Напоследок несколько замечаний относительно разницы в терминологии, путаницы при сравнении рейтингов алгоритмов и т.п.

Посмотрите на краткий перечень форматов, достаточно часто используемых на PC, Apple и UNIX платформах: ADEX, Alpha Microsystems BMP, Autologic, AVHRR, Binary Information File (BIF), Calcomp CCRF, CALS, Core IDC, Cubicomp PictureMaker, Dr. Halo CUT, Encapsulated PostScript, ER Mapper Raster, Erdas LAN/GIS, First Publisher ART, GEM VDI Image File, GIF, GOES, Hitachi Raster Format, PCL, RTL, HP-48sx Graphic Object (GROB), HSI

JPEG, HSI Raw, IFF/ILBM, Img Software Set, Jovian VI, JPEG/JFIF, Lumena CEL, Macintosh PICT/PICT2, MacPaint, MTV Ray Tracer Format, OS/2 Bitmap, PCPAINT/Pictor Page Format, PCX, PDS, Portable BitMap (PBM), QDV, QRT Raw, RIX, Scodl, Silicon Graphics Image, SPOT Image, Stork, Sun Icon, Sun Raster, Targa, TIFF, Utah Raster Toolkit Format, VITec, Vivid Format, Windows Bitmap, WordPerfect Graphic File, XBM, XPM, XWD.

Единственным совпадением оказывается JPEG, а это, согласитесь, не повод, чтобы повсеместно использовать слова 'формат' и 'алгоритм компрессии' как синонимы (что, увы, я постоянно наблюдаю).

Между этими двумя множествами нет взаимно однозначного соответствия. Так, различные модификации алгоритма RLE реализованы в огромном количестве форматов. В том числе в TIFF, BMP, PCX. И, если в определенном формате какой-либо файл занимает много места, это не означает, что плох соответствующий алгоритм компрессии. Это означат, зачастую лишь то, что реализация алгоритма, использованная в этом формате, дает для данного изображения плохие результаты. Не более того.

В то же время многие современные форматы поддерживают запись с использованием нескольких алгоритмов архивации либо без использования архивации. Например, формат TIFF 6.0 может сохранять изображения с использованием алгоритмов RLE-PackBits, RLE-CCITT, LZW, Хаффмана с фиксированной таблицей, JPEG, а может сохранять изображение без архивации. Аналогично форматы BMP и TGA позволяют сохранять файлы как с использованием алгоритма компрессии RLE (разных модификаций!), так и без использования оной.

Вывод 1: Для многих форматов, говоря о размере файлов, необходимоуказывать, использовался ли алгоритм компрессии и если использовался, то какой.

Можно пополнить перечень ситуаций некорректного сравнения алгоритмов. При сохранении абсолютно черного изображения в формате 1000x1000x256 цветов в формате BMP без компрессии мы получаем, как и положено, файл размером чуть более 1000000 байт, а при сохранении с компрессией RLE, можно получить файл размером 64 байта. Это был бы превосходный результат - сжатие в 15 000 раз(!), если бы к нему имела отношение компрессия. Дело в том, что данный файл в 64 байта состоит только из заголовка изображения, в котором указаны все его данные. Несмотря на то, что такая короткая запись изображения стала возможна именно благодаря особенности реализации RLE в BMP, еще раз подчеркну, что в данном случае алгоритм компрессии даже не применялся. И то, что для абсолютно черного изображения 4000x4000x256 мы получаем коэффициент компрессии 250 тысяч раз, совсем не повод для продолжительных эмоций по поводу эффективности RLE. Кстати - данный результат возможен лишь при определенном положении цветов в палитре и далеко не на всех программах, которые умеют записывать BMP с архивацией RLE (однако все стандартные средства, в т.ч. средства системы Windows, читают такой сжатый файл нормально).

Всегда полезно помнить, что на размер файла оказывают существенное влияние большое количество параметров (вариант реализации алгоритма, параметры алгоритма (как внутренние, так и задаваемые пользователем), порядок цветов в палитре и многое другое). Например, для абсолютно черного изображения 1000x1000x256 градаций серого в формате JPEG с помощью одной программы при различных параметрах всегда получался файл примерно в 7 килобайт. В то же время, меняя опции в другой программе, я получил файлы размером от 4 до 68 Кб (всего-то на порядок разницы). При этом декомпрессированное изображение для всех файлов было одинаковым - абсолютно черный квадрат (яркость 0 для всех точек изображения).

Дело в том, что даже для простых форматов одно и то же изображение в одном и том же формате с использованием одного и того же алгоритма архивации можно записать в файл несколькими корректными способами. Для сложных форматов и алгоритмов архивации возникают ситуации, когда многие программы сохраняют изображения разными способами. Такая ситуация, например, сложилась с форматом TIFF (в силу его большой гибкости). Долгое время по-разному сохраняли изображения в формат JPEG, поскольку соответствующая группа ISO (Международной Организации по Стандартизации) подготовила только стандарт алгоритма, но не стандарт формата. Сделано так было для того, чтобы не вызывать 'войны форматов'. Абсолютно противоположное положение сейчас с фрактальной компрессией, поскольку есть стандарт «де-факто» на сохранение фрактальных коэффициентов в файл (стандарт формата), но алгоритм их нахождения (быстрого нахождения!) является технологической тайной создателей программ-компрессоров. В результате для вполне стандартной программы-декомпрессора могут быть подготовлены файлы с коэффициентами, существенно различающиеся как по размеру, так и по качеству получающегося изображения.

Приведенные примеры показывают, что встречаются ситуации, когда алгоритмы записи изображения в файл в различных программах различаются. Однако гораздо чаще причиной разницы файлов являются разные параметры алгоритма. Как уже говорилось, многие алгоритмы позволяют в известных пределах менять свои параметры, но не все программы позволяют это делать пользователю.

Вывод 2: Если вы не умеете пользоваться программами архивации или пользуетесь программами, в которых 'для простоты использования' убрано управление параметрами алгоритма - не удивляйтесь, почему для отличного алгоритма компрессии в результате получаются большие файлы.

Выводы по первой главе.

Первая глава магистерской работы была посвящена изучению и анализу общих положений алгоритмов сжатия изображений. Приведено представление изображений в качестве конечных данных и статистическая избыточность изображений. Изложена вся информация о существующих классах изображений. Проведен сравнительный анализ сжатия изображений. Изучено различие между форматом и алгоритмом.

II. ГЛАВА. ИССЛЕДОВАНИЕ МЕТОДОВ ФРАКТАЛЬНОГО СЖАТИЯ ИЗОБРАЖЕНИЙ

1. Обоснование метода фрактального сжатия

В декабре 1992 года, перед самым Рождеством, компания Microsoft выпустила свой новый компакт-диск Microsoft Encarta. С тех пор эта мультимедиа-энциклопедия, содержащая информацию о животных, цветах, деревьях и живописных местах, не покидает списки наиболее популярных энциклопедий на компакт-дисках. В недавнем опросе Microsoft Encarta опять заняла первое место, опередив ближайшего конкурента - Комптоновскую мультимедиа-энциклопедию. Причина подобной популярности кроется в удобстве использования, высоком качестве статей и, главное, в большом количестве материалов. На диск записано 7 часов звука, 100 анимационных роликов, примерно 800 масштабируемых карт, а также 7000 качественных фотографий. И все это - на одном диске! Напомним, что обычный компакт-диск в 650 Мбайт без использования компрессии может содержать либо 56 минут качественного звука, либо 1 час видео разрешения с разрешением 320x200 в формате MPEG-1, либо 700 полноцветных изображений размером 640x480.

Чтобы разместить больше информации, необходимы достаточно эффективные алгоритмы архивации. Мы не будем останавливаться на методах архивации для видео и звука. Речь пойдет о новом перспективном алгоритме - фрактальном сжатии графической информации.

Когда в 1991 году впервые была опубликована информация о возможностях фрактального сжатия, она наделала много шуму. Майкл Барнсли, один из разработчиков алгоритма, утверждал, что разработан способ нахождения коэффициентов фрактала, сколь угодно близкого к исходной картинке.

Фракталы, эти красивые образы динамических систем, ранее использовались в машинной графике в основном для построения

изображений неба, листьев, гор, травы. Красивое и, что важнее, достоверно имитирующее природный объект изображение могло быть задано всего несколькими коэффициентами. Неудивительно, что идея использовать фракталы при сжатии возникала и раньше, но считалось практически невозможным построить соответствующий алгоритм, который подбирал бы коэффициенты за приемлемое время.

Итак, в 1991 году такой алгоритм был найден. Кроме того, в дальнейших его статьях декларировался ряд уникальных возможностей новой технологии. Так, фрактальный архиватор позволяет, например, при распаковке произвольно менять разрешение (размеры) изображения без появления эффекта зернистости. Более того, он распаковывает гораздо быстрее, чем ближайший конкурент JPEG, и не только статическую графику, но и видео. В качестве примера приводилась программа, показывающая на машине с процессором i386/33 МГц цветной видеофильм с частотой 20 кадров в секунду без всякой аппаратной поддержки. В отличие от JPEG, в алгоритм изначально заложена возможность управлять степенью потерь на участках с максимальными потерями качества. Коэффициент сжатия изображений в целом примерно как у JPEG, но на некоторых реальных картинках достигалось сжатие в 10000 (!) раз.

Звучит это более чем внушительно, поэтому необходимо спокойно разобраться с преимуществами, которые обещает фрактальная компрессия, а также с возможными неприятными сторонами этого алгоритма.

История фрактального сжатия. Рождение фрактальной геометрии обычно связывают с выходом в 1977 году книги Б. Мандельброта «Фрактальная геометрия природы». Одна из основных идей книги заключалась в том, что средствами традиционной геометрии (то есть используя линии и поверхности), чрезвычайно сложно представить природные объекты. Фрактальная геометрия задает их очень просто.

В 1981 году Джон Хатчинсон опубликовал статью "Фракталы и само подобие", в которой была представлена теория построения фракталов с помощью системы итерируемых функций (IFS, Iterated Function System).

Четыре года спустя появилась статья Майкла Барнсли и Стефана Демко, в которой приводилась уже достаточно стройная теория IFS. В 1987 году Барнсли основал Iterated Systems, компанию, основной деятельностью которой является создание новых алгоритмов и ПО с использованием фракталов.

Всего через год, в 1988 году, он выпустил фундаментальный труд «Фракталы повсюду». Помимо описания IFS, в ней был получен результат, известный сейчас как Collage Theorem, который лежит в основе математического обоснования идеи фрактальной компрессии.

Если построение изображений с помощью фрактальной математики можно назвать прямой задачей, то построение по изображению IFS - это обратная задача. Довольно долго она считалась неразрешимой, однако Барнсли, используя Collage Theorem, построил соответствующий алгоритм. (В 1990 и 1991 годах эта идея была защищена патентами.) Если коэффициенты занимают меньше места, чем исходное изображение, то алгоритм является алгоритмом архивации.

Первая статья об успехах Барнсли в области компрессии появилась в журнале BYTE в январе 1988 года. В ней не описывалось решение обратной задачи, но приводилось несколько изображений, сжатых с коэффициентом 1:10000, что было совершенно ошеломительно. Но практически сразу было отмечено, что несмотря на броские названия («Темный лес», «Побережье Монтере», «Поле подсолнухов») изображения в действительности имели искусственную природу. Это, вызвало массу скептических замечаний, подогреваемых еще и заявлением Барнсли о том, что "среднее изображение требует для сжатия порядка 100 часов работы на мощной двухпроцессорной рабочей станции, причем с участием человека".

Отношение к новому методу изменилось в 1992 году, когда Арнауд Джеквин, один из сотрудников Барнсли, при защите диссертации описал практический алгоритм и опубликовал его. Этот алгоритм был крайне медленным и не претендовал на компрессию в 10000 раз (полноцветное 24-разрядное изображение с его помощью могло быть сжато без существенных потерь с коэффициентом 1:8 - 1:50); но его несомненным достоинством было то, что вмешательство человека удалось полностью исключить. Сегодня все известные программы фрактальной компрессии базируются на алгоритме Джеквина. В 1993 году вышел первый коммерческий продукт компании Iterated Systems. Ему было посвящено достаточно много публикаций, но о коммерческом успехе речь не шла, продукт был достаточно "сырой", компания не предпринимала никаких рекламных шагов, и приобрести программу было тяжело.

В 1994 году Ювал Фишер были предоставил во всеобщее пользование исходные тексты исследовательской программы, в которой использовалось разложение изображения в квадродерево и были реализованы алгоритмы оптимизации поиска. Позднее появилось еще несколько исследовательских проектов, которые в качестве начального варианта программы использовали программу Фишера.

В июле 1995 года в Тронхейме (Швеция) состоялась первая школа-конференция, посвященная фрактальной компрессии. Таким образом, многие важные события в области фрактальной компрессии произошли за последние три года: алгоритм только-только начинает развиваться.

Фрактальная архивация основана на том, что с помощью коэффициентов системы итерируемых функций изображение представляется в более компактной форме. Прежде чем рассматривать процесс архивации, разберем, как IFS строит изображение.

Строго говоря, IFS - это набор трехмерных аффинных преобразований, переводящих одно изображение в другое. Преобразованию

подвергаются точки в трехмерном пространстве (x координата, y координата, яркость).

Наиболее наглядно этот процесс продемонстрировал сам Барнсли в своей книге "Фрактальное сжатие изображения". В ней введено понятие Фотокопировальной Машины, состоящей из экрана, на котором изображена исходная картинка, и системы линз, проецирующих изображение на другой экран. Каждая линза проецирует часть исходного изображения. Расставляя линзы и меняя их характеристики, можно управлять получаемым изображением. На линзы накладывается требование они должны уменьшать в размерах проектируемую часть изображения. Кроме того, они могут менять яркость фрагмента и проецируют не круги, а области с произвольной границей.

Одна шаг Машины состоит в построении с помощью проецирования по исходному изображению нового. Утверждается, что на некотором шаге изображение перестанет изменяться. Оно будет зависеть только от расположения и характеристик линз и не будет зависеть от исходной картинки. Это изображение называется неподвижной точкой или аттрактором данной IFS. Collage Theorem гарантирует наличие ровно одной неподвижной точки для каждой IFS. Поскольку отображение линз является сжимающим, каждая линза в явном виде задает самоподобные области в нашем изображении. Благодаря самоподобию мы получаем сложную структуру изображения при любом увеличении.

Наиболее известны два изображения, полученных с помощью IFS треугольник Серпинского и папоротник Барнсли Первое задается тремя, а второе - питью аффинными преобразованиями (или, в нашей терминологии, линзами). Каждое преобразование задается буквально считанными байтами, в то время, как изображение, построенное с их помощью, может занимать и несколько мегабайт.

Становится понятно, как работает архиватор, и почему ему требуется так много времени. Фактически, фрактальная компрессия - это поиск самоподобных областей в изображении и определение для них параметров аффинных преобразований.

В худшем случае, если не будет применяться оптимизирующий алгоритм, потребуется перебор и сравнение всех возможных фрагментов изображения разного размера. Даже для небольших изображений при учете дискретности мы получим астрономическое число перебираемых вариантов. Даже резкое сужение классов преобразований, например, за счет масштабирования только в определенное число раз, не позволит добиться приемлемого времени. Кроме того, при этом теряется качество изображения. Подавляющее большинство исследований в области фрактальной компрессии сейчас направлены на уменьшение времени архивации, необходимого для получения качественного изображения.

Оценка потерь и способы их регулирования. До сих пор мы не затронули несколько важных вопросов. Например, что делать, если алгоритм не может подобрать для какого-либо фрагмента изображения подобный ему? Достаточно очевидное решение - разбить этот фрагмент на более мелкие и попытаться поискать для них. Однако понятно, что процедуру эту нельзя повторять до бесконечности, иначе количество необходимых преобразований станет так велико, что алгоритм перестанет быть алгоритмом компрессии. Следовательно, допускаются потери в какой-то части изображения.

Для фрактального алгоритма компрессии, как и для других алгоритмов сжатия с потерями, очень важны механизмы, с помощью которых можно будет регулировать степень сжатия и степень потерь. К настоящему времени разработан достаточно большой набор таких методов. Во-первых, можно ограничить количество преобразований, заведомо обеспечив степень сжатия не ниже фиксированной величины. Во-вторых, можно потребовать, чтобы в ситуации, когда разница между обрабатываемым фрагментом и

наилучшим его приближением будет выше определенного порогового значения, этот фрагмент дробился обязательно (для него обязательно заводится несколько линз). В-третьих, можно запретить дробить фрагменты размером меньше, допустим, четырех точек. Изменяя пороговые значения и приоритет этих условий, можно очень гибко управлять коэффициентом компрессии изображения: от побитного соответствия, до любой степени сжатия.

Возможности масштабирования. Итак, мы выяснили, что IFS задает фрактальную структуру, сколь угодно близкую к нашему изображению. При внимательном рассмотрении процесса построения изображения с ее помощью становится понятно, что восстанавливаемое изображение может иметь любое (!) разрешение. В самом деле, возвращаясь к аналогии с Фотокопировальной Машиной, можно сказать, что нам не важно до какой сетки растра будет огрубляться установившееся неподвижное изображение. Ведь Машина работает вообще с непрерывными экранами.

На этапе архивации проводится распознавание изображения, и в виде коэффициентов хранится уже не растровая информация, а информация о структуре самого изображения. Именно это и позволяет при развертывании увеличивать его в несколько раз. Особенно впечатляют примеры, в которых при увеличении изображений природных объектов проявляются новые детали, действительно этим объектам присущие (например, когда при увеличении фотографии скалы она приобретает новые, более мелкие неровности).

Но не все так гладко, как может показаться. Если изображение однородно (на фотографии только скала), то при увеличении получаются отличные результаты, однако, если сжимать изображение натюрморта, то предсказать, какие новые фрактальные структуры возникнут, очень сложно. Впрочем, вдвое-втрое можно увеличить практически любое изображение, при архивации которого задавалась небольшая степень потерь.

Масштабирование - уникальная особенность, присущая фрактальной компрессии. Со временем ее, видимо, будут активно использовать как в специальных алгоритмах масштабирования, так и во многих приложениях. Действительно, этого требует концепция "приложение в окне". Было бы неплохо, если бы изображение, показываемое в окне 100x100, хорошо смотрелось при увеличении на полный экран - 1024x768.

Сравнение с JPEG. Сегодня наиболее распространенным алгоритмом архивации графики является JPEG. Сравним его с фрактальной компрессией.

Во-первых, заметим, что и тот, и другой алгоритм оперируют 8-битными (в градациях серого) и 24-битными полноцветными изображениями. Оба являются алгоритмами сжатия с потерями и обеспечивают близкие коэффициенты архивации. И у фрактального алгоритма, и у JPEG существует возможность увеличить степень сжатия за счет увеличения потерь. Кроме того, оба алгоритма очень хорошо распараллеливаются.

Различия начинаются, если мы рассмотрим время, необходимое алгоритмам для архивации/разархивации. Так, фрактальный алгоритм сжимает в сотни и даже в тысячи раз дольше, чем JPEG. Распаковка изображения, наоборот, произойдет в 5-10 раз быстрее. Поэтому, если изображение будет сжато только один раз, а передано по сети и распаковано множество раз, то выгодней использовать фрактальный алгоритм.

JPEG использует разложение изображения по косинусоидальным функциям, поэтому потери в нем (даже при заданных минимальных потерях) проявляются в волнах и ореолах на границе резких переходов цветов. Именно за этот эффект его не любят использовать при сжатии изображений, которые готовят для качественной печати: там этот эффект может стать очень заметен.

Фрактальный алгоритм избавлен от этого недостатка. Более того, при печати изображения каждый раз приходится выполнять операцию масштабирования, поскольку растр (или линиатура) печатающего устройства

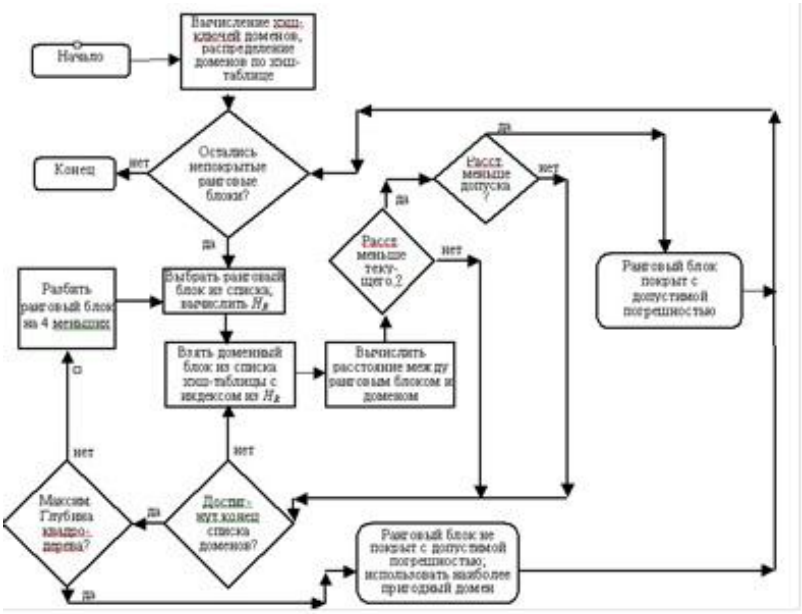
не совпадает с растром изображения. При преобразовании также может возникнуть несколько неприятных эффектов, с которыми можно бороться либо масштабируя изображение программно (для дешевых устройств печати типа обычных лазерных и струйных принтеров), либо снабжая устройство печати своим процессором, винчестером и набором программ обработки изображений (для дорогих фотонаборных автоматов). Как можно догадаться, при использовании фрактального алгоритма таких проблем практически не возникает.

Вытеснение JPEG фрактальным алгоритмом в повсеместном использовании произойдет еще не скоро (хотя бы в силу низкой скорости архивации последнего), однако в области приложений мультимедиа, в компьютерных играх его использование вполне оправдано.

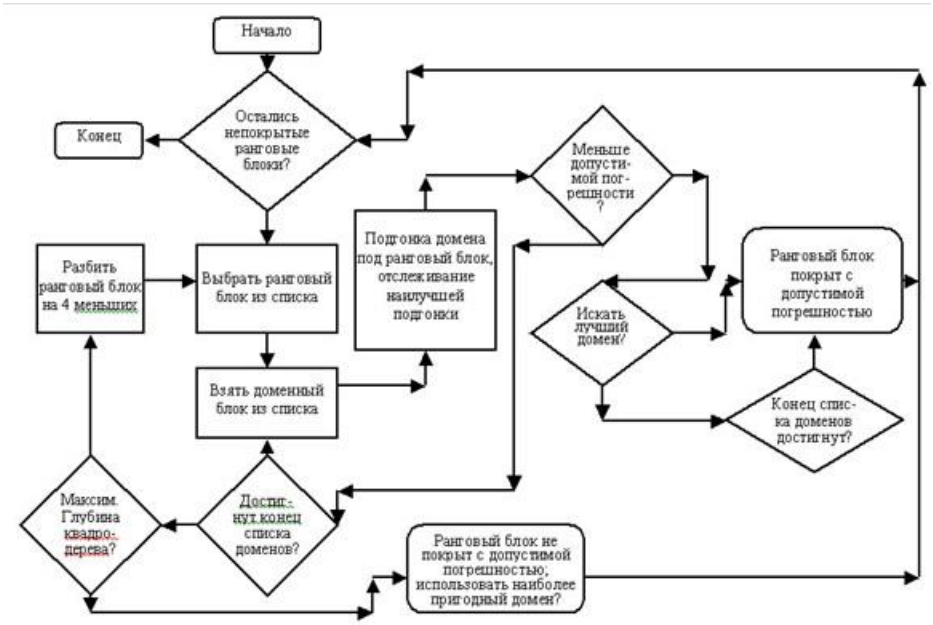
Возможности видеокомпрессии. Итак, одной из основных проблем, с которой пришлось столкнуться при построении алгоритма фрактальной компрессии, является поиск самоподобных участков в изображении. Это основная идея, благодаря которой осуществляется сжатие. Подобный метод можно применить и при архивации видео. Как правило, соседние кадры отличаются не сильно, и изменения между ними в основном, состоят в сдвиге, повороте или растяжении какой-либо части изображения. Таким образом, изменения между двумя кадрами можно задать небольшим количеством аффинных преобразований.

До недавнего времени такой подход рассматривался как утопический, ввиду чрезвычайно большого объема вычислений, требуемых при поиске соответствующих преобразований. Но достижения в области фрактальной архивации статической графики позволяют пересмотреть взгляды. Всего четыре года назад для архивации изображения с помощью фракталов требовалась мощная рабочая станция и многие часы работы. Сегодня с той же задачей справляется средний ПК всего за несколько минут. Найдены алгоритмы, существенно оптимизирующие процесс поиска аффинных

преобразований. При этом, по-прежнему рано говорить о повсеместном использовании фрактальной архивации для видео, поскольку даже если будет тратиться на один кадр 5 минут машинного времени (что совсем немного), то для архивации десятиминутного ролика (то есть 15 тысяч кадров) потребуется почти два месяца непрерывной работы.



2. Рисунок. Базовый алгоритм фрактального кодирования изображений



3. Рисунок. Алгоритм фрактального кодирования с использованием хэш – ключей

2. Алгоритмы Фишера для поиска фрактальной модели изображения

Идея алгоритма заключается в том, чтобы некоторым образом классифицировать D-блоки и R-блоки, а поиск близкого D-блока производить в том же классе, к которому относится ранговая область. Делается это следующим образом.

Исходные блоки разбиваются на четыре части. Для каждой из частей подсчитывается среднее значение A_i и дисперсия V_i пикселей. Далее блоки классифицируются по следующему принципу. Определим три базовых типа блоков

тип 1: $A_1 \geq A_2 \geq A_3 \leq A_4$,

тип 2: $A_1 \geq A_2 \geq A_4 \geq A_3$,

тип 3: $A_1 \geq A_4 \geq A_2 \geq A_3$.

Понятно, что любой блок при помощи соответствующего аффинного преобразования квадрата в квадрат можно привести к виду, соответствующему одному из указанных типов. После того, как мы зафиксировали три основных класса, блоки классифицируются по дисперсии. Таким образом, в каждом из трех классов появляются 24 подкласса, итого 72 класса. Поиск близкого к R-блоку D-блока производится перебором в соответствующем классе.

Генетический алгоритм представляет собой алгоритмический подход к решению экстремальных задач однокритериального выбора, основанный на моделировании основных факторов эволюционного развития популяции.

При использовании ГА для поиска оптимальных решений каждый элемент $x \in X$ пространства оптимизации должен быть представлен как вектор $b \in B$ из N символов двоичного алфавита $A = \{0,1\}$, где $B = AN$. Необходимо также, чтобы пространство оптимизации X состояло из конечного числа элементов.

Популяцией $\Pi = (\chi 1, \chi 2, \ldots, \chi M)$ численности M считается вектор пространства BM, координаты которого называются генотипами особей данной популяции.

Шагом ГА является переход от текущего поколения к следующему, т.е. получение новой популяции Π_{t+1} из Π_t. В построении очередной особи новой популяции участвуют операторы кроссинговера, мутации и случайный оператор отбора, Select: $BM \to \{1,\ldots,M\}$ действие которого состоит в выборе номера особи родителя при порождении очередного потомка.

Для определения ГА в необходимо задать оператор кроссинговера (скрещивания) Cross: $B \times B \to B \times B$ и оператор мутации Mut: $B \to B$.

Действие кроссинговера $(\chi', \tau') = Cross(\chi, \tau)$ заключается в выборе случайным образом некоторой позиции j, равномерно распределенной от 1 до N-1, после чего результат формируется в виде

$$\chi' = (\chi_1, \chi_2, \ldots, \chi_j, \tau_{j+1}, \ldots, \tau_N), \quad \tau' = (\tau_1, \tau_2, \ldots, \tau_j, \chi_{j+1}, \ldots, \chi_N).$$

Влияние кроссинговера регулируют с помощью вероятности P_{Cross} срабатывания этого оператора (в противном случае все остается без изменений).

Оператор мутации в каждой позиции аргумента с заданной вероятностью P_{mut} заменяет ее содержимое на случайный элемент двоичного алфавита A, выбранный в соответствии с равномерным распределением (в противном случае все остается без изменений).

Целевая функция исходной задачи, заменяется в ГА на неотрицательную функцию пригодности генотипа $\Phi(\chi)$, где $\chi \in B$.

Процесс работы алгоритма представляет собой последовательную смену поколений, на каждом шаге которой популяция Π_{t+1} наполняется парами потомков от особей популяции Π_t по формуле

$$(\chi_k^{t+1}, \chi_{k+1}^{t+1}) = Mut(Cross(\chi_{Select(\Pi_t)}^t, \chi_{Select(\Pi_t)}^t)),$$

где $(\chi_k^{t+1}, \chi_{k+1}^{t+1})$ - особи с наименьшей пригодностью популяции $П_t$. То есть индивиды извлекаются попарно из $П_t$ и после кроссинговера и мутации помещаются в $П_{t+1}$. Изменение вероятностей мутации и кроссинговера позволяет регулировать работу ГА и настраивать его на конкретные задачи.

3. Сущность генетических алгоритмов для сжатие изображений

Генетические алгоритмы (ГА) являются одним из самостоятельных разделов теории искусственного интеллекта – эволюционных вычислений, которые основаны на математическом моделировании процессов биологической эволюции.

В основе ГА лежат принципы естественной эволюции, сформулированные Дарвином, – естественный отбор, изменчивость, наследственность. С их помощью ГА моделируют генетические процессы, происходящие в биологических сообществах.

Обычно генетические алгоритмы применяются для решения оптимизационных проблем. Предметная область ГА включает в себя проблемы комбинаторики, биоинформатики, теории игр и др. ГА применяются также для обработки и распознавания образов, в частности изображений.

Принципы работы ГА. Сущность ГА состоит в том, что поиск решения проблемы проходит на подмножестве точек пространства поиска. Это достигается тем, что создается множество потенциальных решений, которое формирует популяцию. Популяция совершенствуется с помощью генетических операторов, отвечающих за изменчивость и фитнес-функции, которая моделирует естественный отбор. Наследственность обеспечивается тем, что новые хромосомы формируются из хромосом предыдущего поколения и, соответственно, имеют общие с ними гены. Если ГА реализован корректно, то с каждым новым поколением среднее значение фитнес-

функции популяции и лучшее значение фитнес-функции увеличиваются в сторону глобального оптимума.

Для правильной работы ГА необходимо выбрать способ кодировки данных и фитнес-функцию.

Кодировка данных подразумевает способ представления потенциального решения. Предполагается, что потенциальное решение можно представить в виде параметров (генов), которые можно соединить в простые структуры данных (хромосомы). Традиционно гены кодируются двоичными числами, и хромосомы представляют собой бинарные строки. Кроме того, гены могут быть представлены с помощью алфавита с большей размерностью или числами с плавающей точкой, а хромосомы могут быть представлены, например, как деревья или матрицы.

Фитнес (целевая)-функция служит для оценки пригодности, приспособленности хромосомы. Для каждой проблемы целевая функция подбирается индивидуально. Для конкретного круга задач обычно известно, что должна оценивать фитнес-функция. В частности, при решении проблем оптимизации функций она должна оценивать величину значения самой функции. Однако определение фитнес-функции не так очевидно для проблем связанных, например, с комбинаторной оптимизацией.

Работу простого генетического алгоритма, описанного Голдбергом, можно представить следующим образом:

1. Создается начальная популяция (набор хромосом). Обычно начальная популяция создается случайным образом. Вычисляется фитнес-функция каждой хромосомы популяции и средняя приспособленность популяции. Устанавливается счетчик эпох.

2. Наращивается счетчик эпох. С помощью оператора репродукции формируется промежуточная популяция – популяция родителей с учетом их приспособленности.

3. Формируется следующее поколение. Случайным образом из промежуточной популяции выбирается пара родителей. С заданной вероятностью производится над генотипами выбранных хромосом кроссинговер. Выбирается один из потомков. К нему последовательно применяется оператор инверсии, а затем мутации с заданными вероятностями. Полученный генотип потомка сохраняется в новой популяции.

4. Если в промежуточном поколении еще есть родители, то возврат на пункт 3, иначе – пункт 5

5. Если счетчик поколений достиг заданного значения, то переход к пункту 6, если нет, то переход к пункту 2.

6. Выбор лучших решений. Конец работы.

Основные генетические операторы, которые использует простой ГА, включают в себя репродукцию, кроссинговер, мутацию и инверсию.

Репродукция – процесс формирования промежуточного поколения. В каноническом ГА вероятность хромосомы попасть в промежуточное поколение пропорциональна значению для нее фитнес-функции.

Биологический смысл кроссинговера – передача признаков родителей потомкам. По Холланду, простой оператор кроссинговера выполняется следующим образом. Выбираются две хромосомы (A = a1, a2, a3,…, aL B = a1’, a2’, a3’,…, aL’) , где L – длина хромосомы; выбирается точка кроссинговера (к). Две новые хромосомы формируются из A и B следующим образом: часть хромосомы A до точки кроссинговера соединяется с частью хромосомы B после точки кроссинговера и формирует первую хромосому-потомок, и, аналогично, часть хромосомы B до точки кроссинговера соединяется с частью хромосомы A после точки кроссинговера и формирует вторую хромосому-потомок.

A’ = a1, a2, a3, …, ак, a’к+1, a’ к+2, a’ к+3,…, a’L B’ = a’1, a’2, a’3,…, a’к, ак+1, a к+2, a к+3,…, aL.

Оператор мутации предназначен для того, чтобы поддерживать разнообразие особей в популяции. Он реализуется следующим образом: в каждой строке, которая подвергается мутации, произвольный бит с вероятностью Pm изменяется на противоположный.

При выполнении оператора инверсии хромосома разбивается на две части, которые потом меняются местами.

Вероятность применения операторов мутации и инверсии обычно очень мала (порядка 0,001).

Существует множество вариаций как самих ГА, так и применяемых генетических операторов.

Использование ГА при решении проблем обработки и распознования изображений. В рамках данной проблемы ГА может быть использован как при подготовке изображения к распознаванию, в частности при фильтрации, сегментации, разметке сцены, так и непосредственно для его распознавания.

ГА. Комбинация использования семантических сетей для представления ограничений области и нечеткой логики для достижения соответствия меток этим ограничениям породили новую стратегию вычисления фитнес-функции для работы ГА. Показана возможность применения данного подхода для идентификации снимков облаков на мультиспектральных спутниковых снимках.

Возможность применения адаптивного генетического алгоритма для решения проблемы сегментации цветного изображения, усложненной необходимостью принятия решения об оптимальном количестве сегментов и точного определения текстурных областей исследуется в [5]. Так как во многих случаях при сегментации топологическим областям могут быть поставлены в соответствие области признаков, данную проблему можно сформулировать как оптимизационную и применить ГА для кластеризации небольших районов пространства признаков.

Для квантования изображения применяется комбинированный генетический алгоритм, который объединяет традиционный генетический алгоритм и метод оптимального квантования изображения. Построенный таким образом генетический алгоритм практически нечувствителен к начальным условиям и работает лучше, чем известные ранее алгоритмы.

К. Делибасис и др. описывают текстурный фильтр, работающий с использованием ГА. Фильтр настраивается на определение различных классов текстуры по их корреляции со спектром Фурье, полученным по шаблонам. Целью является проектирование маски фильтра, которая при корреляции со спектром Фурье каждого шаблона инициирует такой отклик, что межклассовые отличия максимизируют, а внутриклассовые – минимизируются. Здесь ГА используется для выбора оптимальной маски из множества возможных. Предложенный подход работает так же хорошо или, лучше, чем традиционные методы, и имеет преимущество в том, что нет необходимости в выборе меры текстуры для специфической структуры изображения. Метод применяется для сегментации магнитно-резонансных изображений мозга.

ГА, который позволяет объединить этап сегментации и этап распознавания изображения предложен в. Поиск ведется на пространстве возможных сегментов изображения, которые сравниваются с шаблонными сегментами. Эксперименты, проводившиеся с изображением, содержащим сложную сцену, показали удовлетворительные результаты.

Распознавание изображений во многом зависит от набора признаков, используемых классификатором.

Существует метод оптимального выделения набора признаков для классификации, который включает ГА и дерево системной индукции (system induction tree) (ДСИ). Этот подход одновременно минимизирует число признаков, использующихся для работы классификатора, и добивается улучшения работы классификатора. ГА используется для поиска в

пространстве возможных подмножеств большего множества признаков-кандидатов. Для полученного подмножества признаков ДСИ создает дерево решений. Значением фитнес-функции для данного подмножества признаков служит качество работы дерева решений на новых данных. Это значение затем используется ГА для получения лучшего множества признаков. Процесс взаимодействия ГА и ДСИ повторяется до тех пор, пока не будет найдено оптимальное множество признаков, с которым бы классификатор работал удовлетворительно. Результаты экспериментов показали, что данный подход осуществим при решении сложных задач, например распознавания черт лица. При этом также наблюдается уменьшение описательной сложности и улучшение качества распознавания по сравнению со стандартными методами выделения признаков для классификатора.

В используется следующий подход выделения набора признаков для классификации. Ведется поиск наименьшего по размерности (или по стоимости) множества признаков, уровень распознавания по которым не ниже заданного.

Проблема поиска оптимального множества признаков определена как проблема оптимизации с ограничениями. В соответствии с этим штрафная функция встраивается в фитнес-функцию. Эксперименты показали, что такой метод работает лучше других известных методов.

Исследования показали, что для бесконечно большого количества точек в N-мерном пространстве поиска при достаточно большом количестве итераций классификатор образов, основанный на ГА, имеет сходство с классификатором Байеса.

4. Модификаций алгоритма фрактального сжатия изображений

Изображения широко используются в различных сферах, как повседневной жизни человека, так и в конкретных областях науки. Очевидно, что качество изображений анализируемых пользователем играет не

последнюю роль. Для хранения изображений с высоким качеством требуются большие объемы памяти, что может создавать неприемлемые ситуации при работе с изображениями. В зависимости от требований, выдвигаемых к изображению, существует возможность, варьируя методы сжатия, получать приемлемые характеристики изображения. Каждый из алгоритмов и их модернизации имеет ряд характеристик, анализируя которые можно варьировать их применение в зависимости от поставленной задачи.

В настоящее время существует достаточное количество направлений улучшения алгоритмов сжатия изображений. В результате выполнения работы ставится задачей получить сравнительный анализ существующих методов модификации фрактальных алгоритмов сжатия. В качестве критерия для сравнения предлагается анализировать основные характеристики результата работы методов и их этапы работы.

В первую очередь укажем основные позиции для базового фрактального алгоритма. Базовый алгоритм предполагает разбиение исходного изображения на доменные и ранговые блоки. После чего для каждого ранга перебирают доменные блоки (для каждого варианта ориентации домен сжимают до размеров рангового блока и определяют оптимальные значения коэффициентов преобразования методом наименьших квадратов). Затем вычисляют нормированное значение параметра L, который характеризует соответствие полученного сжатого доменного блока в его ориентации ранговому блоку. Возможно два режима работы алгоритма (с поиском и без поиска лучшего домена). В режиме с поиском лучшего домена для каждого ранга перебираются все домены, и выбирается с минимальным L. В режиме без поиска наилучшего домена полный перебор доменов останавливают, как только определяется такой i-й домен и его j-я ориентация, что значение его параметра L не превышает заданной допустимой ошибки. Теперь поэтапно рассмотрим существующие методы для модификации базового алгоритма. Наиболее распространенной

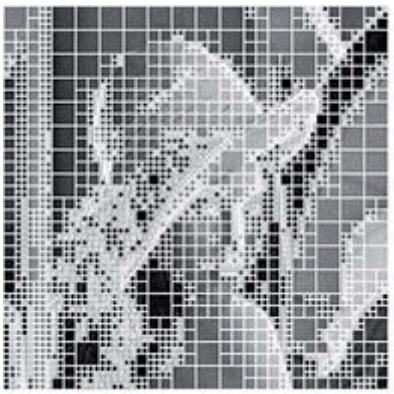
модификацией базового алгоритма является FE-алгоритм. С целью снижения вычислительных затрат в FE-алгоритме выделяют пять характеристик, которые описывают доменные и ранговые блоки. И сначала, проводится именно их сравнение. Это значительно сокращает объем вычислений. Эти характеристики: стандартное отклонение, асимметрия, меж пиксельная контрастность, коэффициент β, которые характеризует отличия значений пикселов от значения центрального пикселя, максимальный градиент – максимум среди горизонтального и вертикального градиентов. При обработке рангового блока вычисляют его вектор характеристик, потом вычисляют расстояния между вектором характеристик данного ранга и вектором характеристик каждого домена. Процедура отбора доменов является своеобразным фильтром, который значительно ограничивает количество доменов, которые перебираются. Для ускорения процесса также является возможным в качестве критерия оптимальности использовать коэффициент корреляции Пирсона: $r^{'} = \rho(R, D)$. Чем лучше реальная зависимость R от D аппроксимируется линейной, тем ближе по модулю к 1 будет их коэффициент корреляции. Использование этого коэффициента позволяет сразу оценить оптимальность текущего домена для данного ранга, без расчета коэффициента преобразования контраста и яркости. Таким образом, эти коэффициенты рассчитываются один раз для каждого ранга. Выполняется преобразование формул, описывающих соотношения для определения коэффициентов яркости и контраста, через коэффициент корреляции. Кроме того, заранее вычисляется среднеквадратичное отклонение яркостей пикселей рангов и доменов. После этого становится возможным рассматривать только те домены, которые удовлетворяют неравенству: $\sigma D > \sigma D$, т.е. контрастность домена должна быть выше контрастности ранга (предлагается подсчитывать среднее значение яркости пикселей для каждого ранга, а не коэффициент яркости для каждого домена, что значительно ускоряет процесс сжатия в среднем в 9,35 раза). Также

существует возможность варьировать и другие характеристики: результат спектрального анализа Фурье, вейвлет анализа, характеристики оттенка или текстуры изображения. Кроме того, алгоритм такого рода эффективно реализован с использованием самообучающихся карт Кохонена.

При анализе литературы по данной тематике было отмечено, что большой интерес для исследователей разных стран составляют алгоритмы с использованием обрамляющих доменов. Перед началом обработки очередного ранга формируются домены, которые обрамляют его. Каждый i-й обрамляющий домен имеет единый геометрический центр со своим рангом и превышает его на величину 2i, где i = 1,2,3,…и т.д. Формирование этих доменов останавливается, как только будет сформирован такой, размер которого в два раза превышает размер ранга. Обрамляющие домены включаются к начально сформированному множеству доменов. При их переборе рассматриваются вначале обрамляющие домены, а потом – другие. Наиболее интересно – высокий процент рангов, для которых выбраны обрамляющие домены, на фоне ухудшения средней пиксельной ошибки.

Широко применяется использование пирамидального метода сравнения. Суть алгоритма следующая. Вместо набора используется непосредственно основной критерий, но применяется он к уменьшенным копиям сравниваемых пар домен-ранг.

Еще одним методом повышения эффективности является алгоритмы, в своей основе имеющие нетривиальные схемы разбиения изображения на блоки. Наиболее популярным является алгоритм квадро-дерева. Он позволяет представить всю схему разбиения в виде древовидной структуры, в которой каждый узел, если он не является последним в ветви, может иметь четыре потомка (рис. а, б).



a. Рисуно к. (а)

b. Рисуно к. (б)

Достоинствами метода является простота реализации и удобство кодирования древовидной структуры, но он считается достаточно жестким и с точки зрения степени сжатия выгодно минимизировать количество блоков в схеме разбиения. Для повышения гибкости предлагается использовать перекрывающиеся ранговые блоки (рис. а, б). Это обеспечивает более рациональное перекрытие изображения блоками по сравнению с алгоритмом квадро-дерева. В результате общее количество блоков уменьшается за счет увеличения количества блоков большего размера. Критерием для выбора размера служит дисперсия яркости пикселей. Такой подход требует больше вычислений, но уменьшение числа рангов позволяет повысить коэффициент сжатия. Относительно новым направлением в модернизации фрактальных методов являются алгоритмы, основанные на минимизации RD-функции Лагранжа. В качестве начальных данных берется некоторый входной набор данных, которому в результате выполнения процедуры сжатия-восстановления ставится в соответствие выходной набор данных той же природы, Y=F(X,u), где u - набор управляющих параметров алгоритма сжатия F. Можно настраивать алгоритм кодирования F на необходимые характеристики. Заслуживающими внимания являются работы описывающие методы анализа эффективности ортогональных преобразований, предназначенных для сжатия коррелированных данных; для сжатия данных

описывается дискретное псевдокосинусное преобразование (ДПКП), новые быстрые алгоритмы вычисления ДПКЛ, на базе которых возможно получать схему компрессии статических изображений (аналогична методу JPEG характеристики). Для обработки неподвижных и динамических изображений предлагаются формализующие процедуры анализа и синтеза схем компрессии цифровых изображений на основе дискретных ортогональных преобразований. Оценки вычислительных затрат показывают, что алгоритм на основе ДПКЛ не уступает JPEG в том числе в части, касающейся вычислительной сложности реализации. В работах иностранных авторов часто упоминается метод для уменьшения времени кодирования за счет сокращения размера бассейна доменов на основе стоимости энтропии каждого доменного блока. Экспериментальные результаты на стандартных изображениях показывают, что предложенный метод дает превосходную производительность по сравнению с обычными алгоритмами фрактального кодирования. Этот метод основан на удалении доменного блока с высокой энтропией, применим в ситуациях, когда требуется очень малое время кодировки и допускается ухудшение качества и является весьма сопоставимым с другими методами ускорения. Таким образом, все бесполезные домены будут удалены из пула, обеспечивая более продуктивный доменный бассейн.

1. Таблица. Краткие замечания и выводы по проведенному анализу

Модификация	Вид изображения	Время кодирования	Средняя пиксельная ошибка	Коэффициент сжатия
Учет частоты использования доменов	Фотореалистичные	Уменьшение на 7-12%	Снижение на 4 %	Практически не изменился
	с резкими переходами	Уменьшение на 33-42%	На 14%	-//-
Обрамляющие домены	Фотореалистичные	Уменьшение на 20-40%	Ухудшение на 28-35%	Увеличивается на 10-28% (наиболее высокий показатель)
	с резкими	Уменьшение	Ухудшение на 30-	-

	переходами	на 9-15%	43%	
Пирамидальный метод сравнения	Фотореалистичные	Повышение в 3-3,5 раз	Улучшение на 43%	Уменьшение на 40-53%
	с резкими переходами	-//-	-//-	-//-
Нелинейное отображение	фотореалистичные	Уменьшение на 13-39%	Ухудшение<=10%	Уменьшается на 4-16%
	с резкими переходами	Уменьшение до 58%	малы	Может увеличиваться не больше чем на 11%
Использование квадро-дерева	требует больше вычислений, но уменьшение числа рангов позволяет повысить коэффициент сжатия.			
Применение ДПКЛ	по всем ключевым характеристикам не уступает JPEG			
Основанные на показателях энтропии	дает превосходную производительность по сравнению с обычными алгоритмами фрактального кодирования			

c. Методы увеличение степени компрессии фрактального сжатие изображение

Основная задача, которую мы решаем — повышение степени сжатия изображений. Когда практически достигнут предел сжатия изображения в целом и различные методы дают очень небольшой выигрыш, мы можем существенно (до 2 раз) увеличить степень сжатия за счет изменения качества разных участков изображения.

d. Рисунок. Локальное улучшение качества областей изображения

Проблемой этого подхода является то, что необходимо каким-то образом получать расположение наиболее важных для человека участков

изображения. Например, таким участком на фотографии человека является лицо, а на лице

Если при сжатии портрета с потерями с большим коэффициентом будут размыты предметы, находящиеся на заднем плане — это будет не существенно. Однако, если будет размыто лицо или глаза — экспертная оценка степени потерь будет низкой.

Работы по автоматическому выделению таких областей сейчас ведутся. В частности, созданы алгоритмы автоматического выделения лиц на изображениях. Идут попытки выделения наиболее значимых контуров и т.д. Однако очевидно, что универсальный алгоритм в ближайшее время создан не будет, поскольку для этого требуется построить схему восприятия изображения мозгом человека.

На сегодня вполне реально применение полуавтоматических систем, в которых качество областей изображения будет задаваться интерактивно. Данный подход уменьшает количество возможных областей применения модифицированного алгоритма, но позволяет достичь большей степени сжатия.

Рассмотрим ограничения на требования приложений к алгоритму:

1) Для приложения должна быть критична (максимальна) степень компрессии, причем настолько, что возможен индивидуальный подход к каждому изображению.
2) Изображение должно сжиматься один раз, а распаковываться множество раз.
3) Изображение должно распаковываться быстро.
4) Скорость сжатия не должна играть роли.

В качестве примеров приложений, удовлетворяющим этим ограничениям, можно привести практически все мультимедийные продукты на CD-ROM. И для CD-ROM энциклопедий, и для игр важно записать на диск как можно больше информации, а графика, как правило, занимает до

70% всего объема диска. При этом технология производства дисков позволяет сжимать каждое изображение индивидуально, максимально повышая степень сжатия.

Интересным примером являются WWW сервера. Для них тоже, как правило, выполняются изложенные выше ограничения. Технологии plug-ins и использование Java позволяют хранить изображения на сервере в любом формате. При этом совершенно не обязательно индивидуально подходить к каждому изображению, поскольку по статистике 10% изображений будут запрашиваться 90% раз. Т.е. для крупных справочных или игровых серверов появляется уменьшать время загрузки изображений и степень загруженности каналов связи адаптивно.

Введение во фрактальную компрессию. Фрактальный алгоритм, впервые описанный в, теперь достаточно хорошо известен. Основным недостатком алгоритма является то, что при осуществлении сжатия изображений требуются значительные вычислительные ресурсы, в то время как разархивация происходит очень быстро.

Как известно, фрактальный алгоритм позволяет достичь самых высоких степеней сжатия (до 1000 раз) для сложных по своей структуре изображений. Применение фрактального сжатия для небольших изображений и небольших степеней сжатия не дает существенного выигрыша, поскольку мы получаем аналогичные результаты для JPEG и Wavelet алгоритмов. Однако при больших степенях сжатия применение фрактального алгоритма позволяет получить заметный выигрыш. Последнее положение и определяет актуальность излагаемого подхода для фрактального алгоритма сжатия изображений.

Основная сложность в применении нашего подхода к фрактальному алгоритму заключается в том, что он использует в качестве «словаря» блоков само изображение при восстановлении. Т.е. ухудшение качества в какой-то области влечет за собой ухудшение качества всего изображения. Для того,

чтобы избавиться от этого эффекта, требуется существенно изменить схему алгоритма компрессии. Подчеркнем, что алгоритм декомпрессии при применении данного подхода не изменяется совершенно.

Кроме того, при использовании данного подхода полностью пропадает возможность использования любых мер определения степени потерь. Только экспертная оценка дает адекватный результат.

Излагаемый подход основан на изменении разбиения изображения при сжатии. К настоящему моменту исследовано довольно много различных схем разбиения. Их можно классифицировать как:

• Схемы иерархического разбиения (квадродере-вом, полигональное и HV разбиение) при которых происходит только дробление элементов в процессе итераций;

• Схемы итеративного разбиения (триангуляция Делоне, прямоугольниками, эвристический поиск, эволюционное и адаптивное разбиения) при которых происходит дробление и объединение элементов в процессе итераций.

Ближе всего к теме данной статьи вопросы построения оптимального иерархического разбиения, которые также можно использовать для квадродеревьев, рассматриваемые с.

Описание алгоритма основная идея. Основная идея реализации предложенной идеи достаточно проста: изменяется критерий выбора блока. Нам необходимо брать блоки для приближения заданного участка не из участка, наилучшим образом приближающего данный, а из участка, который достаточно хорошо приближает данный и не сильно страдает при компрессии. Это позволяет повышать качество выбранного участка не только за счет уменьшения дробления разбиения квадродерева и минимизирует потери от неравномерного приближения разных участков изображения. Поскольку при прочих равных для приближения «качественного» участка используются блоки из других «качественно» приближенных участков. В

качестве критерия оптимального блока берется то же среднеквадратичное расстояние с учетом поправки. Если в качестве исходного расстояния между блоками берется:

$$[\min]R = \sum_{i=1,\,j=1}^{n,n} (s \cdot a_{i,j} + o - b_{i,j})^2$$

где $S \in [0,1]$ и o могут быть явно получены по методу наименьших квадратов, то новый критерий выглядит так: $R' = R + s \sum_{i=1,\,j=1}^{n,n} r_{i,j}$, где $r_{i,j} = \frac{R_{i,j}}{n^2}$ — удельное приближение точки блока A, который переводится в блок B в данном преобразовании.

Сложность этого подхода заключается в том, что критерий оптимальности (маску качества $r_{i,j}$ приближения изображения) мы получаем только после сжатия. Т.е. при ускорении алгоритма и отказе от полного перебора, мы не только не достигаем оптимума, но и вынуждены пользоваться смещенным критерием.

Построение алгоритма. Введем понятия доменных и ранговых областей. Области, D_i из которых производится выбор подобных областей называются ранговыми (A); области R_i, для которых подбираются подобные области, называются доменными (B).

Если исходный алгоритм кратко записать как:

Шаг 1. Подготовка всех структур данных.

Шаг 2. Поиск наилучшего блока:

for (по всем подготовленным ранговым блокам) {

for (по всем доменным блокам) {

current= сохраним текущие координаты;

D=image->скопируем из изображения блок с текущими коорд.;

current_dist=Среднеквадр._расст.(D Rij);

if(current_dist < dist) {

ij distij = запомним расстояние;

bstij = запомним лучшие координаты;

} } }

```
Сохраним_Лучшие_Коэффициенты(все best);
```

В приведенном коде image — сжимаемое изображение, distij — массив в котором хранятся наименьшие расстояния, а bstij — массив с коэффициентами преобразований. Данная схема упрощена, поскольку на практике, при построении разбиения в виде квадродерева, используются более сложные структуры.

Модифицированный алгоритм отличается тем, что мы реально запоминаем не одно преобразование для каждого приближаемого блока, а создаем сортированный список bestlij таких преобразований (т.е. используется уже не массив структур коэффициентов, а массив сортированных списков этих структур), упорядоченных по среднеквадратичному расстоянию current_dist. На псевдокоде изменения будут выглядеть так:

```
for (по всем подготовленным ранговым блокам) {
for (по всем доменным блокам) {
current= сохраним текущие координаты;
D=image->скопируем из изображения блок с текущими коорд.;
current_dist=Среднеквадр._расст.(D R);
ij Вставить_в(bestij, координаты, current_dist);
}
}
//Подготовим оценочное изображение.
for (по всем best) {
ij Закрасить в оц. изобр. квадрат с
яркостью, соответствующей точности
приближения лучшего преобразования
}
for (по всем best) {
ij for (по всем преобразованиям в best)
ij Выбрать в списке лучшее преобразование
}
Сохраним_Лучшие_Коэффициенты(все best);
```

е. Рисунок. Тестовое изображение Barbara 512x512 Grayscale, сжатое только с учетом минимизации среднеквадратичного расстояния изображения

На рисунке 14 представлено тестовое изображение Barbara, сжатое с использованием исходного алгоритма. Данная фотография является стандартной при сравнении алгоритмов сжатия, поскольку содержит много неприятных «полосатых» текстур (скатерть, кресло, одежда), на которых многие алгоритмы сжатия с потерями либо создают муаровые узоры, либо существенно теряют качество. Использованный фрактальный алгоритм на этом изображении, достаточно хорошо воспроизводит эти области за счет использования гибридной схемы с алгоритмом квантования векторов [12]. Однако существенная для восприятия часть изображения — лицо достаточно существенно потеряло в качестве.

f. Рисунок. Откорректированное тестовое изображение Barbara

На откорректированном изображении (рис. 15) видно, как возросло качество лица и правой руки. В увеличенном виде изменения на лице хорошо видны на рисунках 2-3.

g. Рисунок Участок локального улучшения качества в увеличенном виде.

Как уже было сказано выше, реально при построении системы итерируемых функций, приближающих данное изображение, используется разбиение в виде квадродерева. В нашем алгоритме критерием дальнейшего разбиения узла квадродерева служит пороговое значение среднеквадратичного расстояния приближения соответствующего узла. Не останавливаясь в данной статье на подробностях реализации взвешенного

вычисления пороговых величин опишем, как они вычисляются в модифицированном алгоритме.

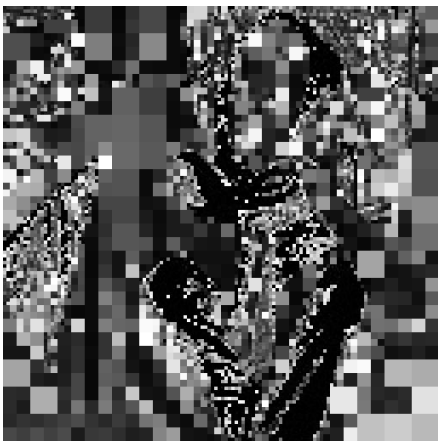
Обозначим как B старое взвешенное предельное значение среднеквадратичного расстояния. Тогда новое вычисляется как:

$$B' = B \cdot \left(1 - \frac{\sum_{i=x}^{x+size} \sum_{j=x}^{y+size} q_{i,j}}{255 \cdot size^2} \right)$$

Где q это яркость точек в изображении, используемом как маска (рис. 3). Фактически по этому изображению строятся весовые коэффициенты, влияющие на разбиение квадродерева. Как $size$ обозначен размер соответствующего блока в квадродереве.

h. Рисунок. Вид изображения-маски

Оценочные изображения, получаемые на последнем шаге алгоритма, задают, насколько точно приближен соответствующий блок квадродерева.

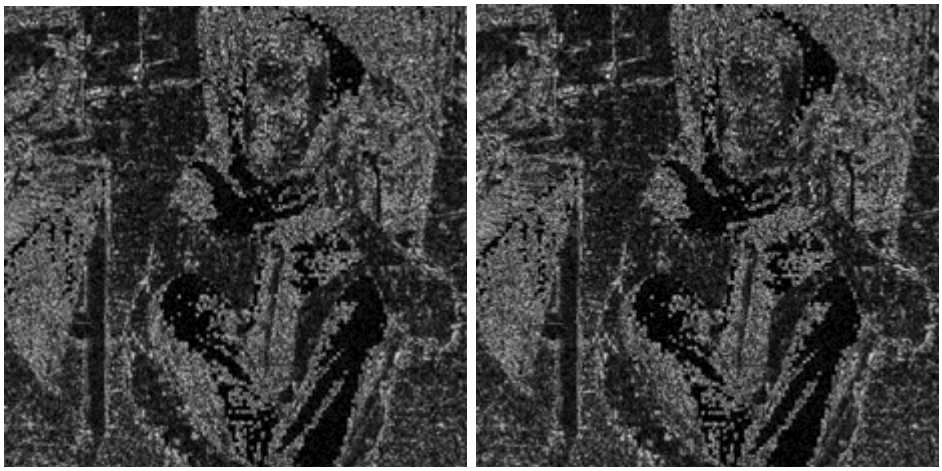


i. Рисунок. Степень приближения блоков квадродерева для оригинального алгоритма.

На рисунках 4-1 и 4-2 более яркие квадраты изображения соответствуют худшему приближению, а более светлые — лучшему. Хорошо видно, как более сложные участки изображения представлены более мелкой сеткой. Использование маски дает заметное на глаз снижение яркости на соответствующих участках изображения.

j. Рисунок. Степень приближения блоков квадродерева для модифицированного алгоритма — маска, использованная для отбора блоков.

Если рассмотреть модуль разности между элементами изображения (рис. 19, контрастность увеличена в 32 раза), то видно, насколько данный алгоритм улучшает приближение выбранных областей изображения. При этом качество остального изображения и степень компрессии практически не изменяются.

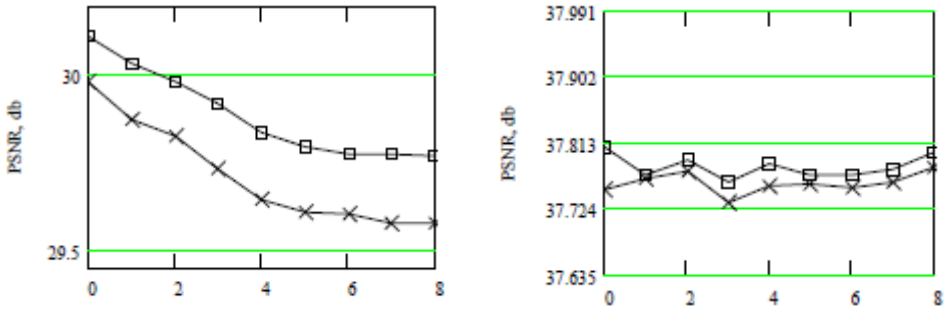


k. Рисунок. Разница между исходным изображением и изображениями, полученными после декомпрессии, для оригинального и модифицированного алгоритмов.

Построенная модификация фрактального алгоритма практически не изменяет время компрессии и совершенно не влияет на процедуру декомпрессии. Последнее весьма существенно для реальных приложений.

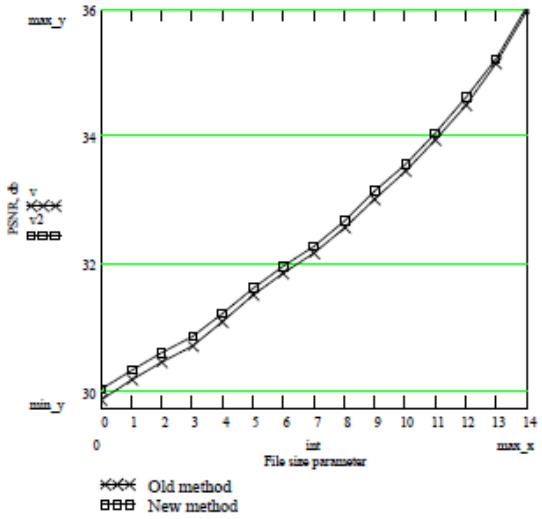
Время и степень сжатия. Основным результатом работы является то, что построен реально функционирующий алгоритм способный адаптивно изменять качество областей изображения. Степень сжатия уменьшается пропорционально размеру светлых областей в изображении-маске. Время компрессии увеличивается крайне незначительно. Даже на Pen-tium 200 это меньше секунды на изображение 512x512. Т.е. даже при максимальной скорости компрессии замедление не превышает 2%.

Повышение качества изображения. Модификация алгоритма такова, что даже если не производится локальное улучшение качества изображения, общее качество изображений увеличивается. На рис. 6 приведено изменение PSNR тестовых изображений для разных скоростей работы алгоритма (используется разное количество классов разбиения множества блоков). Улучшение качества незначительно, но стабильно.



l. Рисунок. Улучшение качества изображения для изображений Barbara и Dmc

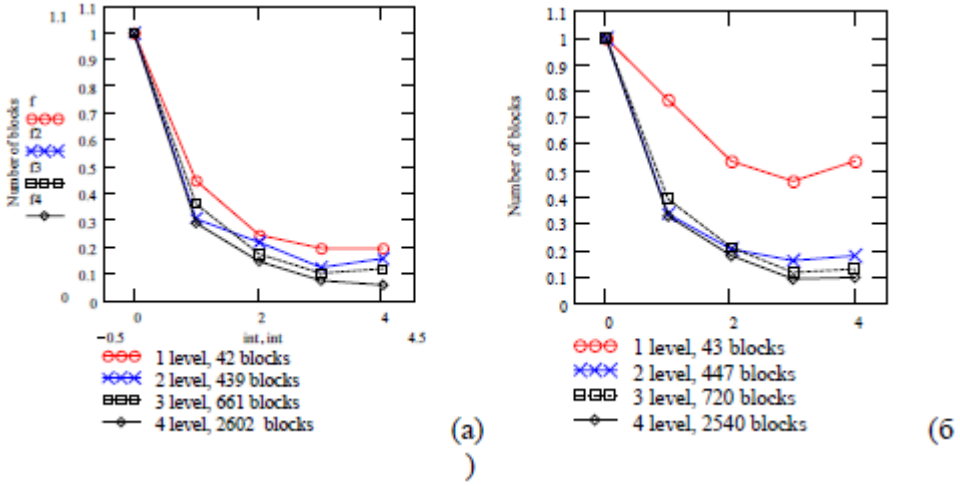
При большей степени компрессии, как и ожидалось, наблюдаемый выигрыш больше, чем для невысоких степеней. Это повышает актуальность фрактального алгоритма, поскольку при невысоких степенях компрессии выгоднее применять JPEG.



m. Рисунок. Улучшение качества изображения для изображений Barbara для разного параметра размера файла

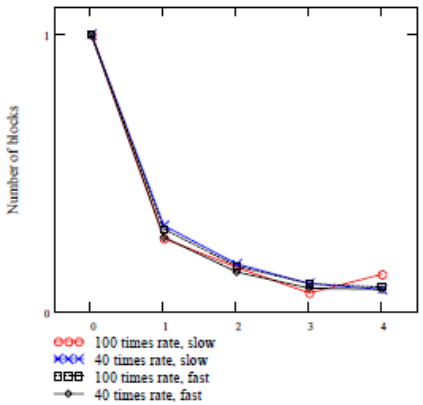
Статистика изменения преобразований. Как уже говорилось, блоки в новом алгоритме помещаются в сортированный список, из которого потом производится выбор оптимального. Выбор нулевого элемента списка — выбор элемента с наименьшим найденным среднеквадратичным расстоянием — соответствует оригинальному алгоритму. Посмотрим, как распределяется

выбор первых пяти элементов списка на первых четырех уровнях квадродерева (рис. 8).

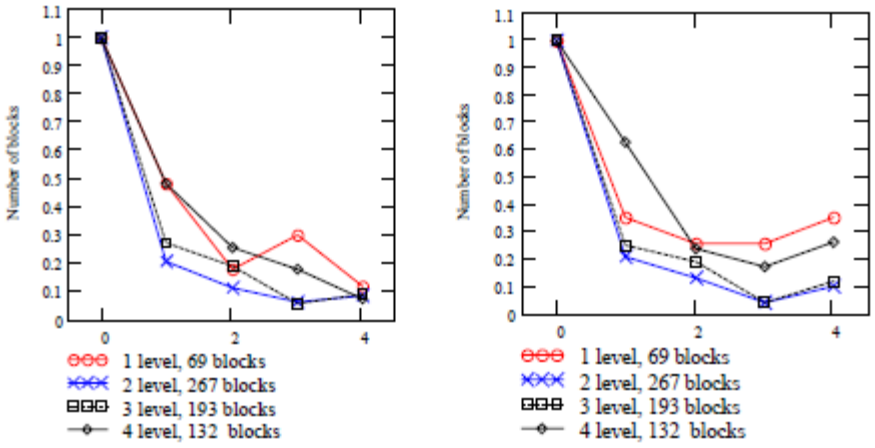


n. Рисунок. Изображение Barbara распределение выбора координат преобразований для малого (а) и большого (б) времени компрессии для сжатия в 7 раз

Для удобства сравнения количество преобразований нормировано относительно преобразований, оставленных без изменений. Хорошо видно, что при увеличении времени компрессии до полного перебора (б) увеличивает количество выбранных ненулевых преобразований. Однако на рис. 9-1 хорошо видно, что в массе своей увеличение времени компрессии не изменяет распределения номеров выбранных преобразований.



o. Рисунок. Распределение выбора координат всех блоков для изображения Dmc малого (slow) и большого (fast) времени компрессии



p. Рисунок. Изображение Dmc распределение для малого (а) и большого (б) времени компрессии для сжатия в 100 раз

Ломаный характер кривых объясняется малым количеством блоков на этих уровнях квадродерева. На рисунке 10 приведено качество изображения для сжатия в 100 раз. При сжатии в 40 раз, параметры которого иллюстрируются на рис. 9-1, качество изображения уже не позволяет сказать, что изображение подвергалось сжатию.

q. Рисунок. Изображение 330x320 TrueColor Dmc сжатое в 100 раз

Выводы по второй главе.

Вторая глава данной магистерской работы посвящена исследованию методов фрактального сжатия данных, а так же изображений. Обоснованы методы фрактального сжатия. Изучена работа алгоритма Фишера. Изложена информация об алгоритмах поиска фрактальной модели изображения. А также в данной главе была приведена сущность генетических алгоритмов для сжатия изображений. Анализированы методы модификации фрактальных изображений.

III. РАЗРАБОТКА ПО ДЛЯ ФРАКТАЛЬНОГО СЖАТИЯ ИЗОБРАЖЕНИЯ

1. Сравнительный анализ технологии для разработки ПО решающей задачи сжатие изображение фрактальным алгоритмом

Пожалуй, наиболее важной вехой в истории программирования, сравнимой по значимости разве что с изобретением письменности, можно считать переход от машинных кодов (тарабарщины типа 0110110101111...) к понятным простому смертному языкам программирования (типа ALGOL, FORTRAN, PL/1, Pascal), а также к широкому использованию методов структурного программирования. Программы стали модульными, состоящими из подпрограмм. Появились библиотеки готовых подпрограмм, облегчающие многие задачи, но все равно программистам хватало трудностей, особенно при разработке пользовательского интерфейса.

Объектно-ориентированное программирование. Качественным шагом в развитии методов структурного программирования стало изобретение объектно-ориентированного программирования (языков SmallTalk, C++, Turbo Pascal и др.). Программы стали строиться не из чудовищных по размеру процедур и функций, перерабатывающих громоздкие структуры данных, а из сравнительно простых кирпичиков-объектов, в которых были упрятаны данные и подпрограммы их обработки. Гибкость объектов позволила очень просто приспосабливать их для собственных целей, прилагая для этого минимум усилий. Программисты обзавелись готовыми библиотеками объектов, но, как и раньше, создание пользовательского интерфейса требовало уйму времени и сил, особенно когда программа должна была работать под управлением популярной операционной системы Windows и иметь графический пользовательский интерфейс.

Визуальное программирование. С изобретением визуального программирования, первой ласточкой которого была среда разработки Visual Basic, создание графического пользовательского интерфейса стало под силу даже новичку. В среде Visual Basic можно было быстро создать приложение для операционной системы Windows, в котором были все присущие графическому пользовательскому интерфейсу элементы: окна, меню, кнопки, поля ввода и т.д. Все эти элементы превратились в строительные блоки программы — компоненты — объекты, имеющие визуальное представление на стадии проектирования и во время работы.

Среда программирования Delphi. Мечта программистов о среде программирования, в которой бы простота и удобство сочетались с мощью и гибкостью, стала реальностью с появлением среды Delphi. Она обеспечивала визуальное проектирование пользовательского интерфейса, имела развитый объектно-ориентированный язык Object Pascal (позже переименованный в Delphi) и уникальные по своей простоте и мощи средства доступа к базам данных. Язык Delphi по возможностям значительно превзошел язык Basic и даже в чем-то язык C++, но при этом он оказался весьма надежным и легким в изучении (особенно в сравнении с языком C++). В результате, среда Delphi позволила программистам легко создавать собственные компоненты и строить из них профессиональные программы. Среда оказалась настолько удачной, что по запросам любителей C++ была позже создана среда C++Builder — клон среды Delphi на основе языка C++ (с расширенным синтаксисом).

Среда Delphi стала, по сути, лучшим средством программирования для операционной системы Windows, но программистов ждало разочарование, если возникало желание перенести программу в другую операционную систему, в частности, в операционную систему Unix.

Технология Java. Практически одновременно со средой программирования Delphi на свет появилась технология Java, включавшая

три составляющих: одноименный язык программирования, очень похожий на язык C++, но более простой и безопасный; универсальный байт-код, в который компилировались программы на языке Java; интерпретатор (виртуальную машину) для выполнения байт-кода в любой операционной системе. Благодаря автоматическому управлению памятью — так называемой «сборке мусора» — резко повысилась надежность программ и скорость их разработки.

Поначалу на технологию Java возлагались большие надежды. Программные библиотеки для языка Java стали единым стандартом, поэтому написанные на нем программы оказались по-настоящему переносимыми. Однажды написанная и компилированная в байт-код программа могла работать на любой платформе без ограничений (единственное требование — наличие на этой платформе виртуальной машины Java).

Среда программирования Kylix. В связи со стремительным распространением операционной системы Linux возникла необходимость в эффективных средствах создания для нее программ. Таким средством стала среда Kylix (произносится «киликс») — первая среда визуального программирования для операционной системы Linux. Среда Kylix явилась полным аналогом среды Delphi и была совместима с ней по языку программирования и библиотекам компонентов. Программу, созданную в среде Delphi, можно было без изменений компилировать в среде Kylix, и наоборот. Эта возможность достигалась за счет новой библиотеки компонентов, которая взаимодействовала с операционной системой не напрямую, а через промежуточный программный слой, скрывающий разницу в работе компонентов в той или иной операционной системе. Программисты получили возможность создавать программы сразу для двух самых популярных операционных систем: Windows и Linux. Фактически вместо принципа абсолютной переносимости программ была предложена идея разумной переносимости.

Технология .NET. Несмотря на трудности и уроки Java-технологии, программисты не желали отказываться от идеи создания полностью переносимых программ. Вместе с тем их совершенно не устраивала необходимость платить производительностью и удобством программ за переносимость. Работы по разрешению этого противоречия привели к появлению на свет технологии под названием .NET (произносится «дот-нет»).

Поначалу технология .NET была доступна только для семейства операционных систем Windows, но со временем этот недостаток был устранен, и на свет появилась платформа Mono — клон технологии .NET для операционных систем Linux и Unix.

2. Назначение среды программирование Delphi 2007 в разработки программной среды

Внешний вид среды программирования Delphi отличается от многих других из тех, что можно увидеть в Windows. К примеру, Borland Pascal for Windows 7.0, Borland C++ 4.0, Word for Windows, Program Manager - это все MDI приложения и выглядят по-другому, чем Delphi. MDI (Multiple Document Interface) - определяет особый способ управления нескольких дочерних окон внутри одного большого окна.

Среда Delphi же следует другой спецификации, называемой Single Document Interface (SDI), и состоит из нескольких отдельно расположенных окон. Это было сделано из-за того, что SDI близок к той модели приложений, что используется в Windows 95.

Если Вы используете SDI приложение типа Delphi, то уже знаете, что перед началом работы лучше минимизировать другие приложения, чтобы их окна не загромождали рабочее пространство. Если нужно переключиться на другое приложение, то просто щелкните мышкой на системную кнопку

минимизации Delphi. Вместе с главным окном свернутся все остальные окна среды программирования, освободив место для работы других программ.

Главные составные части среды программирования

Ниже перечислены основные составные части Delphi:

1. Дизайнер Форм (Form Designer)
2. Окно Редактора Исходного Текста (Editor Window)
3. Палитра Компонент (Component Palette)
4. Инспектор Объектов (Object Inspector)
5. Справочник (On-line help)

Есть, конечно, и другие важные составляющие Delphi, вроде линейки инструментов, системного меню и многие другие, нужные Вам для точной настройки программы и среды программирования.

Программисты на Delphi проводят большинство времени переключаясь между Дизайнером Форм и Окном Редактора Исходного Текста (которое для краткости называют Редактор).

Дизайнер Форм в Delphi столь интуитивно понятен и прост в использовании, что создание визуального интерфейса превращается в детскую игру. Дизайнер Форм первоначально состоит из одного пустого окна, которое Вы заполняете всевозможными объектами, выбранными на Палитре Компонент.

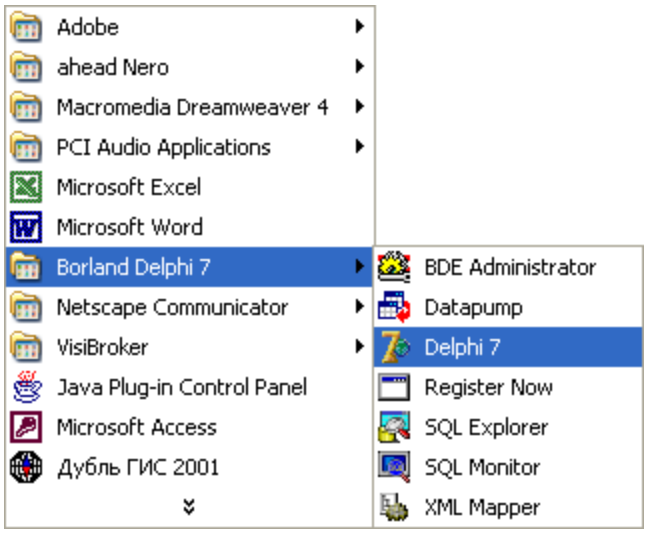
Стандартные компоненты. Для дальнейшего знакомства со средой программирования Delphi потребуется рассказать о составе первой страницы Палитры Компонент.

На первой странице Палитры Компонент размещены 14 объектов определенно важных для использования. Мало кто обойдется длительное время без кнопок, списков, окон ввода и т.д. Все эти объекты такая же часть Windows, как мышь или окно.

Набор и порядок компонент на каждой странице являются конфигурируемыми.

Delphi - это среда быстрой разработки, в которой в качестве языка программирования используется язык Delphi. Язык Delphi - строго типизированный объектно-ориентированный язык, в основе которого лежит хорошо знакомый программистам Object Pascal.

Работа в Delphi. Запускается Delphi обычным образом, т.е. выбором из меню Borland Delphi 7 команды Delphi 7 (рис.1).

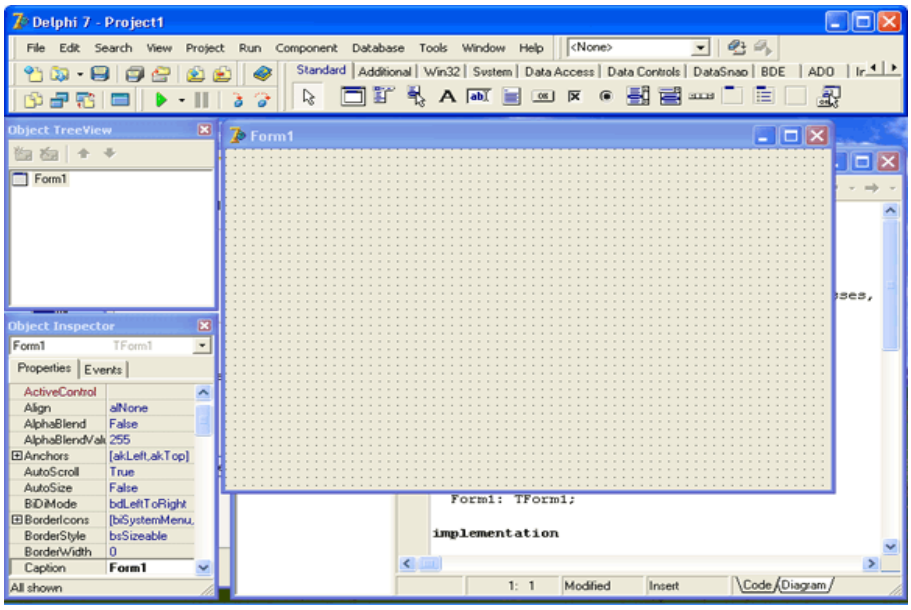


5. Рисунок. Запуск Delphi

Вид экрана после запуска Delphi несколько необычен (рис.2). Вместо одного окна на экране появляются пять:

- главное окно - Delphi 7;
- окно стартовой формы - Form 1;
- окно редактора свойств объектов - Object Inspector;
- окно просмотра списка объектов - Object TreeView;
- окно редактора кода - Unitl. pas.

Окно редактора кода почти полностью закрыто окном стартовой формы.

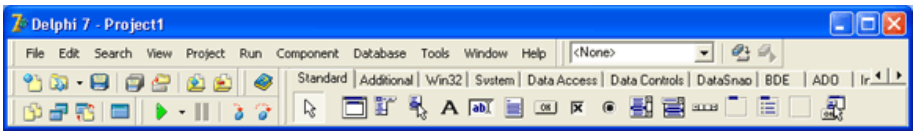


6. Рисунок. Вид экрана после запуска Delphi

В главном окне (рис.28) находится меню команд, панели инструментов и палитра компонентов.

Окно стартовой формы (Forml) представляет собой заготовку главного окна разрабатываемого приложения.

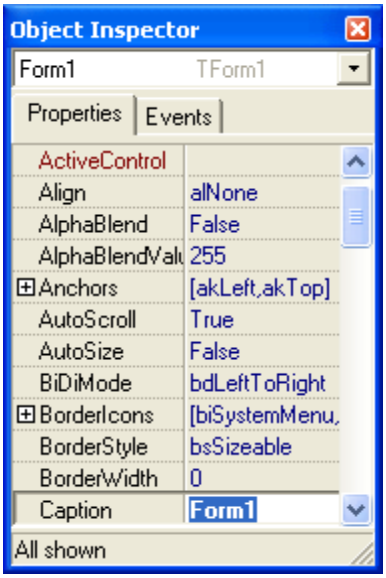
Программное обеспечение принято делить на системное и прикладное. Системное программное обеспечение - это все то, что составляет операционную систему. Остальные программы принято считать прикладными. Для краткости прикладные программы называют приложениями.



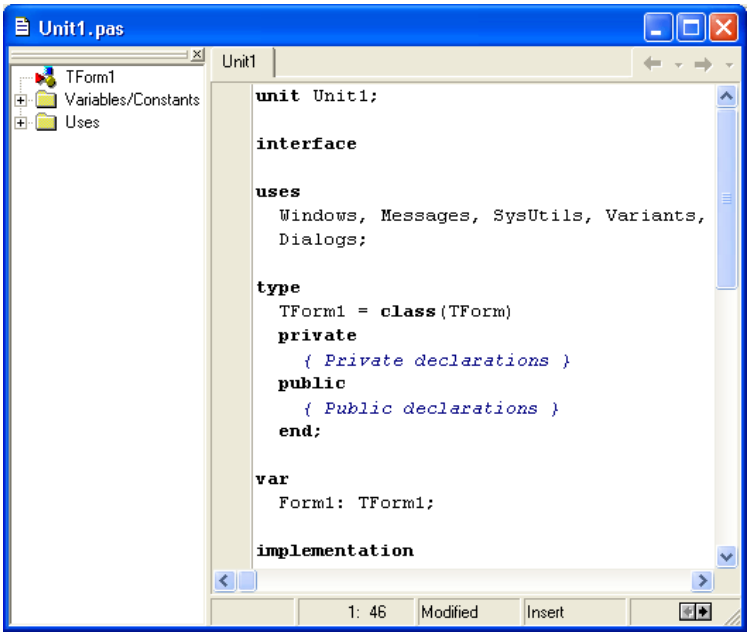
7. Рисунок. Главное окно

Окно Object Inspector (рис.29) - окно редактора свойств объектов предназначено для редактирования значений свойств объектов. В терминологии визуального проектирования объекты - это диалоговые окна и элементы управления (поля ввода и вывода, командные кнопки, переключатели и др.). Свойства объекта - это характеристики, определяющие

вид, положение и поведение объекта. Например, свойства width и Height задают размер (ширину и высоту) формы, свойства top и Left - положение формы на экране, свойство caption - текст заголовка.



8. Рисунок. На вкладке Properties перечислены свойства объекта и указаны их значения



9. Рисунок. Окно редактора кода

В окне редактора кода (рис.31), которое можно увидеть, отодвинув в сторону окно формы, следует набирать текст программы. В начале работы

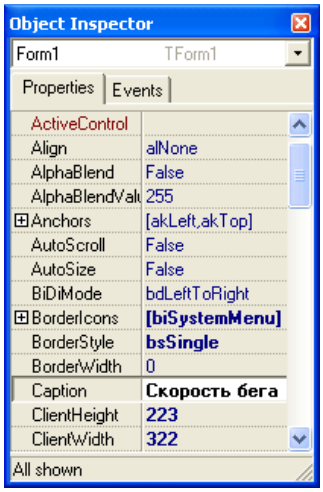
над новым проектом это окно редактора кода содержит сформированный Delphi шаблон программы.

Работа над новым проектом, так в Delphi называется разрабатываемое приложение, начинается с создания стартовой формы. Так на этапе разработки программы называют диалоговые окна.

Стартовая форма создается путем изменения значений свойств формы Form1 и добавления к форме необходимых компонентов (полей ввода и вывода текста, командных кнопок).

Аналогичным образом можно установить значения свойств Height и width, которые определяют высоту и ширину формы. Размер формы и ее положение на экране, а также размер других элементов управления и их положение на поверхности формы задают в пикселах, т.е. точках экрана. Свойствам Height и width надо присвоить значения 250 и 330 соответственно.

Форма - это обычное окно. Поэтому его размер можно изменить точно так же, как размер любого другого окна, т.е. захватом и перемещением (с помощью мыши) границы. По окончании перемещения границ автоматически изменятся значения свойств Height и width. Они будут соответствовать установленному размеру формы.

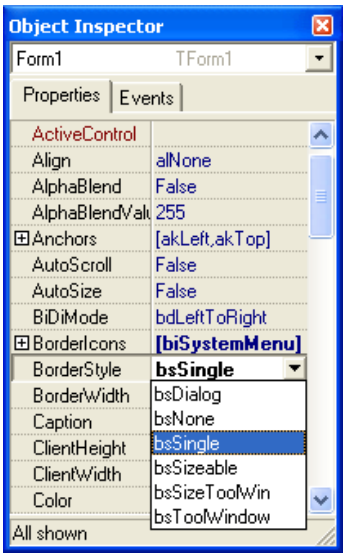


32. Рисунок. Установка значения свойства путем ввода значения

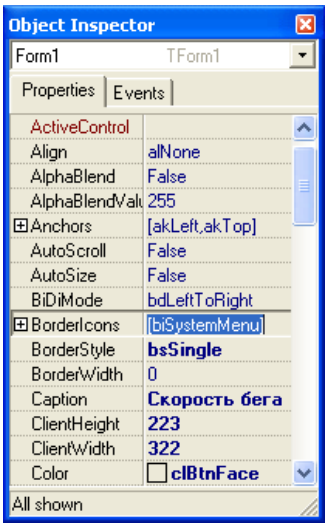
Положение диалогового окна на экране после запуска программы соответствует положению формы во время ее разработки, которое определяется значением свойств top (отступ от верхней границы экрана) и Left (отступ от левой границы экрана). Значения этих свойств также можно задать путем перемещения окна формы при помощи мыши.

При выборе некоторых свойств, например, Borderstyle, справа от текущего значения свойства появляется значок раскрывающегося списка. Очевидно, что значение таких свойств можно задать путем выбора из списка (рис.32).

Некоторые свойства являются сложными, т.е. их значение определяется совокупностью значений других (уточняющих) свойств. Перед именами сложных свойств стоит значок "+", при щелчке на котором раскрывается список уточняющих свойств (рис.4). Например, свойство BorderIcons определяет, какие кнопки управления окном будут доступны во время работы программы. Так, если свойству biMaximize присвоить значение False, то во время работы программы кнопки Развернуть в заголовке окна не будет.



33. Рисунок. Установка значения свойства путем выбора из списка



34. Рисунок. Раскрытый список вложенных свойств сложного свойства BorderIcons

Рядом со значениями некоторых свойств отображается командная кнопка с тремя точками. Это значит, что для задания значения свойства можно воспользоваться дополнительным диалоговым окном. Например, значение сложного свойства Font можно задать путем непосредственного ввода значений уточняющих свойств, а можно воспользоваться стандартным диалоговым окном выбора шрифта.

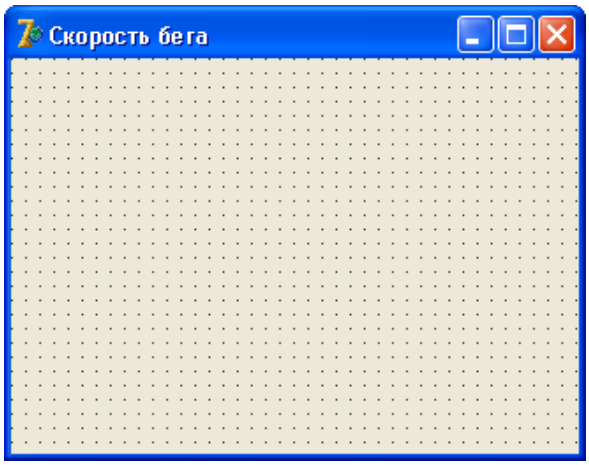
В таблице 3 перечислены свойства формы разрабатываемой программы, которые следует изменить. Остальные свойства оставлены без изменения и в таблице не приведены.

IV. Таблица. Значения свойств стартовой формы

Свойство	Значение
Caption	Скорость бега
Height	250
Width	330
BorderStyle	bsSingle
BorderIcons. biMinimize	False
BorderIcons. biMaximize	False

Font. Size	10

В приведенной таблице в именах некоторых свойств есть точка. Это значит, что надо задать значение уточняющего свойства.



35. Рисунок. Так выглядит форма после установки значений свойств

События

Событие	Происходит
OnClick	При щелчке кнопкой мыши
OnDblClick	При двойном щелчке кнопкой мыши
OnMouseDown	При нажатии кнопки мыши
OnMouseUp	При отпускании кнопки мыши
OnMouseMove	При перемещении мыши
OnKeyPress	При нажатии клавиши клавиатуры
OnKeyDown	При нажатии клавиши клавиатуры. События OnKeyDown и OnKeyPress - это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие OnKeyUp)
OnKeyUp	При отпускании нажатой клавиши клавиатуры
OnCreate	При создании объекта (формы, элемента управления). Процедура обработки этого события обычно используется для инициализации переменных, выполнения подготовительных действий
OnPaint	При появлении окна на экране в начале работы программы, после появления части окна, которая, например, была закрыта другим окном, и в других случаях
OnEnter	При получении элементом управления фокуса

3. Описание и назначение предлагаемой ПО

Для запуска программы от пользователя требуется пройти по следующей ссылке: **Fractal_Compression\FractComp.exe.** Стоит отметить, что разработанное программное обеспечение для решения задачи сжатия данных, а так же изображений методом фрактального сжатия состоит из нескольких окон пользователя. Ниже, приводятся все указания по эксплуатации, разработанные программным обеспечением в ходе выполнение магистерской работы. После запуска программы, у монитора пользователя появляется следующее окно программы

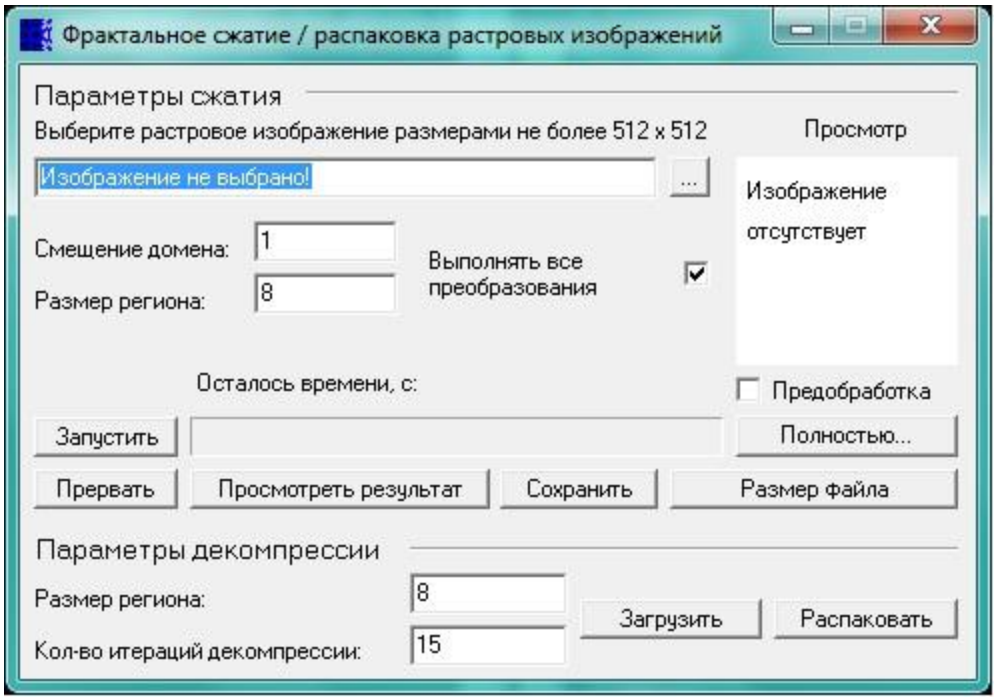


Рисунок 3.1. Главное окно программы

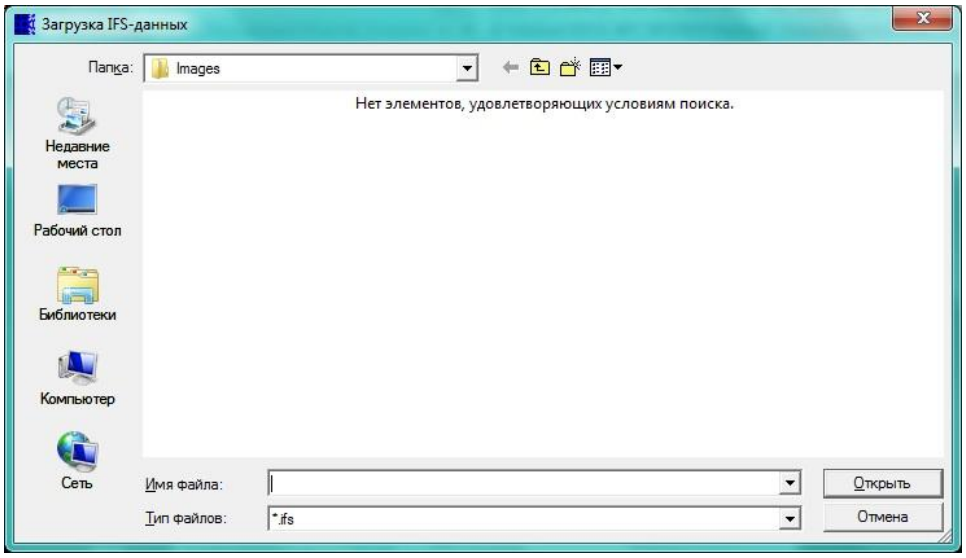


Рисунок 3.2. Загрузка изображения в программу

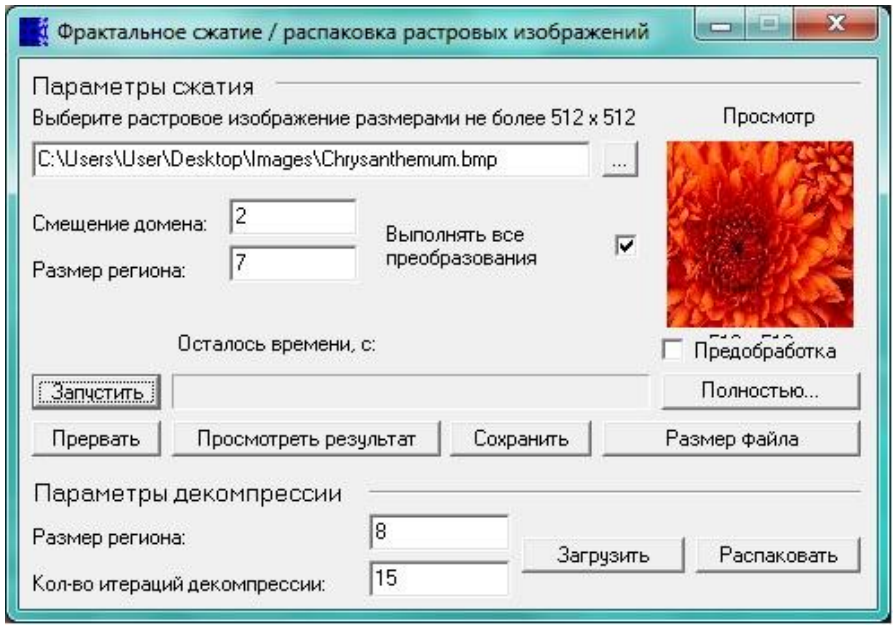


Рисунок 3.3. Настройка параметров сжатия изображения

С помощью кнопки загрузки изображения можно загрузить изображение всех форматов. После настройки параметров сжатия изображений нажимается кнопка «Запуск»

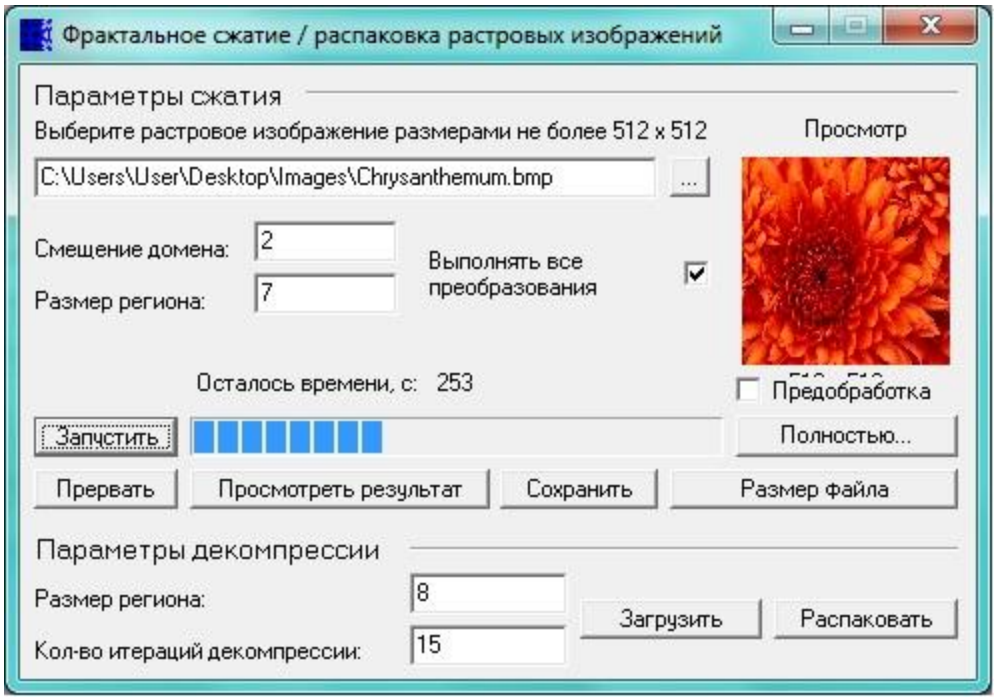


Рисунок 3.4. Процесс сжатия изображения

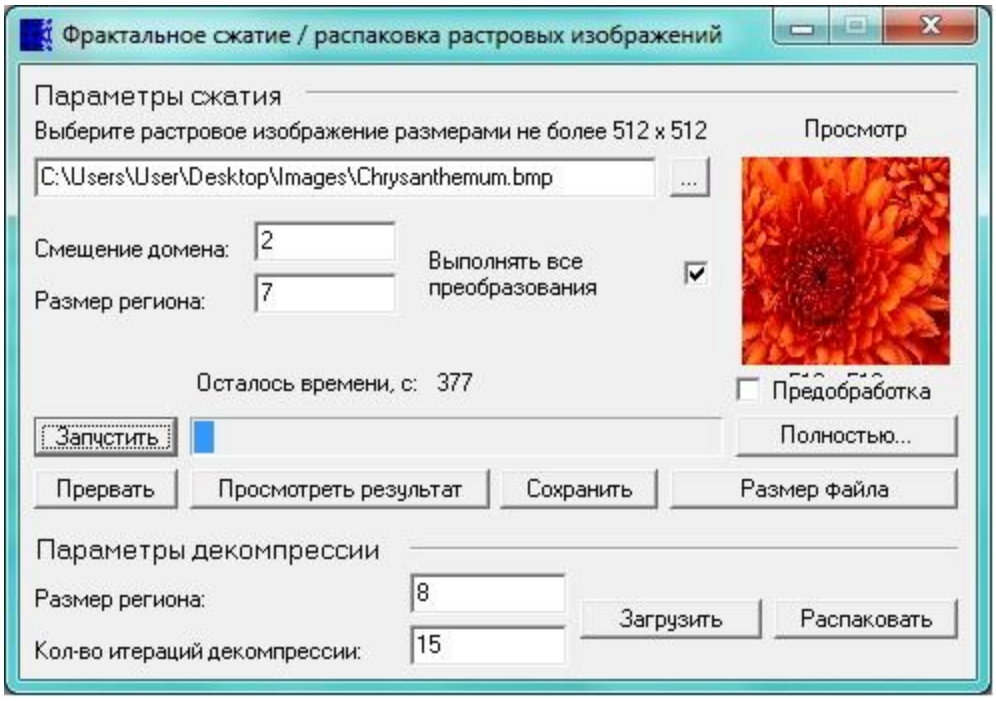


Рисунок 3.5. Процесс сжатия изображения

Рисунок 3.6. Вывод нового размера изображения

Рисунок 3.7.Исходное изображение

Рисунок 3.8. Обработанное изображение

Выводы по третьей главе.

Третья глава посвящена разработке программного обеспечения, решающее задачи сжатия изображения фрактальным методом. В этой главе был приведен сравнительный анализ всех существующих технологий для создания программных продуктов в сфере обработки данных. Изложена роль, а так же место выбранной среды программирования Delphi 2007. Изучены принципы создания программного обеспечения разного рода деятельности. В конце главы приведено описание и назначение созданного программного обеспечения, при процессе выполнения поставленной цели и задачи в магистерской диссертации.

Заключение

В результате проведенных исследований и экспериментов вытекают следующие соображения.

Представляется обоснованным, что эффективность сжатия в немалой степени зависит от начального разбиения исходного изображения. Действительно, не включение удачного домена или его ошибочное отбрасывание сказывается как на степени сжатия, так и на качестве декодирования. Априори представляется правдоподобным, что различие соседних близких пикселей, в целом, менее разительно, чем в областях, выбираемых случайным образом. По этой причине, включение окаймляющих доменов в первоначально формируемое их множество представляется перспективным.

Идея введения упрощенного набора критериев с целью существенного снижения объема вычислений является привлекательной, но в большинстве случаев сравнения доменов с рангами этот набор не позволяет выявить оптимальный домен. То есть данный набор является не вполне адекватным основному критерию. Поэтому представляется целесообразным вместо этого набора использовать основной критерий, но применять его к уменьшенным копиям рассматриваемых пар домен-ранг.

Напрашивается также соображение, непосредственно вытекающее из теоретических основ фрактальных методов сжатия. Так, из теоремы коллажа следует, что степень отличия аттрактора системы сжимающих отображений от исходного изображения не в последнюю очередь определяется качественными характеристиками формируемых отображений. Их повышения естественно ожидать при переходе от простейших, грубых, линейных моделей, используемых в базовых алгоритмах, к более сложным — нелинейным. В этой связи попытка использования таких моделей представляется актуальным.

Практическое значение работы заключаются в разработанном программном обеспечении, реализующего два базовых алгоритма и позволяющего проводить автоматизированный сопоставительный анализ работы реализованных алгоритмов на любых изображениях. На стадии разработки находится программная реализация новых модификаций алгоритма на основе полученных выводов и выдвинутых предложений.

Список использованной литературы

1. Постановление Президента Республики Узбекистан «О мерах по совершенствованию координации и управления развитием науки и технологий»
2. Постановление Президента Республики Узбекистан ПП-1942 «О мерах по дальнейшему совершенствованию системы подготовки кадров в области информационно-коммуникационных технологий» от 26 марта 2013 года
3. И.А. Каримов «Узбекистан на пороге XXI века: угрозы безопасности, условия и гарантии прогресса» 1997 г.
4. И.А. Каримов «Узбекистан на пороге достижения независимости» 2011г.
5. ИКТ в реальный сектор экономики Узбекистана: развитие отраслей экономики и расширение спроса на рынке для разработчиков информационных систем // Infocom.uz, 2014, №04. - С.12-16.
6. Постановление президента республики Узбекистан «О мерах по дальнейшему внедрению информационно-коммуникационных технологий в реальном секторе экономики» 3 апреля 2014 года №ПП-2158. http://www.lex.uz/Pages/GetAct.aspx?lact_id=1986811
7. Закон Республики Узбекистан «Об информатизации», 11 декабря 2003 г., № 560-II. http://lex.uz/pages/getpage.aspx?lact_id=82956
8. G.K.Wallace. «The JPEG still picture compression standard.» Communication of ACM. Volume 34. Number 4 April 1991.
9. B.Smith, L.Rowe. “Algorithm for manipulating compressed images.” Computer Graphics and applications. September 1993.
10. A. Jacquin, «Fractal image coding based on a theory of iterated contractive image transformations», Visual Comm. and Image Processing, vol. SPIE-1360, 1990.

11. Y. Fisher, “Fractal image compression”, SigGraph 92.

12. “Progressive Bi-level Image Compression, Revision 4.1”, ISO/IEC JTC1/SC2/WG9, CD 11544, September 16, 1991.

13. W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, R.B. Arps, “An overview of the basic principles of the Q-coder adaptive binary arithmetic coder”, IBM Journal of research and development, Vol.32, No.6, November 1988, pp. 771-726.

14. D.A. Huffman, “A method for the construction of minimum redundancy codes.” In processing. IRE val.40 1962 pp. 1098-1101.

15. Standardisation of Group 3 Facsimile apparatus for document transmission. CCITT Recommendations. Fascicle VII.2. T.4. 1980.

16. С.В Яблонский “Введение в дискретную математику”. М. “Наука”, 1986. Раздел “Теория кодирования”.

17. А.С. Климов. “Форматы графических файлов”. С.-Петербург, Изд. “ДиаСофт” 1995.

18. Д.С. Ватолин. “Сжатие статических изображений” Открытые системы сегодня. Номер 8 (29) Апрель 1995

19. Д.С. Ватолин. “Применение фракталов в машинной графике” ComputerWorld-Россия. Номер 15. 12 декабря 1995

20. Д.С. Ватолин. “Тенденции развития алгоритмов архивации графики” Открытые системы. Номер 4. Зима 1995

21. Д.С. Ватолин. “Фрактальное сжатие изображений” ComputerWorld-Россия. Номер 6 (23). 20 февраля 1996

22. В.Ю. Романов “Популярные форматы файлов для хранения графических изображений на IBM PC”, Москва “Унитех”, 1992

23. Прэт У. Цифровая обработка изображений. – Москва: Мир, 1982.

24. Vitorino J.C. Ramos, Fernando Muge. Image Colour Segmentation by Genetic Algorithms. Accepted in RecPad'2000 // Proc. of the 11th Portuguese Conf. on Pattern Recognition, Porto, 11 – 12, May, 2000.

25. Delibasis K., Undrill P.E., Cameron G.G. Designing Texture Filters with Genetic Algorithms: an Application to Medical Images. – Department of Biomedical Physics and Bioengineering // University of Aberdeen, Foresterhill, Aberdeen, Scotland, UK // citeceer.nj.nec.com

26. Бубличенко, А.В. Алгоритмы сжатия изображений: сравнительный анализ и модификации / А.В. Бубличенко, В.Н. Беловодский / Квалификационная работа магистра. – 2008.

27. Илюшин С.В. Фрактальное сжатие телемедицинских изображений / С.В. Илюшин, С.Д. Свет / «Электросвязь», №4 – 2009.

28. Прохоров, В.Г. Использование карт Кохонена для ускорения фрактального сжатия изображений / В.Г Прохоров / Прикладне програмне забезпечення, №2. - 2009. – С. 7.

29. Умняшкин, С.В. Математические методы и алгоритмы цифровой компрессии изображений с использованием ортогональных преобразований / С.В. Умняшкин / Автореферат диссертации. – 2001.

30. Hassaballah, M. A fast fractal image compression method based / M. Hassaballah, M.M. Makky and Youssef B. Mahdy / Electronic Letters on Computer Vision and Image Analysis 5(1). –2005. C. 30-40.

31. A.E. Jacquin. Image coding based on a fractal theory of iterated contractive image transforma-tions. // IEEE Trans. Image Processing 1 18-30, 1992;

32. Fractal Image Compression — Theory and Appli-cation to Digital Images. Y. Fisher (ed.). Springer-Verlag, New York 1994;

33. E. Reusens, Partitioning complexity issue for It-erated Function Systems based image coding, in Proc. of VII European Signal Processing Confer-ence, Vol. 1, Edinburg, U.K., September 1994, pp. 171-174

34. F. Davoine, M. Antonini, J.-M. Chassery, M. Bar-laud, Fractal image compression based on Delau-nay triangulation and vector quantization, in Proc. of IEEE Transaction on Image Procession, Vol. 5, No. 2, February 1996, pp. 338-346;

35. F. Davoine, J. Svensson, J.-M. Chassery, A mixed triangular and quadrilateral partition for fractal image coding, in Proc. of IEEE Interna-tional Conference on Image Processing, Washington, D.C, 1

36. L. Thomas, F. Deravi, Region-based fractal im-age compression using heuristic search, in Proc. of IEEE Transaction on Image Procession, Vol. 4, No. 6, June 1995, pp. 832-838;

37. M. Ruhl, H. Hartenstein, D. Saupe, Adaptive par-titioning for fractal image compression, in Proc. of IEEE International Conference on Image Proc-essing, Santa Barbara, October 1997;

38. D. Saupe, M. Ruhl, R. Hamzaoui, L. Grandi, D. Marini, Optimal hierarchical partitions for fractal image compression, in Proc. of IEEE International Conference on Image Processing, Chicago, Octo-ber 1998;

39. Д.С. Ватолин. Гибридная схема фрактальной компрессии и квантования векторов для малых блоков. // Материалы Graphicon-98, 1998;

40. Д.С. Ватолин. Тенденции развития алгорит-мов архивации графики. // Открытые системы, N-4. Зима 1995;

41. Д.С. Ватолин. Использование ДКП для ускорения фрактального сжатия изображений. // Программирование, N 4 1999.

42. Стив Тейксейра, Ксавье Пачеко Delphi 5. Руководство разработчика. Том 1. Основные методы и технологии программирования Вильямс, ISBN 5-8459-0016-6 2008 Вильямс

43. Стив Тейксейра и Ксавье Пачеко Delphi 5. Руководство разработчика. Том 2. Разработка компонентов и работа с базами данных 2010 Вильямс

44. Конопка Рей Создание оригинальных компонент в среде Delphi: Пер. с англ./Рей Конопка. К.: НИПФ - "ДиаСофт Лтд.", 1996. - 512 с. ISBN 5-7707-9551-4

45. Том Сван "Секреты 32-разрядного программирования в Delphi" Диалектика, Киев, 1997. 480 стр., ISBN 966-506-052-X

Приложение