МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Самарский национальный исследовательский университет

имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра информационных систем и технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Исследование алгоритмов фрактального сжатия изображений»

по направлению подготовки 09.04.01 Информатика

и вычислительная техника

(уровень магистратуры)

профиль «Программное обеспечение мобильных устройств»

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Б. Сахибназарова

(подпись, дата)

Руководитель ВКР,

степень, звание\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.А. Кудрина

(подпись, дата)

Консультант *(при наличии)*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.О. Фамилия

(подпись, дата)

Самара 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Самарский национальный исследовательский университет

имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра информационных систем и технологий

УТВЕРЖДАЮ

Заведующий кафедрой ИСТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.А. Прохоров

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ МАГИСТРА

студенту(ке) \_\_\_\_\_\_\_Сахибназаровой Виктории Бахтиёровне\_\_\_\_\_\_\_\_\_\_\_\_\_

группа №\_\_6222-090401D\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема работы:\_Исследование алгоритмов фрактального сжатия изображений\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Исходные данные к работе:

1) объект исследования: алгоритмы фрактального сжатия изображений; \_\_

2) язык программирования: С#;

3) среда программирования Microsoft Visual Studio 2015.

4) операционная система: Windows

Перечень вопросов, подлежащих разработке в работе:

1) анализ предметной обрасти: изучение основного алгоритма фрактального сжатия изображения; изучение различных вариантов реализации основного алгоритма; изучение модификаций алгоритма, направленных на повышение скорости фрактального сжатия;

2) разработка проекта системы по методологии UML;

4) разработка программного обеспечения, реализующего алгоритмы фрактального сжатия; проведение тестирования и отладки;

5) исследование зависимости скорости сжатия исходного изображения и качества декодируемого изображения от использованного алгоритма;

6) оформление документации ВКР.

Руководитель работы

доцент, к.т.н., кафедра ИСТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.А. Кудрина

(подпись)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_\_г.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Б. Сахибназарова

(подпись)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_\_г.

**Реферат**

Пояснительная записка 62 страницы, 24 рисунка, 13 таблиц, 20 источников, 1 приложение.

Презентация: 16 слайдов Microsoft Power Point.

Фрактальное сжатия изображений, компрессия, декомпрессия, ранговый блок, доменный блок, среднеквадратическое отклонение

Целью выпускной работы является исследование вариантов реализации классического алгоритма фрактального сжатия изображений, а также изучение методов повышения скорости фрактального сжатия.

В рамках данной работы проведено исследование зависимости времени сжатия изображения от используемого варианта алгоритма, от примененного метода ускорения, от входных параметров сжатия и от типа изображения. Сравниваются закономерности скорости выполнения алгоритмов для изображений в оттенках серого и цветных изображений. Выявляется зависимость качества декодируемого изображения от параметров примененного алгоритма. Определяется наиболее эффективное сочетание варианта реализации алгоритма фрактального сжатия и метода ускорения для разных типов изображений.

В качестве меры сравнения разных блоков изображения во всех алгоритмах используется среднеквадратическое отклонение значений цветовых компонент пикселей. < - Надо ли?

Что-нибудь про типы исследуемых изображений?

Разработан логический проект системы в среде StarUML 5.0.

Система реализована с помощью средств языка программирования C# в среде разработки Microsoft Visual Studio Community 2015.

**сожержание**

[Введение 6](#_Toc512171951)

[1 Описание и анализ предметной области 8](#_Toc512171952)

[1.1 Математическое обоснование фрактального сжатия изображений 8](#_Toc512171953)

[1.2 Математические формулы преобразования блоков изображений 9](#_Toc512171954)

[1.2.1 Аффинные преобразования 9](#_Toc512171955)

[1.2.2 Яркостные преобразования 11](#_Toc512171956)

[1.3 Классический алгоритм фрактального сжатия 12](#_Toc512171957)

[1.4 Алгоритм декомпрессии 13](#_Toc512171958)

[2 Модификации алгоритма фрактального сжатия 15](#_Toc512171959)

[2.1 Основной алгоритм 15](#_Toc512171960)

[2.1.1 Алгоритм А1 17](#_Toc512171961)

[2.1.2 Алгоритм А2 20](#_Toc512171962)

[2.1.3 Алгоритм Б 22](#_Toc512171963)

[2.2 Методы повышения скорости выполнения фрактального сжатия изображения 24](#_Toc512171964)

[2.2.1 Предварительная классификация блоков 24](#_Toc512171965)

[2.2.2 Метод эталонного блока 24](#_Toc512171966)

[3 Реализация системы 27](#_Toc512171967)

[3.1 Информационный проект системы 27](#_Toc512171968)

[4 Исследования 28](#_Toc512171969)

[4.1 Исследование 1 28](#_Toc512171970)

[Заключение 29](#_Toc512171971)

[Список использованных источников 30](#_Toc512171972)

[Приложение А 31](#_Toc512171973)

Введение

В настоящее время сложно представить себе область деятельности человека, не включающую в себя, хоть в малой степени, необходимость обмена информацией по сети Интернет. При использовании сети важно учитывать два критерия: скорость передачи информации и объем передаваемых данных. Необходимо передать как можно больше информации в сообщении наименьшего размера. В случае передачи графической информации используются различные методы сжатия изображений для уменьшения объема передаваемых данных.

В данной работе рассматривается алгоритм фрактального сжатия изображений, основанный на использовании системы итерируемых функций Iterated Function System (IFS).

Применение IFS к построению фрактальных изображений, стало результатом исследований Майкла Барнсли. Метод базируется на самоподобии элементов изображения и заключается в моделировании рисунка несколькими меньшими фрагментами его самого. Специальные уравнения позволяют переносить, поворачивать и изменять масштаб участков изображения; таким образом, эти участки служат компоновочными блоками остальной части картины[1@1].

Сама система итерируемых функций представляет собой набор трехмерных аффинных преобразований, переводящих одно изображение в другое. Преобразованию подвергаются точки в трехмерном пространстве (X координата, Y координата, яркость) [1]. Примером изображения, основанном на IFS-системе, является чёрный папоротник, в котором каждый лист в действительности представляет собой миниатюрный вариант самого папоротника.

Дальнейшие исследования были направлены на поиск метода, позволяющего находить для любого изображения систему аффинных преобразований, воспроизводящую изображение с заданной точностью.

Первым решение данной задачи нашёл студент Барнсли, Арно Жакан (Arnaud Jacquin). Предложенный метод получил название «Система итерируемых кусочно-определённых функций» (Partitioned Iterated Function System – PIFS) [1%]. Согласно его схеме, отдельные части изображения подобны не всему изображению, а только его частям.

Именно предложенное решение положило начало алгоритму фрактального сжатия, известному сегодня. Согласно ему для осуществления фрактального сжатия (или фрактальное компрессии) исходное изображение делится на подобласти, которые представляют из себя квадраты, называемые *ранговыми блоками*. Ранговые блоки пересекаться не могут. Также на исходном изображении выделяют *доменные блоки* (домены)– являющиеся совокупностью 4-х ранговых блоков. Домены могут пересекаться. Все ранговые блоки и домены – это квадраты со сторонами, параллельными сторонам исходного изображения. И затем, для каждого рангового блока ищется соответствующий ему доменный блок.

Достоинствами фрактальной компрессии являются степень сжатия на уровне JPEG при сравнительно одинаковом качестве, быстрый процесс декодирования и независимость восстанавливаемого изображения от разрешения (хранится структура изображения, а не данные о пикселях). Недостатками являются большие временные затраты сжатия и невозможность гарантировать ту или иную степень потерь (качество декодированного изображения зависит от самоподобия сжимаемого).

Данная работа направлена на изучение различных вариантов реализации алгоритма фрактального сжатия изображений, исследование подходов, позволяющих увеличить скорость фрактального сжатия и выявление зависимости качества декодируемого изображения от параметров примененного алгоритма.

Описание и анализ предметной области

Математическое обоснование фрактального сжатия изображений

Есть отображение , где – множество всех возможных изображений. *W* является объединением отображений *wi*:

где *R* – изображение, а *di* – какие-то (возможно, перекрывающиеся) области изображения *D*. Каждое преобразование *wi* переводит *di* в *ri*. Таким образом:

Представим изображение в виде функции двух переменных *f (x, y)*. На множестве всех таких функций введём метрику (расстояние между изображениями) следующим образом:

Согласно теореме Банаха, существует определённый класс отображений, для которых существует константа 0 *c 1* такая, что для любых изображений *f* и *g* выполняется неравенство

Такие отображения называются *сжимающими*, и для них справедливо следующее утверждение:

Если к какому-то изображению *F0* мы начнём многократно применять отображение *W* таким образом, что

то в пределе, при *i*, стремящемся к бесконечности, мы получим одно и то же изображение вне зависимости от того, какое изображение мы взяли в качестве *F0*:

Это конечное изображение *F* называют *аттрактором*, или *неподвижной точкой отображения W*. Также известно, что если преобразования *wi* являются сжимающими, то их объединение *W* тоже является сжимающим.

Как будет описано далее, процесс декодирования изображения будет осуществляться именно путем многократно применения отображения *W* к базовому изображению. Именно наличие аттрактора у отображения *W* позволяет при декомпрессии в качестве базового использовать любой изображение.

Математические формулы преобразования блоков изображений

* + 1. Аффинные преобразования

Аффинное преобразование − отображение плоскости или пространства в себя, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся − в пересекающиеся, скрещивающиеся − в скрещивающиеся [afin].

В общем виде аффинные преобразования на плоскости описываются следующими формулами:

где A, B, C, D, E, F – некие константы, (x,y) - координаты точки на плоскости до преобразования, (X,Y) – координаты точки на плоскости после преобразования. Преобразование (Ь) можно записать в матричной форме:

Аффинным преобразованием, применяемым мной в данной работе является поворот на градусов. Такое преобразование выражается формулами:

Каждый из приведенных далее алгоритмов фрактального сжатия включает применение 8 аффинных преобразований к доменными блоками.

Примеры этих преобразований можно увидеть в таблице 1.

Таблица 1 – Аффинные преобразования доменного блока

|  |  |  |
| --- | --- | --- |
| № | Название | Пример |
| 1 | Поворот на 0о |  |
| 2 | Поворот на 90о |  |
| 3 | Поворот на 180о |  |
| 4 | Поворот на 270о |  |
| 5 | Вертикальное отражение |  |
| 6 | Горизонтальное отражение |  |
| 7 | Поворот на 90о и вертикальное отражение |  |
| 8 | Поворот на 90о и горизонтальное отражение |  |

* + 1. Яркостные преобразования

При поиске подходящего доменного блока для достижения максимального соответствия блоков часто нам необходимо не только поворачивать доменный блок, но также и изменять его яркость, контрастность или, в случае сжатия цветного изображения, оттенок.

Согласно цветовой модели RGB, цвет пикселя представлен тремя компонентами: красной, зеленой и синей. Значения компонент находятся в диапазоне от 0 до 255. В случае сжатия цветного изображения, соответствующий доменный блок ищется для каждой цветовой составляющей рангового блока отдельно. Если же сжимаемое изображение представлено в оттенках серого, т.е. все его цветовые компоненты имеют одинаковое значение, то для успешного сжатия достаточно найти подходящий доменный блок для одной цветовой составляющей рангового блока.

Для изменения цветовых составляющих блока используют *контрастность s* и *яркость o* – яркостные характеристики преобразования доменного блока к ранговому блоку.

Оптимальные контрастность и яркость минимизируют выражение

в котором и это соответственно значения цветовых компонент пискелей ранговой и доменной областей, а – длина стороны рангового и доменного блоков. Сами яркость и контрастность вычисляются по формулам

где

и – размер стороны рангового (доменного) блока.

Классический алгоритм фрактального сжатия

По своей сути, фрактальное сжатие (или фрактальная компрессия) − это процесс поиска самоподобных областей изображения и определения для них параметров аффинных и яркостых преобразований.

Для реализации алгоритма компрессии необходимо выполнить следующие шаги [1]:

1) исходное изображение разбивается на подобласти, которые представляют из себя квадраты, называемые *ранговыми блоками*. Ранговые блоки пересекаться не могут;

2) на исходном изображении выделяются *домены* – совокупности четырех ранговых блоков. Домены могут пересекаться. Все ранговые блоки и домены – это квадраты со сторонами, параллельными изображению;

3) для каждого рангового блока производится попытка найти на изображении домен, такой чтобы этот домен можно было преобразовать в ранговый блок при помощи аффинных преобразований;

4) перевод домена в ранговый блок производится с помощью поворота домена на 0°, 90°, 180°, 270° и с помощью вертикального и горизонтального зеркальных преобразований;

5) при переводе доменной области в ранговую, ее линейный размер уменьшается в 2 раза;

6) изменение яркости производится кратно некоторому коэффициенту;

7) совпадение преобразованного домена с ранговым блоком может производиться при помощи среднеквадратичного отклонения:

где – точка в домене; – точка в блоке; – пороговое значение «похожести»;

8) если же для некоторого рангового блока не было найдено ни одного удовлетворяющего среднеквадратичному отклонению домена, то ранговый блок разбивается на 4 подобласти, и для каждой из них ищутся домены;

Алгоритм декомпрессии

Для осуществления декомпрессии необходимо задать базовое изображение, k – количество итераций декодирования, фрактальный код.

Алгоритм фрактальной декомпрессии включает следующие шаги:

Задаем значение исходных данных;

1. Из фрактального кода выделяем параметры преобразований для рангового блока;
2. На базовом изображении выделяем соответствующий параметрам доменный блок;
3. Уменьшаем его;
4. Применяем аффинное преобразование;
5. Применяем яркостное преобразование;
6. Копируем преобразованный доменный блок на место текущего рангового блока в базовом изображении;
7. Если декодировали все ранговые блоки, то переходим к п.9, иначе – к п.2;
8. Повторяем данный выполнение данного алгоритма k раз.

Схему алгоритма фрактальной декомпрессии изображения можно видеть на рисунке 1.

s

Копирование преобразованного доменного блока на место рангового блока i

Начало

Конец

*i*  = *i* + 1

Расчет N (количество ранговых блоков)

Базовое изображение

k

Фрактальный код

s

s *>* k

k – количество итераций декодирования

Выделение доменного блока, соответствующего ранговому блоку i

Уменьшение доменного блока

Применение аффинного преобразования к доменному блоку

*i*

s = s + 1

*i*

*i > N*

Применение яркостных преобразований к доменному блоку

Рисунок 1 − Схема алгоритма декомпрессии

1. Модификации алгоритма фрактального сжатия

Основной алгоритм

Степень схожести рангового и доменного блока вычисляется как среднеквадратическое отклонение (СКО):

где – длина стороны рангового и доменного блоков.

С учетом яркостных преобразований, условие (1) принимает следующий вид:

Общий алгоритм фрактального сжатия включает следующие шаги:

1. Исходное изображение разбивается на ранговые блоки;
2. Для каждого рангового блока:
3. Ищется доменный блок, соответствующий ранговому блоку;
4. Сохраняются параметры подошедшего доменного блока;
5. Полученные соответствия между ранговыми и доменными блоками (фрактальный код) сохраняются для дальнейшего использования.

Схема алгоритма представлена на рисунке 1.

Подходящий доменный блок может выбираться несколькими способами:

1. До первого найденного доменного блока, удовлетворяющего условию (1). Если ни один доменный блок не удовлетворяет условию (2):
2. Берем доменный блок с минимальным СКО (алгоритм А1);
3. Разбиваем ранговый блок на 4 блока и для каждого из них ищем подходящий доменный блок (алгоритм А2).
4. Доменный блок с минимальным СКО (алгоритм Б).

Каждый из перечисленных способов включает в себя расчет минимального СКО между ранговым блоком и аффинными преобразованиями доменного блока. Схема расчета представлена на рисунке ?.

*i*

Сохранение параметров преобразования из доменного блока в ранговый блок

Начало

Конец

*i*  = *i* + 1

Расчет N (количество ранговых блоков)

Исходное изображение

Поиск подходящего доменного блока для рангового блока *i*

*i*

*i > N*

Рисунок 1 − Схема общего алгоритма фрактального сжатия

Применение аффинного преобразования *i* к domen

Расчет СКО между rang и domen

Min(afinSKO)

Сохранение СКО в afinSKO[***i***]

Начало

Конец

*i*

i

i *>* 8

*i*  = *i* + 1

rang

domen

rang – ранговый блок

domen – доменный блок

Расчет оптимальных яркостных характеристик преобразования domen к rang

Рисунок 2 − Схема расчета минимального СКО между ранговым блоком и аффинными преобразованиями доменного блока

* + 1. Алгоритм А1

Входные параметры алгоритма: исходное изображение, ранговый блок, коэффициент компрессии ε.

Используемые параметры: *minSKO* (значение СКО, соответствующее минимальному СКО из всех, рассчитанных для заданного рангового блока), *minХ* (координата Х верхнего левого угла доменного блока, соответствующего *minSKO*), *minY* (координата Y верхнего левого угла доменного блока, соответствующего *minSKO*), *minAfin* (номер аффинного преобразования доменного блока, соответствующего *minSKO*).

Шаги алгоритма:

1. Задаем значение исходных данных;
2. Задаем начальные значения для *minSKO*, *minХ*, *minY*, *minAfin*;
3. На исходном изображении выделяем непроверенный доменный блок;
4. Уменьшаем его в 2 раза;
5. Рассчитываем минимальное СКО (*min*) между ранговым блоком и аффинными преобразованиями доменного блока;
6. Если минимальное СКО меньше коэффициента компрессии то сохраняем параметры преобразования текущего доменного блока, иначе – переходим к п.7;
7. Если найденное минимальное СКО меньше значения входного параметра *minSKO* то переходим к п.8, иначе – к п.9;
8. В параметр *minSKO* присваиваем значение *min*, в *minХ*, *minY*, *minAfin* сохраняем соответствующие параметры доменного блока;
9. Если на исходном изображении остались непроверенные доменные блоки, то переходим в п.2, иначе – в п.10;
10. Сохраняем параметры преобразования доменного блока, соответствующего *minSKO*.

Схема алгоритма представлена на рисунке ?.

Выделение доменного блока domen

Уменьшение domen в 2 раза

minSKO = ∞

minX = -1

minY = -1

minAfin = 0

minSKO = min

minX = Х

minY = Y

minAfin

min < ε

Сохранение параметров преобразования domen

Сохранение параметров преобразования доменного блока, соответствующего minSKO

Да

Нет

Да

min < minSKO

Нет

Да

Нет

Начало

Конец

rang

ε

Исходное изображение

rang – ранговый блок

ε - коэффициент компрессии

Расчет min (минимального СКО между ранговым блоком и аффинными преобразованиями доменного блока)

minSKO – минимальное СКО от рангового блока

minx, minY – координаты Х и Y верхнего левого угла доменного блока, имеющего минимальное СКО с ранговым блоком

minAfin – номер аффинного преобразования доменного блока, имеющего минимальное СКО с ранговым блоком

Проверили все доменные блоки?

Рисунок 3 − Схема алгоритма А1

* + 1. Алгоритм А2

Входные параметры алгоритма: исходное изображение, ранговый блок, коэффициент компрессии ε.

Используемые параметры: *minSKO* (значение СКО, соответствующее минимальному СКО из всех, рассчитанных для заданного рангового блока), *minХ* и *minY* (координаты Х и Y верхнего левого угла доменного блока, соответствующего *minSKO*), *minAfin* (номер аффинного преобразования доменного блока, соответствующего *minSKO*).

Шаги алгоритма:

1. Задаем значение исходных данных
2. Задаем начальные значения для *minSKO*, *minХ*, *minY*, *minAfin*;
3. На исходном изображении выделяем непроверенный доменный блок;
4. Уменьшаем его в 2 раза;
5. Рассчитываем минимальное СКО (*min*) между ранговым блоком и аффинными преобразованиями доменного блока;
6. Если минимальное СКО меньше коэффициента компрессии то сохраняем параметры преобразований текущего доменного блока, иначе – переходим к п.7;
7. Если найденное минимальное СКО меньше значения входного параметра *minSKO* то переходим к п.8, иначе – к п.9;
8. В параметр *minSKO* присваиваем значение *min,* в *minХ*, *minY*, *minAfin* сохраняем соответствующие параметры доменного блока;
9. Если на исходном изображении остались непроверенные доменные блоки, то переходим в пункт 2, иначе – в п.10;
10. Если можем разделить ранговый блок на 4 подблока, то переходим к п.11, иначе – п.12;
11. Делим ранговый блок на 4 подблока и для каждого из них выполняем данный алгоритм;
12. Сохраняем параметры преобразований доменного блока, соответствующего *minSKO*.

Схема алгоритма представлена на рисунке ?.

Да

Да

Нет

Выделение доменного блока domen

Уменьшение domen в 2 раза

min < ε

Нет

Да

min < minSKO

Нет

Нет

Да

Деление блок rang на 4 подблока

Начало

Конец

rang

ε

Исходное

изображение

minSKO = ∞

minX = -1

minY = -1

minAfin = 0

rang – ранговый блок

ε - коэффициент компрессии

minSKO – минимальное СКО от рангового блока

minx, minY – координаты Х и Y верхнего левого угла доменного блока, имеющего минимальное СКО с ранговым блоком

minAfin – номер аффинного преобразования доменного блока, имеющего минимальное СКО с ранговым блоком

Проверили все доменные блоки?

Расчет min (минимального СКО между ранговым блоком и аффинными преобразованиями доменного блока)

Выполнение алгоритма А2 для каждого подблока

minSKO = min

minX = Х

minY = Y

minAfin

Сохранение параметров преобразования domen

Сохранение параметров преобразования доменного блока, соответствующего minSKO

Можем разделить rang на подблоки?

Рисунок 4 − Схема алгоритма А2

* + 1. Алгоритм Б

Входные параметры алгоритма: исходное изображение, ранговый блок.

Используемые параметры: *minSKO* (значение СКО, соответствующее минимальному СКО из всех, рассчитанных для заданного рангового блока), *minХ* (координата Х верхнего левого угла доменного блока, соответствующего *minSKO*), *minY* (координата Y верхнего левого угла доменного блока, соответствующего *minSKO*), *minAfin* (номер аффинного преобразования доменного блока, соответствующего *minSKO*).

Шаги алгоритма:

1. Задаем значение исходных данных;
2. Задаем начальные значения для *minSKO*, *minХ*, *minY*, *minAfin*;
3. На исходном изображении выделяем непроверенный доменный блок;
4. Уменьшаем его в 2 раза;
5. Рассчитываем минимальное СКО (*min*) между ранговым блоком и аффинными преобразованиями доменного блока;
6. Если найденное минимальное СКО меньше значения входного параметра *minSKO* то переходим к п.7, иначе – к п.8;
7. В параметр *minSKO* присваиваем значение *min,* в *minХ*, *minY*, *minAfin* сохраняем соответствующие параметры доменного блока;
8. Если на исходном изображении остались непроверенные доменные блоки, то переходим в пункт 2, иначе – в п.9;
9. Сохраняем параметры преобразований доменного блока, соответствующего *minSKO*.

Схема алгоритма представлена на рисунке ?.

Выделение доменного блока domen

Уменьшение domen в 2 раза

minSKO = ∞

minX = -1

minY = -1

minAfin = 0

minSKO = min

minX = Х

minY = Y

minAfin

Сохранение параметров преобразования доменного блока, соответствующего minSKO

min < minSKO

Да

Нет

Начало

Конец

rang

Исходное изображение

rang – ранговый блок

Расчет min (минимального СКО между ранговым блоком и аффинными преобразованиями доменного блока)

minSKO – минимальное СКО от рангового блока

minx, minY – координаты Х и Y верхнего левого угла доменного блока, имеющего минимальное СКО с ранговым блоком

minAfin – номер аффинного преобразования доменного блока, имеющего минимальное СКО с ранговым блоком

i

i *>* N

i

Расчет N (количество ранговых блоков)

*i*  = *i* + 1

Рисунок 5 − Схема алгоритма Б

Методы повышения скорости выполнения фрактального сжатия изображения

* + 1. Предварительная классификация блоков

В случае использования предварительной классификации блоков, прежде чем приступить к выполнению компрессии, каждому ранговому и доменному блоку, выделяемом на изображении, присваивается определенный класс; в дальнейшем, поиск подходящего доменного блока осуществляется только среди доменных блоков, имеющих тот же класс, что и ранговый блок.

В данной работе для классификации блоков я использую следующие подходы:

1. Классификация по значению центра массы блока;
2. Классификация по разнице граничных яркостных значений блока.

Центра масс блока рассчитывается по формуле 1:

в которой N – количество пикселей в блоке, - яркостная характеристика пикселя.

* + 1. Метод эталонного блока

Входные параметры алгоритма: исходное изображение, эталонный блок, коэффициент компрессии ε.

Используемые параметры: *domensSKOs[j][s]* (матрица значений среднеквадратических отклонений афинных преобразований доменного блока от эталонного блока, где - номер доменного блока, - номер аффинного преобразования.

Шаги алгоритма:

1. Задаем значение исходных данных;
2. Рассчитываем СКО между эталонным блоком и каждым аффинным преобразованием всех доменных блоков; сохраняем рассчитанные значения в матрицу *domensSKOs*;
3. Выделяем ранговый блок;
4. Рассчитываем *rangSKO* (значение СКО между эталонным и ранговым блоками);
5. Рассчитываем *min* (минимальный модуль разницы между *rangSKO* и значениями матрицы *domensSKOs*);
6. Если *min* меньше ε, то переходим к п.7, иначе – к п.8;
7. Сохраняем параметры преобразования доменного блока, соответствующего *min*; переходим к п.9;
8. Для текущего доменного блока выполняем алгоритм А1, А2 или Б;
9. Если закодированы не все ранговые блок, то переходим к п.3, иначе – заканчиваем выполнение алгоритма.

Схема алгоритма представлена на рисунке ?.

Да

Нет

Расчет domensSKOs[j][s]

(СКО между эталонным блоком и аффинным преобразованием s доменного блока)

Начало

Конец

etalon – эталонный блок

Исходное изображение

etalon

Расчет N (количество ранговых блоков)

Расчет M (количество доменных блоков)

i

s

s

s *>* 8

i

i *>* M

i

i

i *>* N

Выделение рангового блока i

Расчет rangSKO

(СКО между эталонным блоком и ранговым блоком)

min = Min(|domensSKOs[j][s] - rangSKO |),

i = 1..M,

s = 1..8

min < ε

Сохранение параметров преобразования доменного блока, соответствующего min

Выполнение алгоритма А1 (А2 млм Б) для рангового блока i

s = s + 1

*i*  = *i* + 1

*i*  = *i* + 1

Рисунок 6 − Схема метода эталонного блока

Реализация системы

Информационный проект системы

Проектирование любой программной системы начинается с описания и анализа предметной области. Под предметной областью понимают ту часть реального мира,

Исследования

Исследование 1

Проектирование любой программной системы начинается с описания и анализа предметной области. Под предметной областью понимают ту часть реального мира,

Заключение

пропоо

Список использованных источников

пропоо

Приложение А

Листинг программы

пропоо