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,

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

(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](#_Toc514533882)

[1 Описание и анализ предметной области 8](#_Toc514533883)

[1.1 Математическое обоснование фрактального сжатия изображений 8](#_Toc514533884)

[1.2 Математические формулы преобразования блоков изображений 9](#_Toc514533885)

[1.2.1 Аффинные преобразования 9](#_Toc514533886)

[1.2.2 Яркостные преобразования 11](#_Toc514533887)

[1.2.3 Цветовые преобразования 12](#_Toc514533888)

[1.3 Классический алгоритм фрактального сжатия 13](#_Toc514533889)

[1.4 Алгоритм декомпрессии 14](#_Toc514533890)

[2 Модификации алгоритма фрактального сжатия изображений 16](#_Toc514533891)

[2.1 Основной алгоритм фрактального сжатия 16](#_Toc514533892)

[2.1.1 Первый подходящий доменный блок без разбиения 18](#_Toc514533893)

[2.1.2 Первый подходящий доменный блок с разбиением 21](#_Toc514533894)

[2.1.3 Доменный блок с минимальным СКО 23](#_Toc514533895)

[2.2 Методы повышения скорости выполнения фрактального сжатия изображений 25](#_Toc514533896)

[2.2.1 Предварительная классификация блоков 25](#_Toc514533897)

[2.2.2 Метод эталонного блока 25](#_Toc514533898)

[3 Реализация системы 28](#_Toc514533899)

[3.1 Информационно-логический проект системы 28](#_Toc514533900)

[3.2 Программная реализация системы 31](#_Toc514533901)

[3.3 Выбор и обоснование комплекса программных средств 33](#_Toc514533902)

[3.3.1 Выбор операционной системы 33](#_Toc514533903)

[3.3.2 Выбор языка программирования и средства разработки 34](#_Toc514533904)

[4 Исследования 35](#_Toc514533905)

[4.1 Исследуемые параметры 35](#_Toc514533906)

[4.2 Исследование над изображениями типа «Портрет» 37](#_Toc514533907)

[4.3 Исследование над изображениями с маленьким количеством деталей 40](#_Toc514533908)

[Заключение 41](#_Toc514533909)

[Список использованных источников 42](#_Toc514533910)

[Приложение А 43](#_Toc514533911)

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 | Формулы | Пример |
| 1 | Поворот на 0о |  |  |
| 2 | Поворот на 90о |  |  |
| 3 | Поворот на 180о |  |  |
| 4 | Поворот на 270о |  |  |
| 5 | Отражение относительно оси Y |  |  |
| 6 | Отражение относительно оси Х |  |  |
| 7 | Поворот на 90о и отражение относительно оси Y |  |  |
| 8 | Поворот на 90о и отражение относительно оси Х |  |  |

* + 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.

* + 1. Цветовые преобразования

В данной работе алгоритмы фрактального сжатия применяются к двум типам изображений: изображение в оттенках серого и цветное изображение. Цветное изображение представляется двумя цветовыми моделями: RGB и YIQ.

В случае изображения в градациях серого все три цветовые компоненты пикселя имеют одинаковое значение, вычисляющееся по формуле [?]:

где , и - значения красной, зеленой и синей компонент пикселя.

В цветовой модели YIQ цвет представляется тремя компонентами: Y - яркостная компонента, I и Q - цветоразностные компоненты [?]. Яркостная компонента содержит в оттенках серого изображение в оттенках серого, а оставшиеся две компоненты содержат информацию для восстановления требуемого цвета.

Для перехода от цветовой модели RGB к YIQ используют формулы:

Для возвращения от цветовой модели YIQ к RGB применяют преобразования:

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:

1. задаем значение исходных данных;
2. из фрактального кода выделяем параметры преобразований для рангового блока;
3. на базовом изображении выделяем соответствующий параметрам доменный блок;
4. уменьшаем его;
5. применяем аффинное преобразование;
6. применяем яркостное преобразование;
7. копируем преобразованный доменный блок на место текущего рангового блока в базовом изображении;
8. если декодировали все ранговые блоки, то переходим к п.9, иначе – к п.2;
9. 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 изображений

Основной алгоритм фрактального сжатия

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 (1):
   1. Берем доменный блок с минимальным СКО (алгоритм А1);
   2. Разбиваем ранговый блок на 4 блока и для каждого из них ищем подходящий доменный блок (алгоритм А2).
2. 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. Первый подходящий доменный блок без разбиения

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. Первый подходящий доменный блок с разбиением

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. Доменный блок с минимальным СКО

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 изображений

* + 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:

в которой N – количество пикселей в блоке, – значение цветовой компоненты пикселя.

Разница граничных яркостных значений блока вычисляется по формуле:

Так как значения компонент находятся в диапазоне от 0 до 255 и формула 3 и формула 4 будут возвращать значений в том же диапазоне.

В общем случае данный диапазон можно разбить на 5 интервалов, каждый из которых будет соответствовать одному классу. После вычисления ЦМ или РГЗ для рангового или доменного блока, в зависимости от полученного значения, блоку присваивается соответствующий класс.

* + 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

Информационно-логический проект системы

В данной работе для описания информационно логического проекта системы используется язык UML.

Унифицированный язык моделирования (Unified Modeling Language – UML) – это стандартный инструмент для разработки «чертежей» программного обеспечения. Его можно использовать для визуализации, спецификации, конструирования и документирования артефактов программных систем. UML подходит для моделирования любых систем – от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени [юмл]. Преимуществом методологии UML является объектно-ориентированное представление моделей системы.

Для разрабатываемой системы далее представлены следующие диаграммы, предоставляемые нотацией языка UML: *диаграмма вариантов использования* представляющая собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм и *диаграмма состояний,* предназначенная для моделирования поведения системы.

Визуальное моделирование начинается с модели в форме так называемой диаграммы вариантов использования (*use case diagram*), которая описывает функциональное назначение системы и является исходным концептуальным представлением в процессе ее проектирования и разработки. На ней изображаются отношения между актерами и вариантами использования.

*Актер* (actor) – согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними (это может быть любой объект, субъект или система, взаимодействующая с моделируемой бизнес-системой извне, т.е. человек, техническое устройство, программа и т.п.).

*Вариант использования* – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами (он определяет набор действий, совершаемый системой при диалоге с актером).

На рисунке 7 приведена диаграмма вариантов использования для разрабатываемой системы.



Рисунок 7 − Диаграмма вариантов использования системы

Исходя из приведенной диаграммы, система должна позволять пользователю выбрать сжимаемое изображение, выбрать его тип (цветное или в оттенках серого) и цветовую модель (RGB или YIQ). Пользователь системы должен иметь возможность настраивать параметры компрессии (выбрав алгоритм сжатия, тип классификации, коэффициент компрессии и размер рангового блока) и декомпрессии (выбрав базовое изображение для декомпрессии и количество итераций).

*Диаграмма состояний* (statechart diagram)  это диаграмма, главное предназначение которой  описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла [юмл11].

По существу диаграмма состояний является ориентированным графом специального вида, который представляет некоторый конечный автомат. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние.

Поведение *(behavior)* является спецификацией того, как экземпляр классификатора изменяет значения отдельных характеристик в течение своего времени жизни [юмл12].

*Состояние (state)* – элемент модели поведения, предназначенный для представления ситуации, в ходе которой поддерживается некоторое условие инварианта.

*Переход (*transition*)* является направленным отношением между двумя состояниями, одно из которых является вершиной *источником* (source vertex), а другое – *целевой вершиной* (target vertex).

Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы более детального представления отдельных элементов модели.

На рисунке 9 приведена диаграмма состояний для разрабатываемой системы. После начального состояния система переходит в состояние «Запуск программы», в котором открывается главная форма приложения. Далее система переходит в состояние «Ожидание действий пользователя».

В зависимости от них система оперирует изображением (состояние «Открытие диалогового окна выбора изображения»), изменяет параметры компрессии (состояние «Изменение параметров компрессии»), или декомпрессии (состояние «Изменение параметров декомпрессии»), производит компрессию и декомпрессию, а также выводит на экран информацию о затраченном времени и качестве декодируемого изображения (состояния «Вывод времени компрессии на экран», «Вывод времени декомпрессии и SSIM на экран»).



Рисунок 9 − Диаграмма состояний системы

В случае нажатия кнопки «Выход» система переходит в состояние «Завершение работы с программой», в процессе которого закрывается основная форма приложения*.*

Программная реализация системы

Интерфейс разработанной системы можно видеть на рисунке 66. Как видно из рисунка, в верхней части окна программы располагаются панели настроек параметров компрессии и декомпрессии.

Первой представлена группа радио-кнопок и выпадающий список, позволяющий выбрать тип изображения и цветовую модель (рисунок 67).



Рисунок 67 − Панель выбора типа изображения и цветовой модели

Далее располагаются выпадающие списки, позволяющие выбрать алгоритм поиска доменного блока и используемую классификацию (рисунок 68 и рисунок 69).

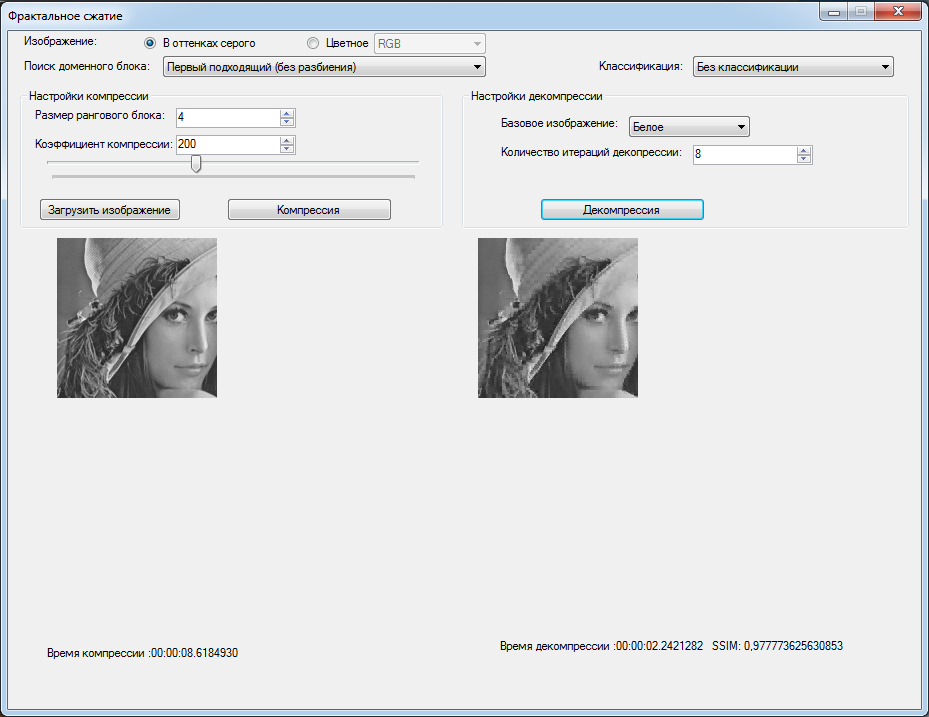


Рисунок 10 − Интерфейс разработанной системы

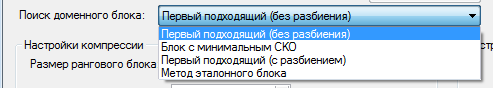


Рисунок 68 − Панель выбора алгоритма поиска доменного блока

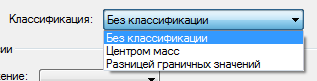


Рисунок 69 − Панель выбора используемой классификации

Далее размещены панели ввода коэффициента компрессии, размера рангового блока (рисунок 70) и параметров декомпрессии (рисунок 71).

За ними располагаются кнопка вызова окна выбора сжимаемого изображения, кнопки запуска процесса компрессии и декомпрессии.

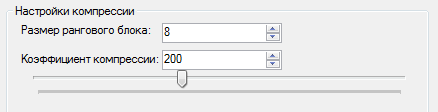


Рисунок 70 − Панель ввода коэффициента компрессии и размера рангового блока

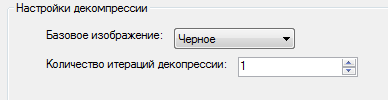


Рисунок 71 − Панель выбора параметров декомпрессии

Под областью настроек параметров на экран выводятся сжимаемое и декодированное изображение, а в нижней части программного окна – время компрессии, время декомпрессии и SSIM (см. рисунок 10).

Выбор и обоснование комплекса программных средств

* + 1. Выбор операционной системы

Операционная система – комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем [16ос]. Выбор ОС зависел от требований, которым должна в оптимальной степени удовлетворять ОС:

* совместимость с выбранной архитектурой ЛВС;
* поддержка выбранного аппаратного обеспечения;
* эффективное управление ресурсами;
* достаточная для решения основных задач производительность;
* масштабируемость;
* совместимость;
* довольно высокая надёжность и отказоустойчивость;
* защита данных от несанкционированного доступа;
* поддержка многозадачности;
* приоритет удобства работы пользователя над скоростью вычислений.

Данным требованиям в той или иной мере соответствуют весьма популярные представители семейства Windows – Windows 7 и Windows 8.

* + 1. Выбор языка программирования и средства разработки

C# является языком программирования, который разработан для создания множества приложений, работающих в среде .NET Framework. Язык C# прост и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую С-подобным языкам [17ос].

Visual C# - это реализация языка C# корпорацией Майкрософт. Поддержка Visual C# в Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и к другим полезным, хорошо спроектированным классам, что существенно ускоряет цикл разработки.

Microsoft Visual Studio Community 2015 - это бесплатная полнофункциональная интегрированная среда разработки с мощными, эффективными возможностями для кодирования, инструментами кроссплатформенных разработок мобильных приложений для Windows, iOS и Android, веб- и облачных приложений, а также доступом к тысячам расширений. Этот выпуск Visual Studio доступен для отдельных разработчиков, для разработки проектов с открытым исходным кодом, академических исследований, образования и небольших групп специалистов [18ос].

Исследования

Исследуемые параметры

Исследуемые параметры можно разделить на 2 типа: параметры компрессии и параметры декомпрессии.

Параметры фрактального сжатия включают в себя *время сжатия*, вычисляемое программной системой, а также *степень сжатия*, которая вычисляется по формуле:

В качестве параметров декомпрессии берутся *время декомпрессии* (измеряемое программной системой) и *качество декодируемого изображения*.

Для оценки качества декодируемого изображения используется метрика SSIM [ssim]. Данная метрика учитывает искажение яркости и контрастности, а также – степень коррелированности между исходным и декодируемым изображением.

Значение SSIM рассчитывается по следующей формуле:

где - яркость, - контрастность, - структура, – матрица значений пикселей оригинального (эталонного) изображения, - матрица значений пикселей оцениваемого изображения, – коэффициенты для регулировки важности составляющих компонент.

Сами компоненты вычисляются по следующим формулам:

где - выравнивающие коэффициенты, предотвращающие деление на число, близкое к 0, при высоком качестве оцениваемого изображения.

где – константа (), - максимальное значение канала. Обычно берут . Для изображений, в которых для хранения значения пикселя используется 8 бит .

Для упрощения выражения (?) берут , а , что приводит к следующему преобразованию:

где – среднее арифметическое значений каналов пикселей изображения , - стандартное отклонение значений каналов пикселей изображения , - ковариация значений каналов пикселей изображений x и y. При этом:

где – значения соответствующих цветовых каналов для пикселей изображений x и y, - размер матриц изображений x и y;

Значения SSIM лежат в диапазоне от -1 до 1, и чем ближе к 1, тем более декодируемое изображение похоже на исходное. Если SSIM равно 1, то два сравниваемых изображения полностью идентичны.

Исследование над изображениями типа «Портрет»

В данном разделе исследования проводятся над набором изображений в оттенках серого, размером 160×160 пикселей.

В таблице ? представлена зависимость параметров компрессии и декомпрессии алгоритма А1 от размера рангового блока и коэффициента компрессии.

Таблица 2 – Зависимость параметров компрессии и декомпрессии алгоритма А1 от размера рангового блока и ε

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размер рангового блока | ε | tкомп, сек | tдекомп, сек | Степень сжатия | SSIM |
| 4 | 10 | 120,67 | 1,71 | 4,39 | 0,9901 |
| 50 | 51,99 | 1,65 | 4,39 | 0,9872 |
| 100 | 25,88 | 1,72 | 4,37 | 0,9833 |
| 150 | 15,54 | 1,72 | 4,37 | 0,9799 |
| 200 | 7,83 | 1,67 | 4,37 | 0,9778 |
| 300 | 2,49 | 1,68 | 4,37 | 0,9730 |
| 8 | 2 | 46,66 | 1,71 | 17,51 | 0,9566 |
| 5 | 39,76 | 2,15 | 17,55 | 0,9564 |
| 10 | 38,33 | 2,04 | 17,55 | 0,9560 |
| 50 | 28,13 | 1,65 | 17,55 | 0,9543 |
| 100 | 14,96 | 1,76 | 17,55 | 0,9517 |
| 150 | 13,32 | 1,72 | 17,55 | 0,9491 |

Из таблицы 4 можно видеть, что с увеличением коэффициента компрессии уменьшается не только время сжатия, но и качество декодируемого изображения. Для дальнейшего исследования для алгоритма А1 размер рангового блока берется равный 4 и коэффициент ε, равный 150.

Таблица 2 – Зависимость параметров компрессии и декомпрессии алгоритмов А1, А2, Б

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Алгоритм выбора доменного блока | Метод классификации | Размер рангового блока | ε | tкомп, сек | tдекомп, сек | Степень сжатия | SSIM |
| Первый подходящий (без разбиения) | - | 4 | 150 | 9,21 | 1,89 | 4,53 | 0,9808 |
| Центр масс | 4 | 150 | 4,06 | 2,01 | 4,49 | 0,9765 |
| Разница граничных значений | 4 | 150 | 2,95 | 2,07 | 4,53 | 0,9773 |
| Первый подходящий (с разбиением) | - | 16 | 5 | 45,08 | 1,83 | 8,79 | 0,9793 |
| Центр масс | 16 | 5 | 16,84 | 1,91 | 8,43 | 0,9797 |
| Разница граничных значений | 16 | 5 | 15,86 | 2,45 | 8,45 | 0,9805 |
| Доменный блок с минимальным СКО | - | 8 | - | 35,72 | 2,15 | 17,67 | 0,9722 |
| Центр масс | 8 | - | 15,75 | 2,13 | 17,71 | 0,9677 |
| Разница граничных значений | 8 | - | 13,43 | 2,21 | 17,72 | 0,9711 |

Как видно из рисунка 33 и предыдущей таблицы, наиболее эффективным с точки зрения затрачиваемого времени и качества декодируемого изображения, будет использование алгоритма А1 с классификацией РГЗ.

Рисунок 33 – Зависимость времени сжатия изображения от алгоритма выбора доменного блока и типа классификации

В таблице 44 приводится исследование зависимости параметров компрессии и декомпрессии метода эталонного блока от размера рангового блока и коэффициента ε.

Таблица 44 – Зависимость параметров компрессии и декомпрессии метода эталонного блока от размера рангового блока и коэффициента ε

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размер рангового блока | ε | tкомп, сек | tдекомп, сек | Степень сжатия | SSIM |
| 4 | 10 | 144,35 | 1,65 | 4,39 | 0,9898 |
| 50 | 84,28 | 1,71 | 4,39 | 0,9860 |
| 100 | 57,08 | 1,73 | 4,42 | 0,9818 |
| 200 | 39,64 | 1,62 | 4,42 | 0,9754 |
| 300 | 34,65 | 1,67 | 4,42 | 0,9719 |
| 400 | 33,46 | 1,73 | 4,42 | 0,9663 |
| 8 | 5 | 53,62 | 1,72 | 17,51 | 0,9564 |
| 10 | 47,25 | 1,83 | 17,51 | 0,9560 |
| 50 | 38,36 | 1,68 | 17,51 | 0,9528 |
| 100 | 25,27 | 1,71 | 17,51 | 0,9503 |
| 150 | 19,73 | 1,65 | 17,55 | 0,9473 |
| 200 | 18,08 | 1,68 | 17,51 | 0,9454 |

Для сравнения метода эталонного блока с алгоритмами А1 и А2(при условии приблизительно равного качества декодируемого изображения) размер рангового блока берется равный 4 и коэффициент ε, равный 300.

Таблица 55 – Зависимость времени сжатия от выбранного подхода

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Алгоритм выбора доменного блока | Метод классификации | Размер рангового блока | ε | tкомп, сек | tдекомп, сек | Степень сжатия | SSIM |
| Первый подходящий (без разбиения) | Разница граничных значений | 4 | 150 | 2,95 | 2,07 | 4,53 | 0,9773 |
| Первый подходящий (с разбиением) | - | 16 | 5 | 45,08 | 1,83 | 8,79 | 0,9793 |
| Первый подходящий (с разбиением) | Разница граничных значений | 16 | 5 | 15,86 | 2,45 | 8,45 | 0,9805 |
| Метод эталонного  блока | - | 4 | 300 | 34,99 | 2,36 | 4,69 | 0,9782 |

Как видно из рисунка 55 и предыдущей таблицы использование классификации РГЗ обеспечивают заметный выигрыш по времени, а метода эталонного блока лишь немного уменьшает время по сравнению с алгоритмом А2. Наиболее эффективным по времени выполнения остается использование поиска первого подходящего доменного блока (алгоритм А1) с классификацией разницей граничных значений.

Рисунок 55 – Зависимость времени сжатия изображения от выбранного алгоритма

Исследование над изображениями с маленьким количеством деталей

В данном разделе исследования проводятся над набором изображений в оттенках серого, размером 160×160 пикселей.

В таблице ? представлена зависимость параметров компрессии и декомпрессии алгоритма А1 от размера рангового блока и коэффициента компрессии.

Таблица 2 – Зависимость параметров компрессии и декомпрессии алгоритма А1 от размера рангового блока и ε

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размер рангового блока | ε | tкомп, сек | tдекомп, сек | Степень сжатия | SSIM |
| 4 | 10 | 155,83 | 1,92 | 4,39 | 0,9854 |
| 50 | 103,41 | 1,85 | 4,39 | 0,9854 |
| 100 | 60,38 | 1,76 | 4,39 | 0,9851 |
| 200 | 22,04 | 1,68 | 4,39 | 0,9841 |
| 300 | 9,64 | 1,71 | 4,39 | 0,9834 |
| 400 | 3,26 | 1,68 | 4,39 | 0,9824 |
| 8 | 5 | 22,21 | 1,69 | 17,51 | 0,9786 |
| 10 | 18,63 | 1,78 | 17,55 | 0,9783 |
| 50 | 14,48 | 1,65 | 17,59 | 0,9771 |
| 100 | 12,36 | 1,99 | 17,55 | 0,9750 |
| 150 | 12,15 | 1,83 | 17,51 | 0,9746 |
| 200 | 10,84 | 1,72 | 17,51 | 0,9745 |

Из таблицы 4 можно видеть, что с увеличением коэффициента компрессии уменьшается не только время сжатия, но и качество декодируемого изображения. Для дальнейшего исследования для алгоритма А1 размер рангового блока берется равный 4 и коэффициент ε, равный 300.

Таблица 2 – Зависимость параметров компрессии и декомпрессии алгоритмов А1, А2, Б

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Алгоритм выбора доменного блока | Метод классификации | Размер рангового блока | ε | tкомп, сек | tдекомп, сек | Степень сжатия | SSIM |
| Первый подходящий (без разбиения) | - | 4 | 300 | 2,12 | 2,50 | 4,38 | 0,9893 |
| Центр масс | 4 | 300 | 1,04 | 2,09 | 4,39 | 0,9902 |
| Разница граничных значений | 4 | 300 | 0,94 | 2,52 | 4,38 | 0,9891 |
| Первый подходящий (с разбиением) | - | 16 | 5 | 27,50 | 2,01 | 21,82 | 0,9886 |
| Центр масс | 16 | 5 | 8,84 | 2,38 | 19,95 | 0,9903 |
| Разница граничных значений | 16 | 5 | 8,90 | 2,17 | 20,19 | 0,9926 |
| Доменный блок с минимальным СКО | - | 8 | - | 46,89 | 2,49 | 17,52 | 0,9921 |
| Центр масс | 8 | - | 14,34 | 2,56 | 17,61 | 0,9887 |
| Разница граничных значений | 8 | - | 30,45 | 2,32 | 17,55 | 0,9891 |

Как видно из рисунка 33 и таблицы 2, наиболее эффективным с точки зрения времени сжатия является алгоритм А1, а поиск доменного блока с наименьшим СКО (алгоритм Б) занимает больше всего времени. Выигрыш по времени алгоритмов А1 и А2 обуславливается отсутствием большого количества деталей в изображениях, что позволяет быстрее найти подходящий доменный блок.

Рисунок 33 – Зависимость времени сжатия изображения от алгоритма и типа классификации.

В таблице 44 приводится исследование зависимости параметров компрессии и декомпрессии метода эталонного блока от размера рангового блока и коэффициента ε.

Таблица 44 – Зависимость параметров компрессии и декомпрессии метода эталонного блока от размера рангового блока и коэффициента ε

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размер рангового блока | ε | tкомп, сек | tдекомп, сек | Степень сжатия | SSIM |
| 4 | 5 | 105,51 | 1,71 | 4,39 | 0,9849 |
| 10 | 89,95 | 1,64 | 4,39 | 0,9854 |
| 50 | 55,15 | 1,72 | 4,39 | 0,9854 |
| 100 | 44,65 | 1,65 | 4,39 | 0,9851 |
| 200 | 20,72 | 1,67 | 4,39 | 0,9841 |
| 300 | 9,17 | 1,68 | 4,39 | 0,9834 |
| 8 | 5 | 22,94 | 1,83 | 17,51 | 0,9786 |
| 10 | 19,59 | 1,79 | 17,55 | 0,9783 |
| 50 | 15,19 | 1,67 | 17,59 | 0,9771 |
| 100 | 13,62 | 1,83 | 17,55 | 0,9750 |
| 200 | 11,01 | 1,76 | 17,51 | 0,9745 |
| 300 | 8,33 | 1,75 | 17,55 | 0,9742 |

Для сравнения метода эталонного блока с алгоритмами А1 и А2(при условии приблизительно равного качества декодируемого изображения) размер рангового блока берется равный 4 и коэффициент ε, равный 300.

Таблица 55 – Зависимость времени сжатия от выбранного подхода

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Алгоритм выбора доменного блока | Метод классификации | Размер рангового блока | ε | tкомп, сек | tдекомп, сек | Степень сжатия | SSIM |
| Первый подходящий (без разбиения) | Разница граничных значений | 4 | 300 | 0,94 | 2,52 | 4,38 | 0,9891 |
| Первый подходящий (с разбиением) | - | 16 | 5 | 27,50 | 2,01 | 21,82 | 0,9886 |
| Первый подходящий (с разбиением) | Центр масс | 16 | 5 | 8,84 | 2,38 | 19,95 | 0,9903 |
| Метод эталонного  блока | - | 4 | 300 | 35,03 | 2,29 | 4,69 | 0,9918 |

Как видно из рисунка 55 и таблицы 55 использование классификации РГЗ обеспечивают заметный выигрыш по времени, а метода эталонного блока для данного типа изображений затрачивает больше времени, чем алгоритм А2. Наиболее эффективным по времени выполнения остается использование поиска первого подходящего доменного блока (алгоритм А1) с классификацией разницей граничных значений.

Рисунок 55 – Зависимость времени сжатия изображения от выбранного алгоритма

Исследование над изображениями с большим количеством деталей

В данном разделе исследования проводятся над набором изображений в оттенках серого, размером 160×160 пикселей.

Заключение

пропоо

Список использованных источников

пропоо

Приложение А

Листинг программы

пропоо