Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ ФРАКТАЛЬНОГО СЖАТИЯ ИЗОБРАЖЕНИЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Павловой Дарьи Сергеевны

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Идея использования метода фрактального сжатия для сжатии графических изображений появилась давно, но на тот момент было практически невозможно построить алгоритм, который за приемлемое время будет подбирать нужные коэффициенты.

Развитие информационных технологий привело к особой популярности цифровых изображений. Для их хранения в цифровом виде необходимы затратные объемы памяти устройства. Значительно снизить расходы на хранение и уменьшить время передачи изображений по каналам связи позволяет сжатие изображений. Поэтому данная тема приобретает особую популярность.

Цель сжатия изображений— уменьшение числа бит, необходимых для его представления. Фрактальное сжатие с помощью коэффициентов системы итерируемых функций представляет изображение в компактной форме. [1]

Целью выпускной квалификационной работы является реализация и сравнительный анализ различных вариантов фрактального сжатия изображений. Для достижения данных целей были поставлены следующие задачи:

- Ознакомиться с различными вариантами ускорения алгоритма
- Разработать приложение, которое реализует различные алгоритмы фрактального сжатия изображений, а именно базовый алгоритм, «квадродерево» и «мозаика»
- Провести сравнительный анализ реализованных алгоритмов

1 Описание исследований

1.1 Алгоритм фрактального сжатия изображений

Способы сжатия цифровых изображений можно разделить на сжатие с потерями и без потерь. Сжатие изображения без потерь гарантирует, что восстановленное изображение с точностью до пикселя будет соответствовать исходному. Однако этот способ не позволяет достичь высоких коэффициентов сжатия изображений. Алгоритмы компрессии с потерями наоборот позволяют без видимого ухудшения качества достигать сжатия до 50 раз. [1]

Основные задумки данного алгоритма принадлежат Барнсли. Первые практические результаты были получены Жаквином [2] в 1992 году. Основой данного алгоритма служит идея подобия между частями изображения. В связи с особенностью фракталов наибольшие коэффициенты достигаются на реальных изображениях. Кроме того восстановленное изображение можно масштабировать, даже при сильном увеличении избегая дробления изображения на квадраты.

Алгоритм фрактального сжатия изображений относится к группе алгоритмов сжатия изображений с потерями. В основе данного алгоритма лежит свойство самоподобия. Сжатие изображений происходит путем многочисленного применения систем итерируемых кусочно-определенных функций, которые иначе говоря являются аффинными преобразованиями частей изображения. [3]

Понятие данных функций, в которых, каждый блок из набора разбиения покрывает не все изображение, а только его часть, ввел Арно Жаквин (Arnaud Jacquin) [2]. После чего фрактальная компрессия получила успех в реализации.

Основным недостатком данного алгоритма является необходимость множества вычислений при компрессии, что приводит к значительному увеличению времени сжатия. Однако при декомпрессии требуется меньше вычислений, чем у JPEG. Данный алгоритм можно считать первым существенно несимметричным алгоритмом.

Алгоритм фрактального сжатия в упрощенной схеме можно разделить на два этапа:

1. формирование пула ранговых и доменных блоков;
2. поиск доменного блока, обеспечивающего приемлемое покрытие рангового путем применения геометрических и аффинных преобразований к

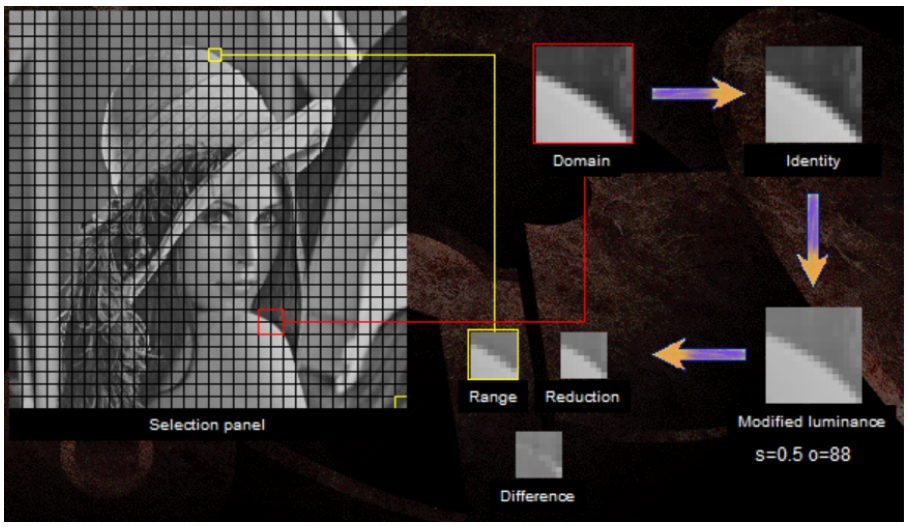

Рисунок 1 – Схема базового алгоритма фрактального сжатия изображений

частям изображения.

Схема данного алгоритма изображена на рис. 1.

1.2 Варианты ускорения алгоритма фрактального сжатия

Основными недостатками данного алгоритма является то, что для сжатия изображения требуется достаточно много времени. Так же степень сжатия во многом зависит от самоподобия изображения, что также влияет на степень искажения восстановленного изображения.

Однако у данного алгоритма есть множество достоинств:

- высокий коэффициент сжатия как и у $JPEG$ алгоритма при сравнительно одинаковом качестве,
- процесс декодирования занимает мало времени,
- декодированное изображение не зависит от разрешения, так как в сжатом изображении хранится информация о самой структуре изображения, а не о пикселях,
- в отличие от алгоритма $JPEG$ искажение изображения, вызванное потерей данных при восстановлении изображения, проявляется в виде размытия, а не в виде высокочастотных шумов.

Выбор метода разбиения От выбранного метода разбиения на ранговые блоки во многом зависит качество сжатия, так как он влияет на наложенные на блоки ограничения, а также определяет их размер и форму. Примерами методов разбиения являются следующие методы:

— разбиение на множество блоков фиксированного размера, которое используется в базовом алгоритме фрактального сжатия,
— разбиение методом квадродерева,
— разбиение при помощи триангуляции.

Переупорядочивание блоков изображения в памяти Дополнительного ускорения можно добиться за счет подготовки памяти к перебору блоков.

Для ускорения процесса сжатия изображения пул доменных и ранговых блоков можно упорядочить так, чтобы ускорить к ним доступ ОЗУ.

При использовании метода разбиения квадродеревом данная оптимизация тоже применима. Для каждого уровня квадродерева в зависимости от размера блока подготавливается доменный пул. [8]

Классификация блоков Дополнительного ускорения можно добиться за счет классификации доменных и ранговых блоков, так как в этом случае произойдет уменьшение перебора блоков.

Доменные блоки классифицируются до начала сжатия изображения. Ранговые блоки классифицируются во время сжатия и сравниваются только с доменными блоками соответствующего класса, либо нескольких близких классов.

Распараллеливание В Осокин А.Н. в своей работе [9] исследовал возможность распараллеливания данного алгоритма. Так же он вывел некоторую зависимость ускорения путем распараллеливания от количества ядер процессора. Например, на одноядерных процессорах Intel с поддержкой технологии HyperThreading ускорение составляет порядка 1,4, а на двухядерных процессорах— 2. [8]

2 Описание разработанного приложения

В данной работе реализовано приложение на языке C++ с использованием технологии Microsoft .NET Framework и инструментальной оболочки Visual Studio 2017. С помощью данного приложения можно сжимать растровые изображения в фрактальные и декодировать их алгоритмами фрактального сжатия с использованием разбиения на ранговые области по алгоритму «IFS», «квадродерево» и «мозаика». Пользователь может выбирать один из предложенных алгоритмов для сжатия изображения.

Также реализованы возможность загрузки изображения— кнопка «Загрузить изображение» в левом нижнем углу, сохранения сжатого изображения в файл с расширением .frac— кнопка «Сохранить как фрактал», сохранение изображения— кнопка «Сохранить как изображение», собственно сжатие по выбранному алгоритму— кнопка «Преобразовать», загрузки сжатого фрактального изображения с компьютера— кнопка «Загрузить фрактал» в правом нижнем углу. Созданное окно приложения содержит два PictureBox для отображения исходного изображения, которое загружается с помощью openSourceDialog, и изображения, восстановленного после сжатия. Один openFileDialog для того, чтобы открывать сжатые изображения с расширением .frac, saveAsBitMapDialog и saveFileDialog для сохранения сжатого изображения в формате .frac и в формате растрового изображения. Элемент progressBar, позволяющий видеть процесс сжатия изображения, который использует в своей работе элемент timer. Элемент comboBox для выбора метода разбиения изображения на ранговые блоки. Пять элементов button для управления приложением.

Функции и объекты для работы с библиотекой Были созданы следующие объекты и функции для работы с библиотекой.

Структура FData для хранения данных библиотеки. Содержит коэффициент аффинного преобразования, размер изображения для ранговых блоков различного размера и само полученное изображение.

Функция FData* CreateData(int index, int Size) создает элемент данных по номеру и размеру преобразования.

Матрица IFS, вектор IFS и точка, а также коэффициент точности сравнения изображений и двумерные массивы для хранения исходного и

преобразованного изображений.

```
double M[3][3];
double V[3];
double P[3];
double P1[3];

int Epsilon = 20;

unsigned char Image1[BlockSize][BlockSize];
unsigned char Image2[BlockSize][BlockSize];
```

Функции для работы с изображениями, векторами и точками, а именно:

- Функция копирования изображений— `void CopyImage`.
- Функция сравнения двух изображений друг с другом с заданной точностью— `bool EqualImages`.
- Функция умножения матрицы `IFS` на вектор `IFS`— `void Multiply`.
- Функция сложения точки P и вектора `IFS V`— `void Add`.
- Функция восстановления `Image2`. Изменение изображения под влиянием точки `P`. Результат в `Image2` (Преобразованное изображение)— `void Recover`.
- Функции определения минимальной, средней и максимальной яркости изображения.
- Функция применения аффинного преобразования по индексу.
- Функции сохранения/загрузки данных— `void SaveData`, `void LoadData`.
- Функция определения расстояние между двумя изображениями. Рассчитывается как сумма модулей разности или сумма квадратов с учетом коррекции яркости изображения— `int Meagure`.
- Функция нахождения ближайшего по конфигурации блока в библиотеке с учетом коррекции яркости (dc)— `int FindNearest`.

Основные функции и объекты В основном файле программы реализованы следующие функции и объекты.

Контейнеры для исходного и восстановленного изображения.

```
Bitmap^ Source = nullptr;
```

```
Bitmap^ Target = nullptr;
```

Класс `xPoint` цветной точки с функциями представления в байтах, записи и чтения, функции, возвращающие/устанавливающие значение цветового канала по индексу.

Класс `Block`, представляющий собой «фрактально сжатый фрагмент изображения». Содержит номера изображений из библиотеки для каждого цветового канала и коррекцию яркости.

Реализация сжатия и распаковки В зависимости от выбранного способа сжатия используется определенная функция сжатия/распаковки.

- Функция базового алгоритма— `ConvertStandart`. Разбиение изображения происходит на ранговые блоки одинакового заданного размера.
- Функция сжатия алгоритмом «квадродерево»— `ConvertQTree`, вспомогательная функция `CQT`, добавляет один или несколько блоков в результат. Сначала генерируется и оценивается «большой» блок. Затем, если погрешность покрытия большая, то добавляются и оцениваются 4 маленьких блока.
- Функция сжатия изображения алгоритмом «мозаика»— `ConvertMosaic`. Сначала конвертируется стандартная сетка, затем перебираются уменьшенные блоки с произвольным положением. Если оценка уменьшенного блока лучше, чем крупных, то добавить этот блок.

ЗАКЛЮЧЕНИЕ

Основными результатами являются сравнительный анализ и программная реализация алгоритмов фрактального сжатия изображений. Для сравнения работы рассмотренных алгоритмов (базовый алгоритм, квадродерева и мозаики) была написана программа на языке C++.

Главным недостатком алгоритмов фрактального сжатия является необходимость большого времени для процесса компрессии изображения, а также требование достаточной вычислительной мощности устройства. Поэтому данный алгоритм практически невозможно использовать в приложениях, где необходимо быстрое сжатие изображений в режиме реального времени. Однако в случае, когда необходимо один раз сжать изображение, а затем передать его по сети и разархивировать множество раз, данный алгоритм обладает большим преимуществом, так как обеспечивает хороший коэффициент сжатия и требует меньше времени для декомпрессии, чем другие современные алгоритмы сжатия изображений. [1]

Определены основные пути ускорения алгоритма фрактального сжатия:

- изменение способа разбиения на ранговые блоки
- адаптация разбиения изображения на блоки к структуре изображения, например, улучшенная схема разбиения квадродерева и мозаика;
- классификация ранговых и доменных блоков с целью ускорения поиска блока для обеспечения приемлемого покрытия
- переупорядочивание блоков в памяти для ускорения доступа к ним;
- распараллеливание алгоритма.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Карачанская, Е. В.* Сравнение фрактальных методов сжатия изображений / Е. В. Карачанская, А. Э. Паздникова // *Ученые заметки ТОГУ*. — 2017. — Т. 8, № 1. — С. 412–420.

2 *Jacquin, A.* A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding: Ph.D. thesis / Georgia Institute of Technology. — 1989.

3 *Уэлстид, С.* Фракталы и вейвлеты для сжатия изображений в действии / С. Уэлстид. — М.: Триумф, 2003.

4 Свойства фракталов. Применение фракталов на рынке [Электронный ресурс]. — URL: `http://study-i.ru/forex/fractal_theory/topic.php?id=svojstva_fraktalov` (Дата обращения 30.05.2018). Загл. с экр. Яз. англ.

5 *Qien, G. E.* A new improved collage theorem with appplications to multiresolution fractal image coding // Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. — Vol. 1. — Adelaide, SA, Australia: 1994. — Pp. 565–568.

6 *Saupe, D.* Fractal image compression — an introductory overview // Fractal Models for Image Synthesis, Compression and Analysis, ACM SIGGRAPH'96 Course Notes 27 / Ed. by D. Saupe, J. Hart. — New Orleans, Louisiana: 1996.

7 Intel Wireless Display Review – No Wires? No Problem [Электронный ресурс]. — URL: `http://www.pcper.com/article.php?` (Дата обращения 25.05.2018). Загл. с экр. Яз. рус.

8 *Осокин, А. Н.* Быстродействующий алгоритм фрактального сжатия изображений / А. Н. Осокин, М. П. Шарабайко // *Известия Томского политехнического университета*. — 2011. — Т. 318.

9 *Осокин, А. Н.* Исследование возможности распараллеливания процесса фрактального сжатия изображений // Молодежь и современные информационные технологии: Сб. трудов VIII Всеросс. научно-практ. конф. студентов, аспирантов и молодых ученых. — Т. 2. — Томск: 2010. — С. 212–213.

10 *Кроновер, Р. М.* Фракталы и хаос в динамических системах / Р. М. Кроновер. — М.: Техносфера, 2006.

11 *Fisher, Y.* Fractal Image Compression: Theory and Application / Y. Fisher. — NY: Springer-Verlag, 1995.

12 *Колебошин, В. Г.* Результаты фрактального сжатия изображений при различных формах ранговых областей / В. Г. Колебошин, А. В. Крапивенко // *Электронный журнал «Труды МАИ»*. — 2009. — Т. 36. — С. 10887.

13 *Прохоров, В. Г.* Использование карт Кохонена для ускорения алгоритма фрактального сжатия изображений / В. Г. Прохоров // *Институт программных систем*. — 2009. — Т. 934.