САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра Системного Программирования

Азимов Рустам Шухратуллович

Оптимизация алгоритма сжатия изображений без потерь

Курсовая работа

Научный руководитель:
к. т. н., доцент Лазарева С. В.

Санкт-Петербург
2015

Оглавление

Введение

В современном мире, во многих областях, например в медицине и астрономии, необходимо хранить большое количество изображений, длительное время, причем без потери качества. Отсюда накладные расходы на хранение информации.

Решить данную проблему позволяют алгоритмы сжатия изображений без потерь. Кроме коэффициента сжатия, характеристикой реализаций данных алгоритмов также является скорость компрессии и декомпрессии. Существуют кодеки с относительной простотой реализации, выигрывающие в скорости, но проигрывающие в эффективности сжатия более сложным алгоритмам.

В данной работе рассматривается новый алгоритм сжатия изображений без потерь, описанный в статье "Lossless Image Compression Scheme with Binary Layers Scanning" [2], теоретически имеющий низкую сложность реализации, но в то же время выигрывающий по эффективности сжатия у аналогов (JPEG-LS [5], JPEG-2000 [4]). Изначальная реализация нового алгоритма получилась сравнительно медленной. Данная работа заключается в оптимизации этой реализации и сравнение с аналогами.

1. Постановка задачи

В рамках данной курсовой работы ставились следующие задачи:

- исследовать алгоритм сжатия изображений без потерь из статьи "Lossless Image Compression Scheme with Binary Layers Scanning";
- оптимизировать(по скорости работы) его предложенную реализацию;
- провести сравнение с современными реализациями алгоритмов сжатия изображений без потерь.

Для сравнения по скорости работы и степени сжатия будут использованы различные наборы изображений:

1. набор медицинских изображений (УЗИ)
2. известный набор для тестирования реализаций алгоритмов сжатия изображений без потерь LPCB [8]

2. Современные методы сжатия изображений без потерь

Основные характеристики того или иного алгоритма сжатия изображений без потерь: скорость работы и степень сжатия.

Самые быстрые алгоритмы для сжатия изображений, как правило, очень просты и имеют низкую эффективность. Например, универсальные алгоритмы сжатия данных. Они могут использовать методы энтропийного кодирования (коды Хаффмана, арифметическое кодирование) или кодирование по словарю (семейство LZ*, RLE). Для сравнения возьмем реализации наиболее эффективных алгоритмов PPM и LZMA (соответственно PPMd [11] и 7z [6]).

Алгоритмы, специализирующиеся именно на сжатии изображений, имеют большую степень сжатия, но уступают в скорости работы. Для использования двумерности данных применяются различные обратимые интегральные преобразования над изображениями или предсказательные модели.

Алгоритмы, использующие обратимые преобразования, изначально разрабатывались для сжатия изображений с потерями малозаметных человеческому глазу данных. Одним из них является стандарт JPEG-2000, который также имеет режим работы без потерь. На фотореалистичных изображениях он имеет среднюю степень сжатия, но существенно проигрывает другим алгоритмам на искусственных изображениях. В сравнении будем использовать реализацию Jpeg-2000 из библиотеки GDCM [3], предназначенную для работы со специальным форматом для хранения медицинских изображений DICOM.

Что касается алгоритмов, использующих предсказательные модели (рисунок 1), то они имеют лучшие показатели по степени сжатия на искусственных изображениях, а степень сжатия фотореалистичных изображений зависит от эффективности предсказательной модели.

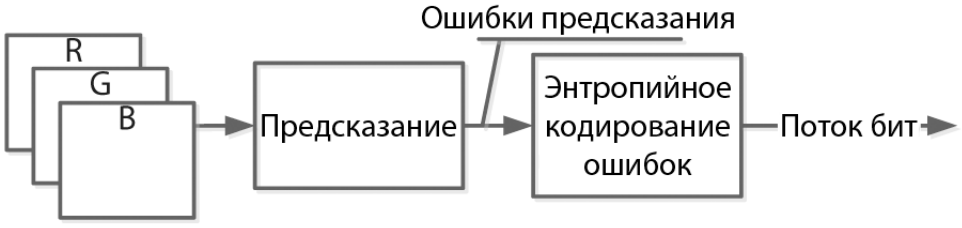

Рис. 1: Схема сжатия изображений, с использованием предсказательной модели

Смысл предсказательной модели состоит в том, чтобы перед кодированием занчений пикселей изображения использовать зависимость (корреляцию) цветовых компонент. Этого можно добиться, например изменением цветового пространства. После

алгоритм пытается предсказывать значение пикселей, основываясь на значениях соседей. Далее, ошибки предсказания кодируются энтропийными методами. С помощью эффективной предсказательной модели можно сильно снизить область допустимых значений ошибок, что приводит к значительному увеличению степени сжатия. Стандарт Jpeg-LS имеет простую предсказательную схему MED [5] и обладает высокой скоростью работы. Более современный алгоритм GraLIC [7] использует сложные модели и, следовательно, большую степень сжатия, но имеет более низкую скорость работы.

3. Описание нового алгоритма

Рассматриваемый алгоритм, описанный в статье [2], не трансформирует цветовое пространство RGB. Он относиться к алгоритмам, использующим предсказательную модель. Значит два основных этапа его работы: предсказательная модель и энтропийное кодирование ошибки предсказания.

3.1. Предсказатель

Различают два вида зависимости данных в изображении: внутри одной цветовой компоненты (intra) и между различными компонентами (inter).

Для избавления от inter корреляции используют, например, трансформацию в цветовое пространство YCbCr [9]. Но в данном алгоритме мы этого не делаем, чтобы локально адаптировать веса предсказателей, направленных на использование intra или inter зависимости. Итоговое предсказание - взвешенная сумма двух предсказателей. Проход по изображению идет построчно слева направо, сверху вниз для каждой из трех цветовых компонент. После исследований, проведенных авторами статьи данного алгоритма, было установлено, что зеленая цветовая компонента имеет самую большую корреляцию с другими компонентами. Таким образом, первым делом, обрабатывается зеленая компонента, с использованием только intra предсказателя. Остальные две цветовых компоненты также используют inter корреляцию.

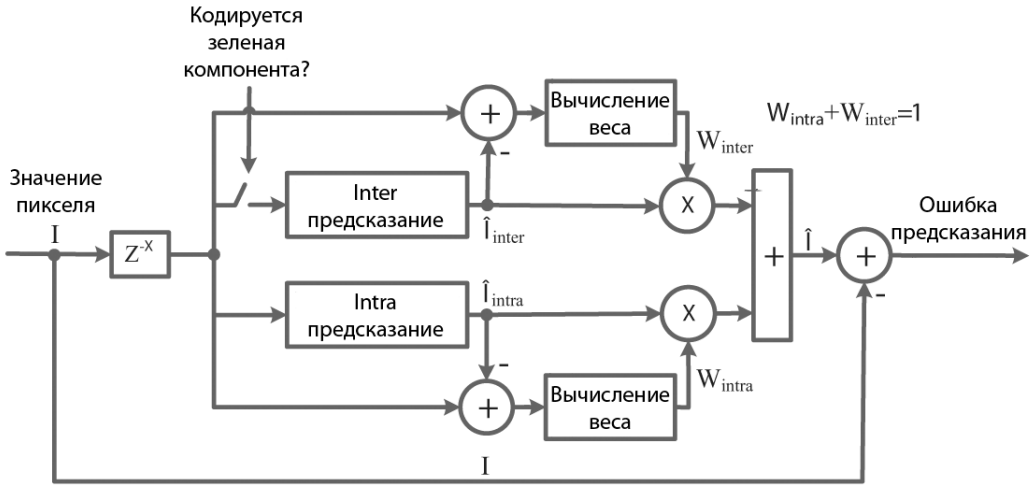

Рис. 2: Схема предсказателя

В качестве intra предсказателя берется MED (Median Edge Detector) [5], вычисля-

емый следующим образом:

$$MED = \begin{cases} min(A, C) & \text{если } B \geq max(A, C) \\ max(A, C) & \text{если } B \leq min(A, C) \\ A + C - B & \text{иначе} \end{cases} \quad (1)$$

где C - значение цветовой компоненты в верхнем, относительно текущего, пикселе, A - в левом, B - левом верхнем.

Inter предсказание вычисляется аналогично методу, применяемому в кодеке SICLIC [1]. Основывается на предположении о схожести градиентов двух цветовых компонент.

Ошибка предсказания - есть разность истинного и предсказанного значения. Также хранится история ошибок пройденных пикселей цветовой компоненты для каждого из двух предсказателей.

Далее вычисляется коэффициент (вес) каждого из предсказателей для текущего пикселя по формуле:

$$W = \sum_{i=1}^{7} err_i^2 * e^{-\lambda * err_i} \quad (2)$$

где сумма берется по истории ошибок предсказания тем или иным методом в соседних (относительно X) пикселях B, C, A, WW, NN, NNW и WWN (рисунок 3), а параметр λ - константа, равная 0.05 в текущей реализации.

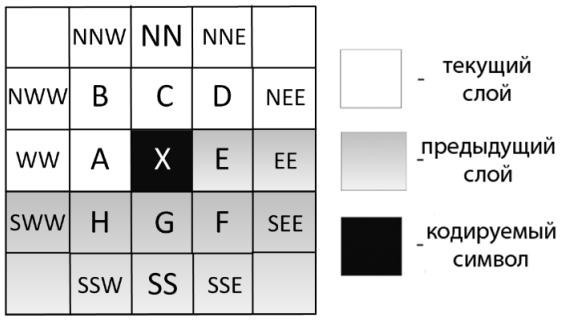

Рис. 3: Используемые пиксели

Таким образом, локально адаптируемся, меняя веса каждого из предсказателей, и отдаем предпочтение тому, который дает лучшие результаты в текущей области изображения.

В конце разделяем ошибку предсказания на абсолютное значение и знак. Они будут кодироваться энтропийными методами отдельно, причем если ошибка равна 0 равно нулю, то знак кодироваться не будет.

3.2. Энтропийное кодирование

Первым делом модули ошибок предсказания подвергаются инверсному унарному кодированию (например: 5 → 000001). Дальнейшее кодирование будет проходить по каждому унарному слою начиная с нижнего и выше (рисунок 4).

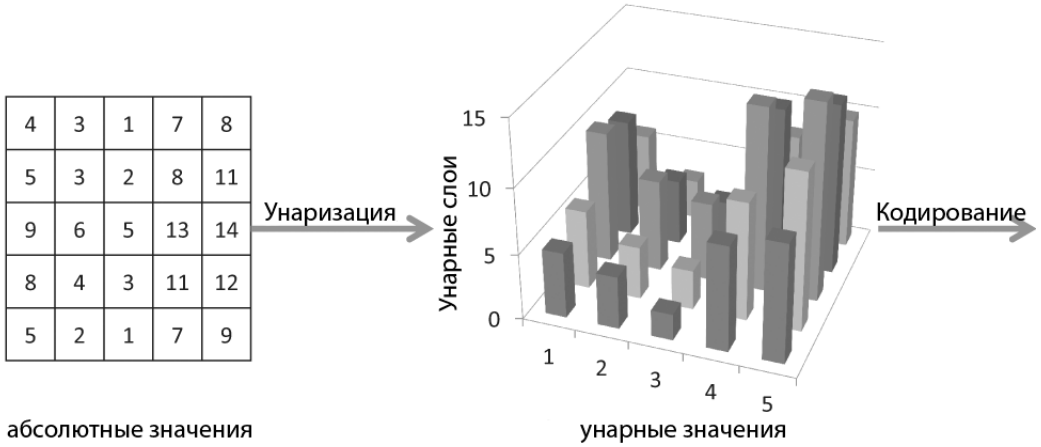

Рис. 4: Унаризация модулей ошибок

Значение пикселя в позиции (i, j) на унарном слое k (нумерация идет снизу вверх) можно вычислить по формуле:

$$l_{i,j}^{(k)} = \begin{cases} 0 & \text{если } |err|_{i,j} > k \\ 1 & \text{если } |err|_{i,j} = k \\ \text{-} & \text{если } |err|_{i,j} < k \end{cases} \tag{3}$$

При обходе одного унарного слоя, производится бинарное арифметическое кодирование [10], с предшествующем контекстным моделированием. Т.е. отдельно кодируются значения $l_{i,j}^{(k)}$ на слое k, принадлежащие разным моделям. Всего прдеусмотрено 9 контекстных моделей, определяемыми количеством соседей со значением 0. Из-за обхода слева направо, сверху вниз, левые верхние (A, B, C, D) соседи берутся с текущего слоя k, а нижние правые (E, F, G, H) со слоя $k-1$.

Для вероятностной оценки появления конкретного значения 0 или 1 для $l_{i,j}^{(k)}$ используется следующая формула:

$$P = \frac{n_x}{n_0 + n_1} \tag{4}$$

где n_x равняется количеству значений x (0 или 1) в уже закодированных значениях. Для каждой из 9 контекстных моделей заводятся свои счетчики для n_0 и n_1.

Также кодируется знак ошибки предсказания. Это происходит аналогично кодированию модуля за исключением того, что контекстная модель вычисляется только

по значениям$(-1, 0, 1)$ соседей A, B, C и D. Т.е.всео получается $3^4 = 81$ контекстных моделей.

3.3. Реализация

Предложенная реализация представляет собой консольное приложение на языке C++, симметричное по времени кодирования/декодирования. Также особенностью является возможность работы с изображениями с различной глубиной цвета, что хорошо подходит для медицинских изображений.

Для профилирования приложения использовалась утилита Callgrind из пакета инструментального ПО Valgrind.

Оптимизировался модуль предсказания, как правило, занимающий основное время работы. В ходе курсовой работы была реализована параллельная обработка цветовых компонент (изначально они выполнялись последовательно). Для этого они были избавлены от зависимостей (например, общая история ошибок). Данная оптимизация дала прирост скорости около 10 - 15% (после оптимизации начальной реализации моим коллегой Глазачевым Владимиром Александровичем).

Также предпринимались попытки векторизовать вычисления в предсказателе(т.е. обрабатывать одновременно несколько пикселей данного цветового слоя), но была обнаружена следующая проблема: нельзя начать обработку пикселя X, пока не будут обработаны пиксели B, C, A, WW, NN, NNW и WWN (рисунок 5).

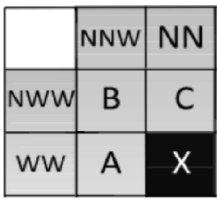

Рис. 5: Проблемы векторизации

Это связано с тем, что при вычислении пикселя X используются истории ошибок его перечисленных 7 соседей. Таким образом обработка сразу нескольких пикселей в данной реализации невозможна из-за зависимости текущего пикселя X от предыдущих в обходе пикселей A и WW.

Также в ходе данной курсовой работы был придуман и реализован новый обход пикселей (рисунок 6). Изменения затронули все основные части программы (например: запись / чтение изображений и предсказательная модель при кодировании / декодировании).

В новом обходе последовательно обрабатываются пиксели, лежащие на одной побочной диагонали. Т.е. 7 перечисленных соседей каждого из пикселей рассматриваемой диагонали лежат на предыдущей диагонали, а значит - уже обработаны. Таким

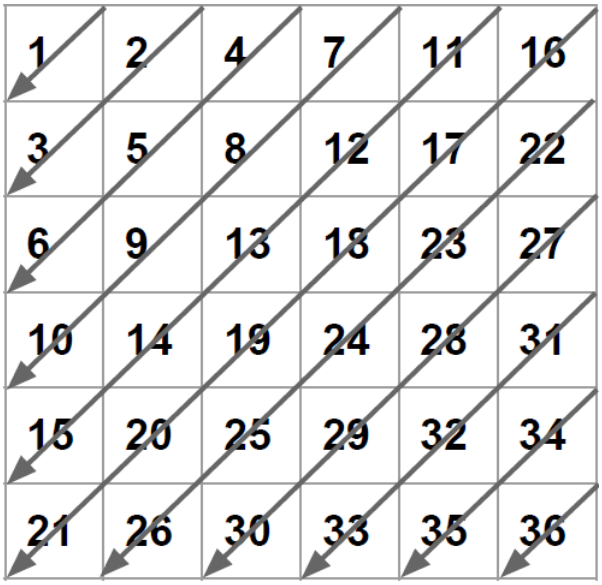

Рис. 6: Новый обход цветовых компонент

образом, реализованный обход позволит в дальнейшем провести векторизацию предсказательной модели и еще больше ускорить ее работу.

4. Сравнение

Тестирование проводилось на компьютере со следующими характеристиками:

- CPU - Intel Core 2 Duo T6670 (2.20ГГц)
- RAM - 4 Гб DDR3 1066 МГц

4.1. Medical image benchmark

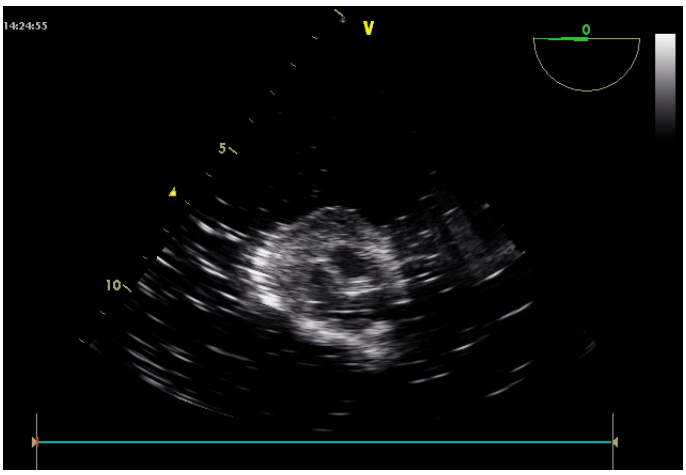

Рис. 7: Пример изображения УЗИ (с наложенной метаинформацией)

Первый набор для тестирования и сравнения реализаций - база медицинских изображений (УЗИ). Он содержит несколько сотен изображений, аналогичных рисунку 7. Оптимизации были применены после первых оптимизаций моего коллеги В.А. Глазачева.

Кодек	Сжатие (bits/pel)	Скорость (мб/сек)	Ст. отклонение скорости σ
Начальная реализация	1.8	0.974	0.030
Первая оптимизация	1.8	3.143	0.160
Текущая реализация	1.8	3.532	0.163
Jpeg-2000	4	3.544	0.270
Jpeg-LS	2	14.753	0.933
7-Zip (LZMA)	2.5	5.205	0.695
PPMd	2.15	9.719	0.804

Таблица 1: Сравнение кодеков на медицинских изображениях

В изображениях такого типа имеется много однородных областей. Поэтому алгоритмы с предсказательными схемами (Jpeg-LS, PPMd) имеют высокую степень сжатия.

4.2. Lossles Photo Compression Benchmark

Lossles Photo Compression Benchmark (LPCB) [8] является стандартным набором для тестирования и сравнения алгоритмов сжатия изображений без потерь. LPCB содержит 107 различных фотореалистичных изображений в высоком разрешении с общим весом 3.5GB. Так, например, в него входят астрономические и пейзажные изображения высокого качества. В таблице также приведена первая оптимизация, выполненная В.А. Глазачевым.

Кодек	Место в бенчмарке	Сжатие (bits/pel)	Скорость (мб/сек)	Ст. отклонение скорости σ
GraLIC	1	6.74	1.476	0.215
Начальная реализация	9	7.98	0.584	0.121
Первая оптимизация	9	7.98	1.878	0.467
Текущая реализация	9	7.98	2.069	0.535
Jpeg-2000	19	8.64	6.710	0.812
Jpeg-LS	27	8.97	13.581	1.017

Таблица 2: Сравнение кодеков на бенчмарке LPCB

Рассматриваемый алгоритм показывает высокую степень сжатия на фотореалистичных изображениях и работает быстрее более сложного алгоритма GraLIC.

Заключение

В рамках курсовой работы был изучен и оптимизирован (по времени работы) современный алгоритм сжатия изображений без потерь. А также был реализован новый обход пикселей, позволяющий в дальнейшей работе векторизовать предсказательную модель.

Также проведено сравнение реализации алгоритма с современными кодеками на различных наборах изображений. Не смотря на хорошую степень сжатия, алгоритм обладает все еще относительно низкой скоростью. Требуется дальнейшая работа по оптимизации скорости алгоритма (например, векторизация предсказательной модели).

Список литературы

[1] Barequet Raz, Feder Meir. SICLIC: A Simple Inter-Color LosslessIage Coder // IEEE Data Compression Conference. — 1999.

[2] Gilmutdinov Marat, Egorov Nickolay, Novikov Dmitriy. Lossless Image Compression Scheme with Binary Layers Scanning.

[3] Grassroots DICOM library. — 2013. — URL: `http://gdcm.sourceforge.net/wiki/index.php/Main_Page`.

[4] Information Technology JPEG 2000 image codingsystem, ISO/IEC 15444-1:2004, Dec. 2009.

[5] Information Technology Lossless and Near-lossless Compression of Continuous-Tone Still Images: Baseline, ISO/IEC, ISO/IEC 14495-1:1999.

[6] Pavlov Igor. LZMA SDK (Software Development Kit). — 2015. — URL: `http://www.7-zip.org/sdk.html` (дата обращения: 20.04.2015).

[7] Rhatushnyak Alexander. GraLIC - new lossless image compressor. — 2010. — URL: `http://encode.ru/threads/595-GraLIC-new-lossless-image-compressor` (online; accessed: 19.04.2015).

[8] Rhatushnyak Alex. Lossless Photo Compression Benchmark. — 2013. — URL: `http://imagecompression.info/gralic/LPCB.html` (online; accessed: 15.04.2015).

[9] Union International Telecommunication. BT.2020 : Parameter values for ultra-high definition television systems for production and international programme exchange. — 2012.

[10] Witten Ian H., M Radford. Arithmetic Coding for Data Compression // Communications of the ACM. — 1987.

[11] Шкарин Дмитрий. PPMd compression library. — 2003. — URL: `http://www.compression.ru/arctest/archive/comp_soft_others_arctest_18072003.html` (online; accessed: 19.04.2015).