САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра Системного Программирования

Глазачев Владимир Александрович

Оптимизация алгоритма сжатия изображений без потерь

Курсовая работа

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

Санкт-Петербург
2015

Оглавление

Введение **3**

1. Постановка задачи **4**

2. Современные подходы и решения **5**

3. Описание алгоритма **7**

3.1. Динамическая схема предсказания 7

3.2. Энтропийное кодирование 9

3.3. Реализация 10

4. Сравнение **11**

4.1. Medical image benchmark 11

4.2. Lossles Photo Compression Benchmark 12

Заключение **13**

Список литературы **14**

Введение

Алгоритмы сжатия изображений - бурно развивающаяся область машинной графики. В современном мире производится огромное количество цифровых изображений и они, в отличии от текстовых данных, требуют для хранения гораздо большие объемы памяти. Часто можно существенно снизить хранимый объем, отбросив незначительные для человеческого глаза детали. Однако, есть области, в которых необходимо иметь изображение в том виде, в котором оно было получено с оборудования. Чтобы снизить издержки на хранение такого типа изображений разрабатываются специальные алгоритмы сжатия изображений без потерь.

Изображения, в отличии, например, от текста, обладают избыточностью в двух измерениях. То есть, как правило, соседние точки в изображении близки по цвету. Кроме того, мы можем использовать подобие между цветовыми компонентами, что дает возможность создавать еще более эффективные алгоритмы.

Современные стандарты, такие как JPEG-LS [5] и JPEG-2000 [4] имеют режимы сжатия без потерь, однако они не учитывают некоторые свойства изображений. Целью работы является эффективная реализация алгоритма, описанного в статье "Lossless Image Compression Scheme with Binary Layers Scanning" [2], использующего динамическую предсказательную схему и кодирование по унарным слоям.

1. Постановка задачи

В рамках данной курсовой работы ставились следующие задачи:

- Оптимизировать (по скорости работы) предложенную реализацию алгоритма из статьи "Lossless Image Compression Scheme with Binary Layers Scanning" [2]
- Провести сравнение с другими решениями в области сжатия изображений без потерь

Для сравнения алгоритмов необходимо использовать различные наборы изображений. В работе используется набор медицинских изображений (УЗИ), а также стандартный набор для тестирования алгоритмов сжатия LPCB [8]. Основными критериями для сравнения являются степень сжатия и время, затраченное на работу алгоритма.

2. Современные подходы и решения

Методы сжатия изображений без потерь можно разделить на три класса:

1. Алгоритмы, использующие предсказательные модели, учитывающие двумерность изображения

2. Алгоритмы, использующие различные обратимые интегральные преобразования над изображениями

3. Универсальные алгоритмы сжатия, работающие с произвольными строками

Алгоритмы третьего типа можно разделить на два подкласса - алгоритмы, использующие методы энтропийного кодирования (арифметическое кодирование, коды Хаффмана) и использующие кодирование по словарю (RLE, Defflate, семейство LZ). Для многих современных реализаций характерна очень высокая скорость сжатия, однако размер выходного файла очень сильно зависит от типа данных. Алгоритмы этого типа показывают высокую степень сжатия на искусственных изображениях, но существенно проигрывают на фотореалистичных изображениях. В качестве примеров будем рассматривать наиболее прогрессивные алгоритмы этого типа: PPM и LZMA и их реализации PPMd [12] и 7z [6]. Также стоит отметить, что алгоритмы этого типа часто используются как один из этапов более сложных реализаций.

Ярким примером алгоритмов второго типа является Jpeg-2000. Стандартом Jpeg-2000 поддерживается возможность сжатия без потерь, однако это не дает значимых результатов. Такие методы изначально разрабатывались для сжатия с потерями, чтобы избавиться от малозаметных человеческому глазу данных. В результате Jpeg-2000 имеет среднюю степень сжатия на фотореалистичных изображениях и существенно проигрывает другим методам на искусственных изображениях. В сравнении будем использовать реализацию Jpeg-2000 из библиотеки GDCM [3] (библиотека для работы с изображениями в специальном формате для хранения медицинских изображений DICOM).

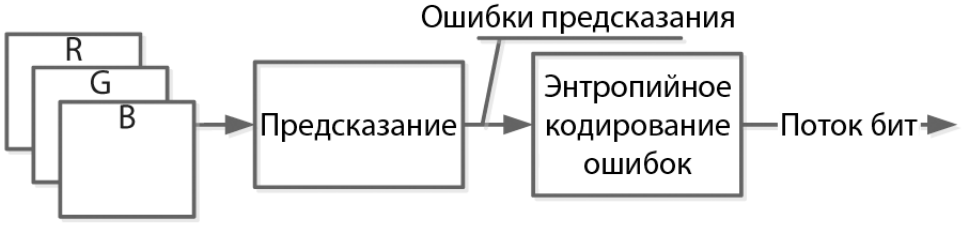

Рис. 1: Типичная схема сжатия

Наконец, алгоритмы первого типа используют предсказательные модели. Эти методы создавались специально для сжатия изображений без потерь. Вместо того, чтобы

кодировать значение в пикселе, сначала применяются различные методы снижения зависимости между цветовыми компонентами (декорреляция), например, изменение цветового пространства. После этого мы пытаемся предсказать значение в текущем пикселе по его соседям, вычисляем ошибку предсказания и кодируем её, используя энтропийные методы. Алгоритмы этого типа имеют лучшие показатели по степени сжатия на искусственных изображениях, а качество кодирования фотореалистичных изображений зависит от сложности предсказательной модели. Стандарт Jpeg-LS использует простую предсказательную схему MED [5] и обладает высокой скоростью работы. Более современный алгоритм GraLIC [7] использует сложные модели, но имеет низкую скорость сжатия.

3. Описание алгоритма

На вход алгоритму подается изображение в цветовом пространстве RGB. На выходе мы должны получить отформатированный специальным образом поток бит.

Описываемый алгоритм относится к первому типу, описанному в главе 2. Таким образом, его работу можно разделить на два независимых этапа - предсказание и энтропийное кодирование.

3.1. Динамическая схема предсказания

Для достижения высокой степени компрессии необходимо избавится от избыточности, появляющейся из-за зависимости значений пикселей как внутри одной компоненты, так и между разными компонентами. Будем обозначать словом inter зависимость между различными компонентами, а intra - внутри одной компоненты.

Часто для того, чтобы снизить корреляцию между компонентами (inter) используется переход в цветовое пространство YCbCr [9]. В алгоритме [2] мы остаемся в пространстве RGB, это позволяет использовать зависимость между компонентами для улучшения качества предсказания.

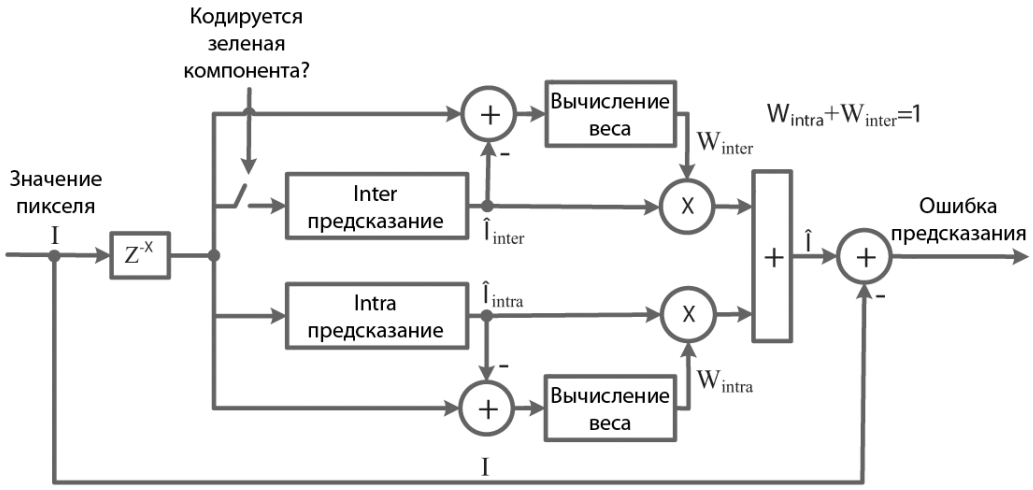

Рис. 2: Схема предсказателя

Для финального предсказания используется взвешенная сумма inter и intra предсказателей. В качестве intra предсказателя используется Median Edge Detector (MED) [5]. MED вычисляется по формуле:

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

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

Inter предсказание вычисляется аналогично методу, применяемому в кодеке SICLIC [1]. Мы смотрим на две цветовых компоненты, считаем градиенты на этих компонентах и предполагаем, что градиент на текущей компоненте на текущем пикселе будет близок к вычисленным.

Ошибка предсказания вычисляется как разность истинного и предсказанного значения.

Далее вычисляется коэффициент для каждого из предсказателей по формуле:

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

где сумма берется по ошибкам предсказания в соседних пикселях B, C, A, WW, NN, NNW и WWN (рисунок 3), а λ - константа, равная в текущей реализации 0.05.

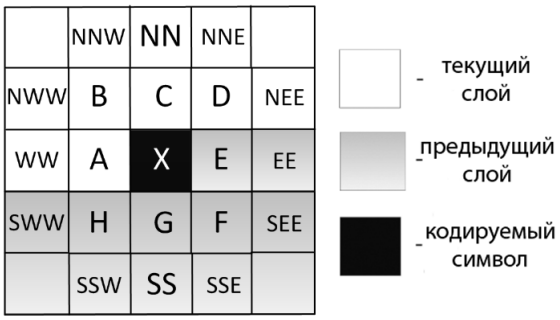

Рис. 3: Используемые пиксели

Таким образом, мы динамически меняем веса каждого из предсказателей и отдаем предпочтение тому методу, который дает лучшие результаты на текущем пикселе.

В конце разделяем абсолютное значение и знак ошибки предсказания. Они будут кодироваться отдельно. Если абсолютное значение ошибки равно нулю, знак не будет кодироваться.

Стоит заметить, что когда мы кодируем первую компоненту, мы не можем использовать значения с соседних необработанных компонент. Было экспериментально установлено, что зеленая компонента имеет наибольшую корреляцию с другими компонентами, так что кодирование происходит в следующем порядке:

1. Кодирование зеленой компоненты используя только intra предсказание
2. Кодирование красной компоненты, используя зеленую компоненту для inter предсказания
3. Кодирование синей компоненты, используя зеленую и красную компоненты

Таким образом, хотя в реализации и предусмотрено кодирование изображений в градациях одного цвета, алгоритм будет давать наилучшие результаты именно на RGB изображениях.

3.2. Энтропийное кодирование

На вход этому этапу подается массив из абсолютных значений ошибок $|err|$ предсказания. Далее мы переводим значение $|err|$ в унарный вид (пример: $5 \rightarrow 000001$). Кодирование будет происходит по каждому унарному слою, начиная с нижнего (рисунок 4).

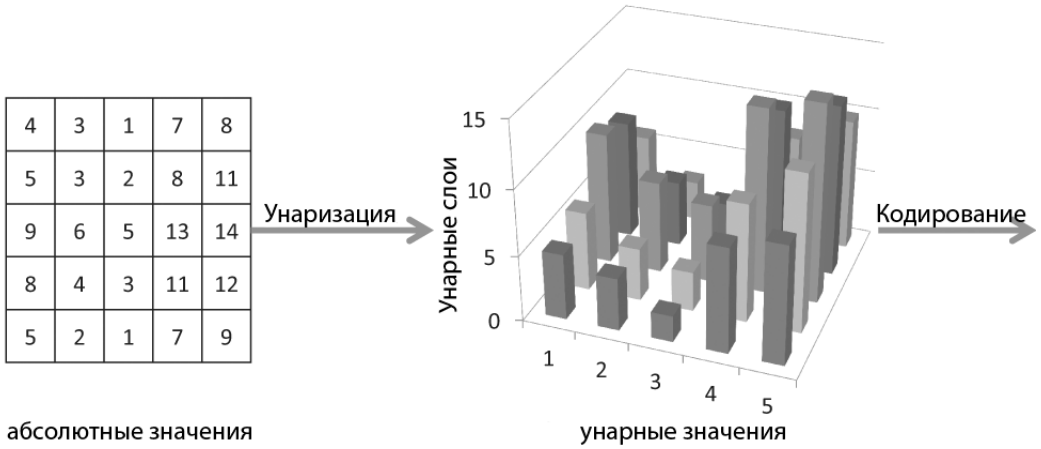

Рис. 4: Унаризация модулей ошибок

Значение пикселя в позиции (i, j) на унарном слое k вычисляется по следующему правилу:

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

Для каждого пикселя бинарного слоя k вычисляем количество соседей со значением 0. При этом левые верхние (A, B, C, D) соседи берутся с текущего слоя k, а нижние правые (E, F, G, H) со слоя $k-1$. В зависимости от количества нулевых соседей выбираем контекстную модель. В нашем случае получается 9 возможных контекстных моделей.

Для сжатия полученных значения $l_{i,j}^{(k)}$ применяется бинарное арифметическое кодирование [10]. Для вероятностной оценки появления конкретного значения $l_{i,j}^{(k)}$ используется следующая формула:

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

где n_x равняется количеству нулей или единиц (в зависимости от значения $l_{i,j}^{(k)}$) в уже закодированных значениях. Для каждой контекстной модели заводятся свои счетчики для n_0 и n_1.

После кодирования модулей ошибок мы переходим к знакам. Кодирование знаков происходит аналогично кодированию в унарных слоях за исключением того, что контекстная модель вычисляется только по соседям A, B, C и D. Для знака существует три возможных значения: $-1, 0, 1$, таким образом, получаем всего $3^4 = 81$ контекстную модель. Если знак равняется нулю, то его можно не кодировать.

3.3. Реализация

Реализация представляет собой консольное приложение на языке C++. Реализация является симметричной по времени кодирования/декодирования и может работать с изображениями с произвольной глубиной цвета (влезающих в int). На вход подаются изображения в формате bmp или pnm.

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

В ходе курсовой работы был полностью переписан модуль предсказания. Это позволило увеличить скорость работы программы более чем в три раза. Основной прирост скорости дала замена функции, вычисляющей вес конкретного предсказателя (2). Вычисление экспоненты занимало огромное количество времени, однако, мы заранее знаем, что ошибка предсказания err_i не будет по модулю превосходить глубину цвета. Поэтому, вначале мы можем сохранить все возможные значения $err_i^2 * e^{-\lambda * err_i}$ в массив и при дальнейших вычислениях использовать значения оттуда.

Была добавлена возможность использования GAP [11] предиктора вместо MED в intra предсказании. Это дало прирост степени сжатия на несколько процентов, но замедлило время работы алгоритма. Так что мы временно отказались от усложнения предсказательной модели.

Также был существенно переписан модуль кодирования. Унарные слои и контекст теперь вычисляются на месте (до этого использовался отдельный массив такой же размерности как изображение). Была существенно упрощена логика работы, в дальнейшем это может позволить векторизовать данный этап.

Стоит заметить, что наиболее важной частью алгоритма является именно предсказательная модель. А наиболее медленным местом кодирование ошибок. Замена алгоритма кодирования на более простой может дать очень высокий прирост к скорости работы, при небольших потерях в степени сжатия.

4. Сравнение

Для сравнения кодеков был реализован скрипт на языке Python. В скрипте необходимо указать пути до тестируемых исполняемых файлов и папки с кодируемыми изображениями. 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

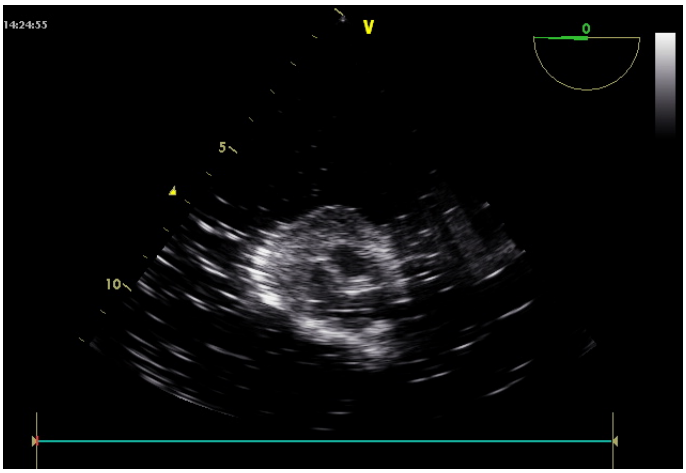

Рис. 5: Пример изображения УЗИ (с наложенной метаинформацией)

В качестве первого набора для тестирования и сравнения кодеков используется база медицинских изображений (УЗИ). Набор содержит несколько сотен изображений аналогичных рисунку 5.

Кодек	Сжатие (bits/pel)	Скорость (мб/сек)	Ст. отклонение скорости σ
Начальная реализация	1.8	0.974	0.030
Текущая реализация	1.8	3.143	0.160
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

Характерным для такого вида изображений является то, что в них много однотипных областей. Это позволяет алгоритмам, использующим предсказательные схемы (Jpeg-LS, PPMd), показывать высокие степени компрессии.

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 более 100 (3.5GB) различных фотореалистичных изображений в высоком разрешении. Так, например, в него входят сырые (raw) изображения с современных фотоаппаратов и изображений полученные со спутников и телескопов.

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
Текущая реализация	9	7.98	1.878	0.467
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

По таблице видно, что алгоритм показывает высокую степень сжатия на фотореалистичных изображениях и занимает 9 место в LPCB.

Заключение

В рамках курсовой работы был изучен и существенно оптимизирован (по времени работы) современный алгоритм сжатия изображений без потерь.

Было проведено сравнение алгоритма с современными кодеками на различных наборах изображений. Алгоритм показывает хорошую степень сжатия как на специализированных (медицинских), так и на фотореалистичных изображениях, однако все еще обладает относительно низкой скоростью компрессии/декомпрессии.

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] Wu Xiaolin, Memon N. CALIC-a context based adaptive losslessimage codec // IEEE Acoustics, Speech, and Signal Processing. — 1996. — Vol. 4.

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