УДК 004.074.2

ПОДХОДЫ К ОПТИМИЗАЦИИ СЖАТИЯ ИЗОБРАЖЕНИЙ БЕЗ ВИДИМОЙ ПОТЕРИ КАЧЕСТВА

Е.А. МЕНИЦКИЙ
(Представлено: канд. физ.-мат. наук., доц. Д.Ф. ПАСТУХОВ)

В статье представлены основные способы оптимизации изображения посредством оптимизации передаваемых параметров, поиска оптимальной логики передачи и хранения медиаданных, алгоритмы и инструменты для сжатия изображений.

Ключевые слова: *информационные технологии, сжатие файлов, оптимизация размера данных, хранение данных.*

Введение. С увеличением объема получаемой информации, остро встает вопрос в необходимости иметь все большие вычислительные мощности по их хранению. В свою очередь, это приводит к дополнительным финансовым потерям на приобретение данных ресурсов. Конечно, всегда есть возможность по удалению неактуальной или неприоритетной информации. Но данный способ не подходит для систем, которым в силу юридических, программах и других особенностей, необходимо хранения информации длительное время. Возьмем как пример для такого случая медицинскую информационную систему – как структуру с наиболее трудным взаимодействием с изображениями.

Большинство медицинских информационных систем юридически обязаны длительно хранить медицинские документы. При этом наиболее ресурсоемкая информация в данных системах – это изображения различных исследований (рентгенография, ультразвуковое исследование). В данном случае, очень важно качество получаемых изображений, для получения верного диагноза. Именно поэтому, данные системы не могут позволить себе обычное сжатие изображения с большой потерей качества. Отсюда возникает потребность по изучению подходов по оптимальному получению, хранению и сжатию изображений.

Основные методы оптимального сжатия изображения без потерь. Изначально необходимо решить вопрос о том, как будет происходить сжатие входного изображения: собственными средствами или сторонними сервисами. Если же все также говорить на примере медицинских информационных систем, то подойдет способ при разработке своего решения. Это даст большей гибкости при работе с программными инструментами, а также не допустит передачу конфиденциальных данных третьим лицам.

Практически любая логика по обработке изображений в приложении сводится к его сжатию. Сжатие данных – алгоритмическое преобразование данных, производимое с целью уменьшения занимаемого ими объема. Применяется для более рационального использования устройств хранения и передачи данных [1].

На данный момент можно выделить несколько основных направления по оптимизации сжатия данных:

- изменений программных параметров сжатии изображений;
- изменения в алгоритмической логике медиа приложения;
- изменения в энкодере.

Изменений программных параметров сжатия изображений. К данному методу относятся все возможности настройки приложений, устройств или языков программирования по изначальному получению наиболее подходящего изображения.

Примером может служить передаваемый параметр optimize=True при сохранении изображения посредством использования библиотеки Pillow для языка Python, которая реализует алгоритм Хаффмана – жадный алгоритм оптимального префиксного кодирования алфавита с минимальной избыточностью [2]. При наличии данного флага происходит дополнительный проход при сканировании каждого изображения. Каждый первый проход, вместо записи в файл, вычисляет статистику вхождений по каждому значению код-символ.

На рисунке 1 можно увидеть, как изменилось значения размера файла после применения флага optimize=True. Разница в 4657 Кб не является существенной для одного файла, но даст более ощутимый эффект при большом их количестве. При этом, разрешение самого файла остается исходным, т.е. не происходит потери качества.

Изменения в алгоритмической логике медиа приложения. Сюда необходимо отнести условия по работе только с несколькими форматами, определенными разрешениями или размерами. Зачастую эти ограничения устанавливаются программно и служат для приведения алгоритма программы к одному единому процессу. Например, большинство пользовательского контента – JPEG и PNG форматы. JPEG отлично подходит для фотографий, но обычно не справляется с высококонтрастным дизайнерским контентом (таким как логотипы). В отличие от него, PNG сжимает изображение абсолютно без потерь, отлично

подходит для графики, но слишком громоздок для фотографий, где маленькие искажения все равно не заметны. Исходя из этого и из потребностей системы, можно сделать упор на загрузку изображений только в нужном формате. Например, в тех случаях, когда пользователи загружают фотографии в формате PNG, но при этом допустима некоторая потеря, мы можем сэкономить много места, если распознаем такие файлы и сохраним их предварительно сконвертировав в JPEG или вовсе ограничим их к загрузке.

```
>>> from PIL import Image
>>> import os
>>> foo = Image.open("image.jpg")
>>> print(os.stat('image.jpg').st_size)
12778
>>> foo.save("image.jpg",optimize=True)
>>> print(os.stat('image.jpg').st_size)
8121
>>>
```

Рисунок 1. – Пример сжатия изображения через флаг optimize=True

Еще одним способом оптимизации работы с изображениями можно назвать применение более современных форматов вроде JPEG2k. JPEG2k – графический формат, который вместо дискретного косинусного преобразования, применяемого в формате JPEG, использует технологию вейвлет-преобразования, основывающуюся на представлении сигнала в виде суперпозиции базовых функций – волновых пакетов. В результате такой компрессии изображение получается более гладким и четким, а размер файла по сравнению с JPEG при одинаковом качестве оказывается меньшим. JPEG2k полностью свободен от главного недостатка своего предшественника: благодаря использованию вейвлетов, изображения, сохраненные в этом формате, при высоких степенях сжатия не содержат артефактов в виде «решетки» из блоков размером 8x8 пикселей [3].

Конечно же, основным способом уменьшения размера файла будет является изменение самого алгоритма сжатия. Рассмотрим этот способ на примере файлов JPEG.

Самый известный способ уменьшить размер файлов JPEG – настройка под названием quality. Многие приложения, способные сохранять в формате JPEG, определяют quality в виде числа от 0 до 100 и соответствуют различным таблицам квантования. Квантирование – это один из шагов в процессе кодирования JPEG, когда теряется информация. Мы можем динамически изменять настройки качества, оптимизируя их для каждого отдельного изображения, чтобы достичь идеального баланса между качеством и размером, добавляя больше шумов, в том количестве, которые не будут видны человеческому глазу. Есть два способа сделать это:

– снизу вверх: Эти алгоритмы генерируют настроенные таблицы квантования, обрабатывая изображение на уровне блоков 8×8 пикселей. Они одновременно рассчитывают, сколько теоретического качества было потеряно и как эти потерянные данные усиливают или сокращают видимость искажений для человеческого глаза;

– сверху вниз: Эти алгоритмы сравнивают целое изображение с его оригинальной версией и определяют, сколько информации было потеряно. Последовательно генерируя кандидатов с различными настройками качества, мы можем выбрать того, который соответствует минимальному уровню оценки.

Данные алгоритмы позволяют оценивать изменения качества для отдельных блоков изображения или же путем создания изображений-кандидатов на разных уровнях качества, для последующей оценки падения качества каждого фрагмента изображения путем вычисления его индекса структурного сходства (SSIM) до тех пор, пока это значение находится в пределах настраиваемого, но статичного порога. Это позволяет нам выборочно понизить средний размер файла (и среднее качество) только для изображений, которые выше воспринимаемого порога.

Добавления данных подходов к алгоритмам обработки изображений, дает показатель quality в диапазоне от 80 до 85 (включительно). Это позволяет нам снизить средний размер файла без существенной потери в качестве изображения.

Изменения в энкодере. Большинство современных энкодеров изображений в своей реализации используют таблицы квантования по умолчанию данные в спецификации к формату, для сжатия файлов и конвертации в другие форматы. При этом всегда можно реализовать свои собственные улучшенные таблицы. Примером такой реализации может служить свободно распространяемый энкодер Mozjpeg разработанный компанией Mozilla [4]. Одно из отличий mozjpeg в том, что он использует альтернативную таблицу квантования. При это Mozjpeg оптимизирует как последовательные, так и прогрессивные JPEG, в среднем,

на 5% лучше, чем энкодер libjpeg-turbo, который используется во многих библиотеках по работе с изображениями. MozJPEG также сохраняя полную совместимость со стандартом JPEG и со всеми браузерами, что позволяет использовать набор инструментов этого энкодера в своих программах решениях, без существенной смены поддерживаемых форматов.

Заключение. Как видно из рассмотренной статьи, имеется ряд способов для оптимизации сжатия изображения популярных форматов. При этом многие и них, практически не создают значимых потерь в качестве. Такой результат достигается в основном пропорциональному добавлению потерь данных. Казалось бы, что при минимальных потерях, эффект от оптимизации будет также минимальным. Но если рассматривать данные способы для большого количества изображений, а также рассмотреть комбинации данных методов, то эффект от оптимизации может достигать значимых результатов и существенно влиять на ресурсопотребление системы.

ЛИТЕРАТУРА

1. Сжатие данных [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Сжатие_данных. – Дата доступа: 25.09.2019.
2. Код Хаффмана [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Код_Хаффмана. – Дата доступа: 25.09.2019.
3. JPEG_2000 [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/JPEG_2000. – Дата доступа: 25.09.2019.
4. Mozilla Advances JPEG Encoding with mozjpeg 2.0 [Электронный ресурс]. Режим доступа: https://research.mozilla.org/2014/07/15/mozilla-advances-jpeg-encoding-with-mozjpeg-2-0/. – Дата доступа: 25.09.2019.