Алгоритмы сжатия изображений, форматы графических файлов

Легко подсчитать, что несжатое полноцветное изображение, размером $2000 * 1000$ пикселов будет иметь размер около 6 мегабайт. Если говорить об изображениях, получаемых с профессиональных камер или сканеров высокого разрешения, то их размер может быть ещё больше. Не смотря на быстрый рост ёмкости устройств хранения, по-прежнему весьма актуальными остаются различные алгоритмы сжатия изображений.

Все существующие алгоритмы можно разделить на два больших класса:

- Алгоритмы сжатия без потерь;
- Алгоритмы сжатия с потерями.

Когда мы говорим о сжатии без потерь, мы имеем в виду, что существует алгоритм, обратный алгоритму сжатия, позволяющий точно восстановить исходное изображение. Для алгоритмов сжатия с потерями обратного алгоритма не существует. Существует алгоритм, восстанавливающий изображение не обязательно точно совпадающее с исходным. Алгоритмы сжатия и восстановления подбираются так, чтобы добиться высокой степени сжатия и при этом сохранить визуальное качество изображения.

Алгоритмы сжатия без потерь

Алгоритм RLE

Все алгоритмы серии RLE основаны на очень простой идее: повторяющиеся группы элементов заменяются на пару (количество повторов, повторяющийся элемент). Рассмотрим этот алгоритм на примере последовательности бит[1]. В этой последовательности будут чередовать группы нулей и единиц. Причём в группах зачастую будет более одного элемента. Тогда последовательности $11111\ 000000\ 11111111\ 00$ будет соответствовать следующий набор чисел $5\ 6\ 8\ 2$. Эти числа обозначают количество повторений (отсчёт начинается с единиц), но эти числа тоже необходимо кодировать. Будем считать, что число повторений лежит в пределах от 0 до 7 (т.е. нам хватит 3 бит для кодирования числа повторов). Тогда рассмотренная выше последовательность кодируется следующей последовательностью чисел $5\ 6\ 7\ 0\ 1\ 2$. Легко подсчитать, что для кодирования исходной последовательности требуется 21 бит, а в сжатом по методу RLE виде эта последовательность занимает 18 бит.

Хоть этот алгоритм и очень прост, но эффективность его сравнительно низка. Более того, в некоторых случаях применение этого алгоритма приводит не к уменьшению, а к увеличению длины последовательности. Для примера рассмотрим следующую последовательность

[1] Этот алгоритм может применяться не только к последовательностям бит, его можно обобщить и на байты.

111 0000 11111111 00. Соответствующая ей -последовательность выглядит так: 3 4 7 0 1 2. Длина исходной последовательности – 17 бит, длина сжатой последовательности – 18 бит.

Этот алгоритм наиболее эффективен для чёрно-белых изображений. Также он часто используется, как один из промежуточных этапов сжатия более сложных алгоритмов.

Словарные алгоритмы

Идея, лежащая в основе словарных алгоритмов, заключается в том, что происходит кодирование цепочек элементов исходной последовательности. При этом кодировании используется специальный словарь, который получается на основе исходной последовательности.

Существует целое семейство словарных алгоритмов, но мы рассмотрим наиболее распространённый алгоритм LZW, названный в честь его разработчиков Лепеля, Зива и Уэлча.

Словарь в этом алгоритме представляет собой таблицу, которая заполняется цепочками кодирования по мере работы алгоритма. При декодировании сжатого кода словарь восстанавливается автоматически, поэтому нет необходимости передавать словарь вместе с сжатым кодом.

Словарь инициализируется всеми одноэлементными цепочками, т.е. первые строки словаря представляют собой алфавит, в котором мы производим кодирование. При сжатии происходит поиск наиболее длинной цепочки уже записанной в словарь. Каждый раз, когда встречается цепочка, ещё не записанная в словарь, она добавляется туда, при этом выводится сжатый код, соответствующий уже записанной в словаре цепочки. В теории на размер словаря не накладывается никаких ограничений, но на практике есть смысл этот размер ограничивать, так как со временем начинаются встречаться цепочки, которые больше в тексте не встречаются. Кроме того, при увеличении размеры таблицы вдвое мы должны выделять лишний бит для хранения сжатых кодов. Для того чтобы не допускать таких ситуаций, вводится специальный код, символизирующий инициализацию таблицы всеми одноэлементными цепочками.

Рассмотрим пример сжатия алгоритмом. Будем сжимать строку кукушкакукушонкукупилакапюшон. Предположим, что словарь будет вмещать 32 позиции, а значит, каждый его код будет занимать 5 бит. Изначально словарь заполнен следующим образом:

Символ	Код
К	00001
У	00010
Ш	00011
А	00100
О	00101
Н	00110
П	00111
И	01000
Л	01001
Ю	01010

Эта таблица есть, как и на стороне того, кто сжимает информацию, так и на стороне того, кто распаковывает. Сейчас мы рассмотрим процесс сжатия.

Цепочка из исходной строки	Строка, добавляемая в словарь и её код		Сжатый код
К	КУ	01011	00001
У	УК	01100	00010
КУ	КУШ	01101	01011
Ш	ШК	01110	00011
К	КА	01111	00001
А	АК	10000	00100
КУ	КУК	10001	01011
КУШ	КУШО	10010	01101
О	ОН	10011	00101
Н	НК	10100	00110
КУК	КУКУ	10101	10001
У	УП	10110	00010
П	ПИ	10111	00111
И	ИЛ	11000	01000
Л	ЛА	11001	01001
АК	АКА	11010	10000
А	АП	11011	00100
П	ПЮ	11100	00111
Ю	ЮШ	11101	01010
Ш	ШО	11110	00011
ОН			10011

В таблице представлен процесс заполнения словаря[2]. Легко подсчитать, что полученный сжатый код занимает 105 бит, а исходный текст (при условии, что на кодирование одного символа мы тратим 4 бита) занимает 116 бит.

По сути, процесс декодирования сводится к прямой расшифровке кодов, при этом важно, чтобы таблица была инициализирована также, как и при кодировании. Теперь рассмотрим алгоритм декодирования.

Данные	Символы на выходе	Строка, добавляемая в словарь и её код	
00001	К	К?	01011
00010	У	У?	01100
01011	КУ	КУ?	01101
…			

Строку, добавленную в словарь на -ом шаге мы можем полностью определить только на $i + 1$. Очевидно, что -ая строка должна заканчиваться на первый символ $i + 1$ строки. Т.о. мы только что разобрались, как можно восстанавливать словарь. Некоторый интерес представляет ситуация, когда кодируется последовательность вида $cScSc$, где c — это один символ, а S — строка, причём слово cS уже есть в словаре. На первый взгляд может показаться, что декодер не сможет разрешить такую ситуацию, но на самом деле все строки такого типа всегда должны заканчиваться на тот же символ, на который они начинаются.

[2] Если в процессе работы размер словаря превысит 32 символа, необходимо заново инициализировать таблицу словаря, при этом важно оставить ту часть словаря, которая отвечает за алфавит, без изменений.

Алгоритмы статистического кодирования

Алгоритмы этой серии ставят наиболее частым элементам последовательностей наиболее короткий сжатый код. Т.е. последовательности одинаковой длины кодируются сжатыми кодами различной длины. Причём, чем чаще встречается последовательность, тем короче, соответствующий ей сжатый код.

Алгоритм Хаффмана

Алгоритм Хаффмана позволяет строить префиксные коды[3]. Можно рассматривать префиксные коды как пути на двоичном дереве: прохождение от узла к его левому сыну соответствует 0 в коде, а к правому сыну – 1. Если мы пометим листья дерева кодируемыми символами, то получим представление префиксного кода в виде двоичного дерева.

Опишем алгоритм построения дерева Хаффмана и получения кодов Хаффмана.

1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который равен частоте появления символа
2. Выбираются два свободных узла дерева с наименьшими весами
3. Создается их родитель с весом, равным их суммарному весу
4. Родитель добавляется в список свободных узлов, а двое его детей удаляются из этого списка
5. Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой — бит 0
6. Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.

С помощью этого алгоритма мы можем получить коды Хаффмана для заданного алфавита с учётом частоты появления символов.

Арифметическое кодирование

Алгоритмы арифметического кодирования кодируют цепочки элементов в дробь. При этом учитывается распределение частот элементов. На данный момент алгоритмы арифметического кодирования защищены патентами, поэтому мы рассмотрим только основную идею.

Пусть наш алфавит состоит из N символов $a_1, \dots, a_N$, а частоты их появления $p_1, \dots, p_N$ соответственно. Разобьем полуинтервал $[0; 1)$ на N непересекающихся полуинтервалов. Каждый полуинтервал соответствует элементам a_i, при этом длина полуинтервала пропорциональна частоте p_i.

Кодирующая дробь строится следующим образом: строится система вложенных интервалов так, чтобы каждый последующий полуинтервал занимал в предыдущем место, соответствующее положению элемента в исходном разбиении. После того, как все интервалы вложены друг в друга можно взять любое число из получившегося полуинтервала. Запись этого числа в двоичном коде и будет представлять собой сжатый код.

---

[3] Префиксным называется код, который обладает следующим свойством: менее короткие коды не соответствуют начальной (префиксной) части более длинных кодов.

Декодирование – расшифровка дроби по известному распределению вероятностей. Очевидно, что для декодирования необходимо хранить таблицу частот.

Арифметическое кодирование чрезвычайно эффективно. Коды, получаемые с его помощью, приближаются к теоретическому пределу. Это позволяет утверждать, что по мере истечения сроков патентов, арифметическое кодирование будет становиться всё более и более популярным.

Алгоритмы сжатия с потерями

Не смотря на множество весьма эффективных алгоритмов сжатия без потерь, становится очевидно, что эти алгоритмы не обеспечивают (и не могут обеспечить) достаточной степени сжатия.

Сжатие с потерями (применительно к изображениям) основывается на особенностях человеческого зрения. Мы рассмотрим основные идеи, лежащие в основе алгоритма сжатия изображений $JPEG$.

Алгоритм сжатия JPEG

$JPEG$ на данный момент один из самых распространенных способов сжатия изображений с потерями. Опишем основные шаги, лежащие в основе этого алгоритма. Будем считать, что на вход алгоритма сжатия поступает изображение с глубиной цвета 24 бита на пиксел (изображение представлено в цветовой модели RGB).

Перевод в цветовое пространство $YCbCr$

В цветовой модели $YCbCr$ мы представляем изображение в виде яркостной компоненты (Y) и двух цветоразностных компонент (Cb, Cr). Человеческий глаз более восприимчив к яркости, а не к цвету, поэтому алгоритм $JPEG$ вносит по возможности минимальные изменения в яркостную компоненту (Y), а в цветоразностные компоненты могут вноситься значительные изменения. Перевод осуществляется по следующей формуле:

$$\begin{cases} Kg = 1 - Kr - Kb \\ Y = minY + (maxY - minY) * (Kr * R + Kg * G + Kb * b) \\ Cb = \dfrac{minC + maxC}{2} + \dfrac{maxC - minC}{2} * \dfrac{1}{1 - Kb} * (-Kr * R - Kg * G + (1 - Kb) * B) \\ Cr = \dfrac{minC + maxC}{2} + \dfrac{maxC - minC}{2} * \dfrac{1}{1 - Kr} * ((1 - Kr) * R - Kg * G - Kb * B) \end{cases}$$

Коэффициенты Kr и Kb описаны в стандарте $JPEG$.

Субдискретизация компонент цветности

После перевода в цветовое пространство $YCbCr$ выполняется дискретизация. Возможен один из трёх способов дискретизации:

- $4{:}\,4{:}\,4$ – отсутствует субдискретизация;
- $4{:}\,2{:}\,2$ – компоненты цветности меняются через одну по горизонтали;
- $4{:}\,2{:}\,0$ – компоненты цветности меняются через одну строку по горизонтали, при этом по вертикали они меняются через строку.

При использовании второго или третьего способа мы избавляется от $\frac{1}{3}$ или $\frac{1}{2}$ информации соответственно. Очевидно, что чем больше информации мы теряем, тем сильнее будут искажения в итоговом изображении.

Дискретное косинусное преобразование

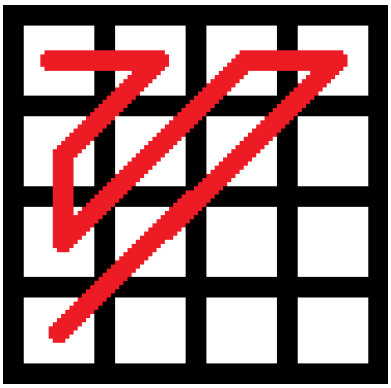
Изображение разбивается на компоненты $8 * 8$ пикселов, к каждой компоненте применятся ДКП. Это приводит к уплотнению энергии в коде. Преобразования применяются к компонентам независимо.

Квантование

Человек практически не способен замечать изменения в высокочастотных составляющих, поэтому коэффициенты, отвечающие за высокие частоты можно хранить с меньшей точностью. Для этого используется покомпонентное умножение (и округление) матриц, полученных в результате ДКП, на матрицу квантования[4]. На данном этапе тоже можно регулировать степень сжатия (чем ближе к нулю компоненты матрицы квантования, тем меньше будет диапазон итоговой матрицы).

Зигзаг-обход матриц

Зигзаг-обход матрицы – это специальное направление обхода, представленное на рисунке:



При этом для большинства реальных изображений в начале будут идти ненулевые коэффициенты, а ближе к концу будут идти нули.

RLE- кодировние

Используется особый вид -кодирования: выводятся пары чисел, причём первое число в паре кодирует количество нулей, а второе – значение после последовательности нулей. Т.е. код для последовательности $0\ 0\ 15\ 42\ 0\ 0\ 0\ 44$ будет следующим $(2; 15)(0; 42)(3; 44)$.

Кодирование методом Хаффмана

Используется описанный выше алгоритм Хаффмана. При кодировании используется заранее определённая таблица.

Алгоритм декодирования заключается в обращении выполненных преобразований.

К достоинствам алгоритма можно отнести высокую степень сжатие (5 и более раз), относительно невысокая сложность (с учётом специальных процессорных инструкций), патентная чистота. Недостаток – артефакты, заметные для человеческого глаза.

---

[4] Для яркостных и цветоразностных компонент матрицы отличаются.

Алгоритм сжатия JPEG2000

Алгоритм сжатия $JPEG2000$ дальнейшее развитие традиционного алгоритма $JPEG$. $JPEG2000$ вместо дискретного косинусного преобразования, применяемого в алгоритме $JPEG$, использует технологию вейвлет-преобразования, основывающуюся на представлении сигнала в виде суперпозиции базовых функций — волновых пакетов.

Вейвлет-преобразования превосходят по параметрам локализации традиционные косинусные преобразования. Это позволяет использовать более сильное квантование, что, в свою очередь, позволяет более эффективно использовать алгоритмы сжатия без потерь.

При использовании алгоритма $JPEG2000$ по сравнению с алгоритмом $JPEG$ сжатое изображение получается более чётким, а размер файла оказывается меньшим. Основным недостатком этого алгоритма является его малая распространённость.

Фрактальное сжатие

Фрактальное сжатие – это относительно новая область. Фрактал – сложная геометрическая фигура, обладающая свойством самоподобия. Алгоритмы фрактального сжатия сейчас активно развиваются, но идеи, лежащие в их основе можно описать следующей последовательностью действий.

Процесс сжатия:

1. Разделение изображения на неперекрывающиеся области (домены). Набор доменов должен покрывать всё изображение полностью.

2. Выбор ранговых областей. Ранговые области могут перекрываться и не покрывать целиком всё изображение.

3. Фрактальное преобразование: для каждого домена подбирается такая ранговая область, которая после аффинного преобразования наиболее точно аппроксимирует домен.

4. Сжатие и сохранение параметров аффинного преобразования. В файл записывается информация о расположении доменов и ранговых областей, а также сжатые коэффициенты аффинных преобразований.

Этапы восстановления изображения:

1. Создание двух изображений одинакового размера A и B. Размер и содержание областей не имеют значения.

2. Изображение B делится на домены так же, как и на первой стадии процесса сжатия. Для каждого домена области B проводится соответствующее аффинное преобразование ранговых областей изображения A, описанное коэффициентами из сжатого файла. Результат помещается в область B. После преобразования получается совершенно новое изображение.

3. Преобразование данных из области B в область A. Этот шаг повторяет шаг 3, только изображения A и B поменялись местами.

4. Шаги 3 и 4 повторяются до тех пор, пока изображения A и B не станут неразличимыми.

Точность полученного изображения зависит от точности аффинного преобразования.

Сложность алгоритмов фрактального сжатия в том, что используется целочисленная арифметика и специальные довольно сложные методы, уменьшающие ошибки округления.

Отличительной особенностью фрактального сжатия является его ярко выраженная ассиметрия. Алгоритмы сжатия и восстановления существенно различаются (сжатие требует гораздо большего количества вычислений).

Форматы графических файлов

Для хранения графических файлов используются различные форматы. Но каждый формат должен содержать информацию, достаточную для корректного отображения исходного изображения. Ниже рассмотрено несколько распространённых форматов.

BMP

BMP (BitMaP) – это формат хранения растровых изображений. Основное распространение он получил в Microsoft Windows.

В файлах BMP информация о цвете каждого пиксела кодируются $1,4,8,16$ или 24 битами на пиксел. Количество бит на пиксел определяет максимальное количество цветов в изображении. При использовании 1 бита на пиксел мы можем кодировать всего 2 цвета, а при использовании 24 бит на пиксел, мы можем закодировать более 16 млн. цветов.

Файл состоит из четырёх логических разделов:

1. Заголовок файла;
2. Информационный заголовок;
3. Таблица цветов;
4. Изображение.

Заголовок хранит необходимую информацию о файле (сигнатуру, размер файла, смещение от начала файла до описания изображения и т.д.).

Информационный заголовок содержит информацию об изображении: его размер.

Талица цветов используется для задания точных значений основных цветов. Также в некоторых случаях может использоваться палитра.

Изображение хранится в виде потока байт, начиная с нижней строки слева направо.

TIFF

TIFF (Tagged Image File Format) — формат хранения растровых изображений. В настоящее время TIFF стал популярным форматом для хранения изображений с большой глубиной цвета. Он широко используется в полиграфии.

Структура формата гибкая и позволяет сохранять изображения в режиме цветов с палитрой, а также в различных цветовых пространствах:

- Бинарном;
- Полутоновом;
- С палитрой;
- RGB;
- CMYK;
- YCbCr;
- CIE;

Поддерживаются режимы с использованием $8,16,32$ и 64 бит на канал при целочисленном кодировании, а также 32 и 64 бит на канал при кодировании числами с плавающей запятой.

TIFF поддерживает хранение изображений с сжатием и без сжатия. Формат TIFF позволяет использовать следующие алгоритмы сжатия:

- RLE
- LZW
- LZ77 (алгоритм с использованием словаря)
- ZIP
- JBIG
- JPEG
- CCITT Group 3, CCITT Group 4[5]

GIF

GIF (Graphics Interchange Format) — формат хранения растровых изображений. В GIF можно хранить сжатые изображения без потери качества в формате не более 256 цветов. Первая версия формата GIF была разработана в 1987 году. В 1989 году формат был модифицирован, были добавлены поддержка прозрачности и анимации.

Изображение в формате GIF хранится построчно, поддерживается только формат с индексированной палитрой цветов. Стандарт разрабатывался для поддержки 256-цветной палитры. Один из цветов в палитре может быть объявлен прозрачным. В этом случае в программах, которые поддерживают прозрачность сквозь пиксели, окрашенные прозрачным цветом, будет виден фон. Полупрозрачность пикселей (альфа-канал) не поддерживается.

Формат GIF поддерживает анимационные изображения. Они представляют собой последовательность из нескольких статичных кадров, а также информацию о том, сколько времени каждый кадр должен быть показан на экране. Анимацию можно сделать цикличной, тогда вслед за последним кадром начнётся воспроизведение первого кадра и т.д.

[5] Алгоритмы CCITT Group 3 и 4 предназначены для кодирования бинарных растровых изображений.

Недокументированной, но поддерживаемой возможностью является сохранение большего количества цветов с помощью анимированного GIF с нулевой задержкой между кадрами. При этом преодолевается ограничение в 256 цветов так как каждый кадр содержит свою палитру.

Формат GIF допускает чересстрочное хранение данных. При этом строки разбиваются на группы, и меняется порядок хранения строк в файле. При загрузке изображение проявляется постепенно, в несколько проходов. Благодаря этому, имея только часть файла, можно увидеть изображение целиком, но с меньшим разрешением.

PNG

PNG (portable network graphics) — формат хранения растровых изображений, использующий сжатие без потерь по алгоритму Deflate[6]. PNG был создан как свободный формат для замены GIF[7], поэтому в Интернете появился рекурсивный акроним «PNG's Not GIF».

Формат PNG спроектирован для замены устаревшего и более простого формата GIF, а также, в некоторой степени, для замены значительно более сложного формата TIFF. Формат PNG позиционируется прежде всего для использования в Интернете и редактирования графики.

Формат PNG поддерживает три основных типа растровых изображений:

1. Полутоновое изображение (с глубиной цвета 16 бит);
2. Цветное индексированное изображение (палитра 8 бит для цвета глубиной 24 бит);
3. Полноцветное изображение (с глубиной цвета 48 бит).

Он имеет следующие основные преимущества перед GIF:

- Большое количество цветов;
- Поддержка альфа-канала;
- Возможность гамма-коррекции;
- Двумерная чересстрочная развёртка;
- Возможность расширения формата пользовательскими блоками.

В качестве недостатка можно указать отсутствие поддержки анимации, правда эту проблему решает формат APNG.

JPEG

JPEG (Joint Photographic Experts Group) — формат хранения растровых изображений. Алгоритм JPEG в наибольшей степени пригоден для сжатия фотографий и картин, содержащих реалистичные сцены с плавными переходами яркости и цвета. Наибольшее распространение JPEG получил в цифровой фотографии и для хранения и передачи изображений с использованием сети Интернет.

---

[6] Алгоритм Deflate заключается в комбинированном применении алгоритма LZ77 и алгоритма Хаффмана

[7] До 2006 года алгоритм GIF был защищён патентами

С другой стороны, JPEG малопригоден для сжатия чертежей, текстовой и знаковой графики, где резкий контраст между соседними пикселами приводит к появлению заметных артефактов. Такие изображения целесообразно сохранять в форматах без потерь, таких как TIFF, GIF, PNG.

JPEG (как и другие методы сжатия с потерями) не подходит для сжатия изображений при многоступенчатой обработке, так как искажения в изображения будут вноситься каждый раз при сохранении промежуточных результатов обработки. Также JPEG не должен использоваться и в тех случаях, когда недопустимы даже минимальные потери, например, при сжатии карт. В таких случаях может быть рекомендован предусмотренный стандартом JPEG режим сжатия без потерь.

PDF

PDF (Portable Document Format) — кроссплатформенный формат электронных документов. В первую очередь формат предназначен для представления в электронном виде полиграфической продукции.

Формат PDF позволяет внедрять необходимые шрифты, векторные и растровые изображения, формы ввода и мультимедиа-вставки. PDF может содержать текстовый слой, что позволяет осуществлять полнотекстовый поиск по документу. Формат также включает механизм электронных подписей для защиты и проверки подлинности документов. В этом формате распространяется большое количество сопутствующей документации.

DJVU

DjVu (déjà vu) — технология сжатия изображений с потерями, разработанная специально для хранения сканированных документов — книг, журналов, рукописей и прочее, где обилие формул, схем, рисунков и рукописных символов делает чрезвычайно трудоёмким их полноценное распознавание. Также является эффективным решением, если необходимо передать все нюансы оформления (например, исторических документов, где важное значение имеет не только содержание, но и цвет и фактура бумаги; дефекты пергамента: трещинки, следы от складывания; исправления, кляксы, отпечатки пальцев; следы, оставленные другими предметами и т.д.).

Формат оптимизирован для передачи по сети таким образом, что страницу можно просматривать ещё до завершения загрузки файла. DjVu может содержать текстовый слой, что позволяет осуществлять полнотекстовый поиск по документу. Кроме того, DjVu может содержать встроенное интерактивное оглавление и активные области — ссылки, что позволяет реализовать удобную навигацию в DjVu-книгах.

Для сжатия цветных изображений в DjVu применяется специальная технология, разделяющая исходное изображение на три слоя:

1. Передний план;
2. Задний план;
3. Чёрно-белую маску.

Маска сохраняется с разрешением исходного файла; именно она содержит изображение текста и прочие чёткие детали. Разрешение заднего плана, в котором остаются иллюстрации и текстура страницы, по умолчанию понижается для экономии места. Передний план содержит цветовую информацию о маске, его разрешение обычно понижается ещё сильнее.

Для сжатия большинства книг можно обойтись только двумя цветами. В этом случае используется всего один слой, что позволяет достичь рекордной степени сжатия. В типичной книге с чёрно-белыми иллюстрациями, отсканированной с разрешением $600\,dpi$, средний размер страницы составляет около 15 Кб, то есть приблизительно в 100 раз меньше, чем исходный файл. В присутствии сложного заднего плана выигрыш объёма составляет обычно $4-10$ раз. Однако при этом не стоит забывать, что в DjVu используется сжатие данных с потерями, поэтому для особо важных документов, возможно, будет разумнее использовать форматы сжатия без потерь: PNG, TIFF и др.

WMF

WMF (Windows MetaFile) — универсальный формат векторных графических файлов для Windows приложений. Формат разработан Microsoft и является неотъемлемой частью Windows, так как сохраняет последовательность аппаратно-независимых функций GDI (Graphical Device Interface), непосредственно выводящих изображение в заданный контекст графического устройства (на экран, на принтер и т.п.). Очень часто WMF неявно используется для сохранения образа окна вывода программы и его последующего восстановления, а также при переносе информации через буфер обмена. Из MS Windows запись и чтение в файл этого формата осуществляются чрезвычайно просто и быстро, в других операционных системах поддержка этого формата бесполезна. Файл этого формата может быть открыт с помощью кроссплатформенных программ GIMP (с предварительной растеризацией) и Inkscape.