РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ
№ 5 1993

УДК 681.3.053

В. О. Криворучко
(Новосибирск)

СЖАТИЕ ЦВЕТНЫХ СИНТЕЗИРОВАННЫХ ИЗОБРАЖЕНИЙ*

Введение. Машинная графика находит все более широкое применение на телевидении, в информационных системах, при моделировании. При этом часто необходимо иметь динамичное изображение, когда частота обновления кадров составляет не менее 50 Гц. Построение изображения по математическому описанию в реальном времени является непосильной задачей даже для самых мощных современных компьютеров. Поэтому динамичное изображение получают путем его синтеза в нереальном времени с покадровой записью на жесткий диск, а затем покадровой перезаписью на видеомагнитофон с соответствующими возможностями, что часто неприемлемо по экономическим соображениям.

Для вывода последовательности кадров (фильма) с диска с частотой телевизионной развертки необходимо преодолеть несколько проблем.

Во-первых, существует серьезная проблема с хранением фильма. Размер фильма на диске можно найти как

$$M \times N \times P \times Fr \times T \text{ байт},$$

где $M \times N$ — размер кадра в пикселах, P — разрядность пиксела в байтах, Fr — частота кадров, T — время фрагмента фильма в секундах.

Вторая проблема связана с обеспечением необходимой скорости передачи данных. Требуемая скорость передачи данных находится как

$$M \times N \times P \times Fr \text{ байт/с}.$$

При телевизионном качестве изображений перечислим минимальные требования к этим параметрам: $M \times N = 384 \times 288$, $P = 3$, $Fr = 50$ кадр/с, $T = 10$ с. Тогда размер фильма на диске составит $384 \times 288 \times 3 \times 50 \times 10 = 158$ Мбайт.

Такие размеры существенно затрудняют хранение фильма и не позволяют использовать стандартную вычислительную технику типа персональных компьютеров. При этом для воспроизведения требуется скорость передачи информации 15,8 Мбайт/с, что также очень затруднительно реализовать на стандартном оборудовании, таком как IBM PC/AT, даже с использованием быстрых жестких дисков.

* Работа выполнена при финансовой поддержке Международного благотворительного фонда «Культурная инициатива».

Однако каждый кадр как информационное сообщение содержит некоторое количество избыточной информации. При ее устранении требования к аппаратуре значительно снижаются. Необходимо лишь найти алгоритм, позволяющий сжимать данные, а затем разжимать с необходимой для телевидения скоростью. В настоящее время получил широкое распространение метод сжатия изображений в соответствии со стандартом JPEG.

Стандарт JPEG. Задачу сжатия изображения можно разделить на две относительно независимые части: преобразование исходного изображения с учетом свойств сигнала и метода сжатия и непосредственно сжатие.

В стандарте JPEG [1] используются дискретное косинусное преобразование (ДКП) матрицы пикселов 8×8 (в YUV-представлении) и кодирование трансформант с использованием метода Хаффмана. Аппаратная реализация JPEG требует применения СБИС-технологии; ряд фирм выпускают соответствующие микросхемы [2].

Слабые стороны метода известны: сложность реализации, фрагментация,

С другой стороны, для синтезированных изображений характерно отсутствие шума и небольшой набор используемых цветов. Поэтому можно попытаться избежать применения методов типа JPEG.

Методы сжатия информации без потерь. Хорошо известен метод Хаффмана [3, 4], который основан на сборе статистики источника сообщений (в нашем случае кадра) и перекодировке исходных символов кодом переменной длины, т. е. часто встречающимся символам присваивается более короткий код, а более редким — более длинный. Существует динамический метод кодировки, в котором используется построение таблицы вероятностей «на ходу», без предварительного просмотра всего сообщения [5]. Поскольку данный метод имеет очень сложный алгоритм декодирования и отличается малой скоростью работы, в данной публикации он не рассматривается.

Известен также метод, основанный на идее нахождения во входном потоке данных повторяющихся последовательностей и замены этих последовательностей ссылками на предыдущее вхождение данной последовательности символов. Этот метод носит название LZW-кодирование (сокращение от Lempel—Ziv—Welch). Существует две разновидности LZW-кодирования. Первая из них использует хеш-таблицу при кодировании и декодировании данных [4]. Данный метод применяется в широко известном формате обмена изображениями GIF, где кодовые слова являются просто индексами в хеш-таблице. Вторая разновидность метода основана на идее поиска повторяющихся последовательностей в некоторой фиксированной части сообщения, т. е. в некотором окне [6].

Критерии отбора методов сжатия. Для наших целей метод сжатия должен удовлетворять ряду требований.

Метод должен давать коэффициенты сжатия не хуже чем 3—4 (см. ниже).

Декодирование должно отличаться простотой реализации и максимальной скоростью работы, а значит, нежелательно наличие таких операций, как поиск или умножение. Переменная длина кодового слова также усложняет реализацию.

Метод должен легко настраиваться на работу с данными разной разрядности, поскольку пиксел исходного изображения может быть представлен числом с разрядностью от 8 до 32 бит.

Оценим указанные выше методы по приведенным критериям. Основными недостатками метода Хаффмана для нашего случая являются переменная длина кодового слова и малая степень сжатия. Первая из этих особенностей делает очень трудной реализацию алгоритма аппаратно. Кроме того, размер таблицы кодировки может достигать $2^{P8}P$ байт. Если пиксел кодируется тремя

байтами ($P = 3$), то кодовая таблица может стать очень большой. При разделении цветовых компонент пиксела и кодировании их отдельно коэффициент сжатия не превысит 8, но на практике намного меньше. Кроме того, данный метод требует предварительного просмотра всего сообщения для построения таблицы Хаффмана. Если под сообщением понимать один кадр, то необходимо загружать новую таблицу для каждого кадра, если же собирать статистику всего фильма, то следует загружать таблицу один раз за фильм. В последнем случае очевидно, что для каждого кадра таблица будет далека от оптимальной и возможны случаи, когда отдельные сжатые кадры будут иметь слишком большие размеры, т. е. потребуется более высокая скорость передачи сжатых данных. В литературе описаны способы кодирования методом Хаффмана с использованием предсказания, где применяется локальная корреляция между пикселами [7]. В этом случае кодируются не сами отсчеты, а например, разность между текущим пикселом и предыдущим. Этот способ дает некоторое улучшение степени сжатия, однако не устраняет всех упомянутых выше недостатков метода Хаффмана.

Первый метод сжатия типа LZW обеспечивает гораздо лучший коэффициент сжатия, но этот метод также использует кодовые слова переменной длины. Кроме того, работа с данными большой разрядности (больше 8) в этом случае также весьма затруднительна, поскольку минимальная длина кодового слова равна $8P + 1$ бит. Другим недостатком метода является сложность необходимых структур данных.

Вторая разновидность LZW-кода, а именно кодирование с контекстной подстановкой [6], гораздо лучше подходит к поставленной задаче.

Суть предлагаемого метода. Рассмотрим его работу на примере. Пусть есть исходные данные

А Б В Г А Б В Д В Г ... .

После кодирования имеем

А Б В Г (−4,3) Д (−6,2) ...,

где подстрока «А Б В» заменяется на указатель местоположения такого же набора символов в предыдущей части сообщения (относительно текущего) и счетчик длины этой последовательности. Разрядности указателя и счетчика оказывают непосредственное влияние на эффективность кодирования. Так, максимальный коэффициент сжатия не превышает максимального значения счетчика. Если под счетчик мы отводим 6 бит, то теоретически можно получить коэффициент сжатия, равный 64. Оптимальное соотношение разрядностей счетчика и указателя может изменяться в каждом конкретном случае.

Ссылки на предыдущее вхождение подстроки возможны лишь в пределах, определяемых разрядностью указателя, поэтому для поиска повторений нужно держать в памяти лишь некоторую часть исходных данных. Например, при разрядности указателя 10 бит нужен буфер под 1024 символа, т. е. размер окна равен 1024 символам.

Для поиска повторяющихся последовательностей используется следующий алгоритм.

А. Берем символ, на который указывает указатель текущего положения (текущий символ).

Б. Просматриваем предыдущую часть сообщения, начиная от текущего положения минус размер окна, с целью нахождения символа, совпадающего с текущим. Если такого символа не находим, текущий символ передаем в выходной поток, указатель текущего положения увеличиваем на единицу и переходим на п. А.

В. Если символ, совпадающий с текущим символом, найден, то сравниваем две последовательности: одна начинается с текущего символа, другая — с только что найденного. Запоминаем длину совпавших частей и относительный адрес совпавшего символа. Затем опять просматриваем сообщение внутри окна, начиная с позиции на единицу больше той, где был найден предыдущий

совпадающий символ. Если опять найдется совпадающая последовательность и ее длина окажется больше предыдущей, то заменяем запомненные значения длины и адреса на новые. Продолжаем процесс, пока не просмотрим все окно, т. е. до текущего символа.

Г. Увеличиваем указатель текущего положения на найденную длину, передаем в выходной поток значение счетчика и ссылки и переходим на п. А.

Таким образом, в любом случае находится повторяющаяся последовательность максимальной длины.

Достоинством данного алгоритма является постоянство длины кодового слова. Фактически у нас есть три типа данных: признак (один бит), ссылка (указатель плюс счетчик занимают 16 бит) и исходные данные (8 бит). Признаки могут быть сгруппированы по 8 либо по 16 бит. Работа с этими данными не представляет труда. Принцип работы алгоритма не зависит от разрядности данных (числа бит на пиксел), хотя если данные не упакованы в байты или слова, реализация алгоритма несколько усложняется. В алгоритме декодирования отсутствуют такие операции, как поиск, что делает декодирование быстрым.

Экспериментальные данные. В экспериментах использовались кадры из полностью синтезированных фильмов с разрешением 384 × 288 в формате RGB по 8 бит на компоненту (размер файла 331776 байт). Исследовались следующие алгоритмы: Huff — статический алгоритм Хаффмана со сбором статистики по кадру (при этом кадр рассматривается как байтовый файл), GIF — первый из упомянутых алгоритмов класса LZW (кадр также рассматривается как байтовый файл), JPEG — стандартный (общедоступный вариант программной реализации от марта 1992 года) алгоритм при установке параметра качества 100 (без квантования), LZW — предлагаемый алгоритм.

Все методы были реализованы автором программно.

Результаты приведены в таблице в виде размера сжатого кадра в байтах и соответствующего коэффициента сжатия для набора кадров, представленных на рис. 1—6, взятых из реальных синтезированных фильмов. Данные кадры синтезированы с разрешением 768 × 576, затем преобразованы в формат 384 × 576, откуда выбраны только четные строки. Для всех алгоритмов, кроме JPEG, кадры представлены в виде, где компоненты R, G и B разделены и записаны одна за другой. Таким образом в точности повторен процесс производства видеопродукции для чересстрочного телевидения. Хотя очевидно, что выбор только четных строк (или только нечетных) уменьшает коэффициент сжатия при использовании предлагаемого алгоритма.

Поскольку предлагаемый алгоритм в отличие от других легко настраивается на работу с пикселами (24 бита), в колонке LZW24 приведены соответствующие результаты для тех же кадров, но в формате, где компоненты каждого

Рисунок	Huff	GIF	LZW	JPEG	LZW24
1	149134 2,22	113246 2,93	91938 3,61	126636 2,62	93180 3,56
2	145146 2,29	62256 5,33	64264 5,16	35259 9,41	50376 6,59
3	149170 2,22	75433 4,40	82884 4,0	59353 5,59	77148 4,30
4	248190 1,34	103335 3,21	102370 3,24	63439 5,23	93180 3,56
5	261452 1,27	90735 3,66	84462 3,93	57204 5,80	83134 3,99
6	121740 2,73	76327 4,35	82152 4,04	35193 9,43	66840 4,96

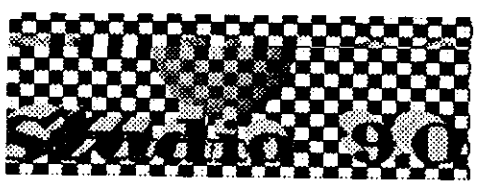

Рис. 1

Рис. 2

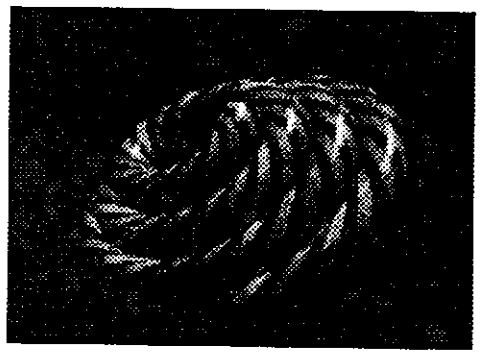

Рис. 3

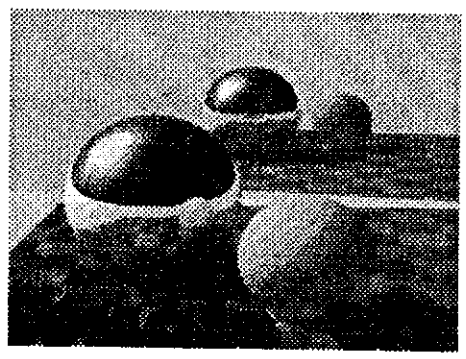

Рис. 4

Рис. 5

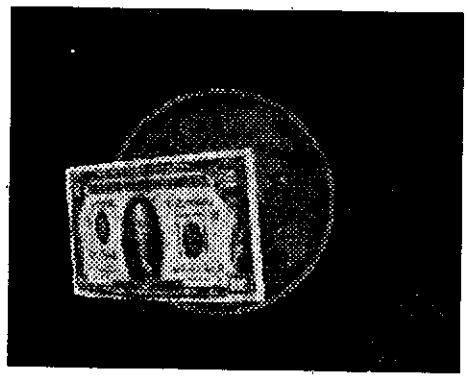

Рис. 6

пиксела *R*, *G* и *B* идут одна за другой, как в стандартном полноцветном TIFF-файле.

Сравнение алгоритма JPEG с другими не совсем корректно, поскольку JPEG использует промежуточные преобразования. Его можно рассматривать лишь как некий эталон.

Как видно из таблицы, алгоритм Хаффмана не дает сколько-нибудь существенного сжатия и явно проигрывает всем остальным. Алгоритм GIF дает примерно такой же коэффициент сжатия, как и исследуемый, но отличается большой сложностью декодирования. Предлагаемый алгоритм дает хорошее по сравнению с другими сжатие, одновременно удовлетворяя и остальным требованиям.

Возможная аппаратная конфигурация. Блок-схема одного из возможных вариантов реализации предлагаемого метода показана на рис. 7.

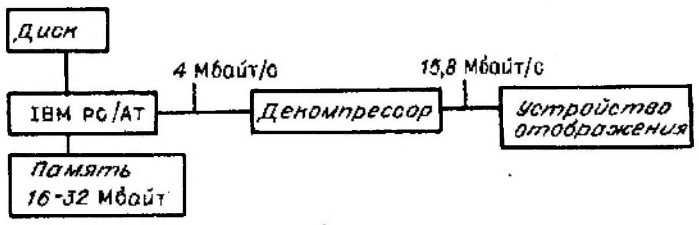

Рис. 7

Синтезированный в нереальном времени фильм вначале считывается с жесткого диска в большую память компьютера, а затем передается в декомпрессор для декодирования и вывода с нужной скоростью.

Как было показано, для записи кадра в устройство отображения за 20 мс необходима скорость передачи информации 15,8 Мбайт/с.

При среднем коэффициенте сжатия, равном 4, необходима скорость передачи сжатой информации из компьютера в декомпрессор, равная 15,8/4 = = 4 Мбайт/с, что вполне возможно при использовании достаточно мощного компьютера типа AT-386 с 32-разрядной шиной. При кодировании пикселов, а не байтов запись в устройство отображения может производиться по 24 бита за цикл, т. е. скорость работы декомпрессора может быть снижена в 3 раза по сравнению с байтовой реализацией.

Заключение. Таким образом, исследуемый алгоритм дает неплохие результаты по сравнению с другими общеизвестными алгоритмами, отличается предельной простотой декомпрессии, большой скоростью выдачи данных и отсутствием потерь информации. Дальнейшие работы должны быть направлены на поиск предварительного преобразования изображений, учитывающего используемый метод сжатия и характер входного сигнала, для повышения коэффициента сжатия.

Автор выражает благодарность А. М. Ковалеву за ценные замечания, высказанные при подготовке работы.

СПИСОК ЛИТЕРАТУРЫ

1. **Peng H. Ang.** Video compression makes big gains // IEEE Spectrum.—1991.—28, N 1.—P. 16.
2. Integrated Information Technology Inc. (IIT): Using the IIT Vision Processor in JPEG Applications, September, 1991.
3. **Huffman D. A.** A method for the construction of minimal-redundancy codes // Proc. IRE.—1952.—40.—P. 1098.
4. **Lelewer Debra A.** Data compression // ACM Comput. Surv.—1987.—19, N 3.—P. 261.
5. **Knuth D. E.** Dynamic Huffman coding // J. Algo.—1985.—6.—P. 163.
6. **Edward R. Fiala.** Data compression with finite windows // Commun. ACM.—1989.—32, N 4.—P. 490.
7. **Ефимов В. М., Золотухин Ю. Н., Колесников А. Н.** Оценка эффективности некоторых алгоритмов сокращения избыточности информации при абсолютной точности воспроизведения // Автометрия.—1991.—№ 6.

Поступила в редакцию 27 января 1993 г.