РТС-1201_РАЗРАБОТКА АППАРАТНО-ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ РЕАЛИЗАЦИИ МЕТОДОВ СЖАТИЯ ВИДЕОИНФОРМАЦИИ НА БАЗЕ ВЕЙВЛЕТ-ФРАКТАЛЬНЫХ АЛГОРИТМОВ

Есиков Д.С., студент 3 курса, Голиков А.М., научн. рук. доцент каф. РТС

Один и тот же алгоритм сжатия часто можно реализовать разными способами. Многие известные алгоритмы, такие как RLE, LZW или JPEG, имеют десятки различающихся реализаций. Кроме того, у алгоритмов сжатия бывает несколько явных параметров, варьируя которые, можно изменять характеристики процессов архивации и разархивации. При конкретной реализации эти параметры фиксируются, исходя из наиболее вероятных характеристик входных изображений, требований на экономию памяти, требований на время архивации и т.д. Поэтому у алгоритмов сжатия одного семейства лучший и худший коэффициенты сжатия могут отличаться, но качественно картина не изменится.

Фрактальный алгоритм

Фрактальная архивация основана на том, что мы представляем изображение в более компактной форме — с помощью коэффициентов системы итерируемых функций (Iterated Function System — далее по тексту как IFS). Прежде, чем рассматривать сам процесс архивации, разберем, как IFS строит изображение, т.е. процесс декомпрессии.

Строго говоря, IFS представляет собой набор трехмерных аффинных преобразований, в нашем случае переводящих одно изображение в другое. Преобразованию подвергаются точки в трехмерном пространстве (x_координата, y_координата, яркость).

Наиболее наглядно этот процесс продемонстрировал Барнсли в своей книге “Fractal Image Compression”. Там введено понятие Фотокопировальной Машины, состоящей из экрана, на котором изображена исходная картинка, и системы линз, проецирующих изображение на другой экран:

Линзы могут проецировать часть изображения произвольной формы в любое другое место нового изображения.

Области, в которые проецируются изображения, не пересекаются.

Линза может менять яркость и уменьшать контрастность.

Линза может зеркально отражать и поворачивать свой фрагмент изображения.

Линза должна масштабировать (уменьшать) свой фрагмент изображения.

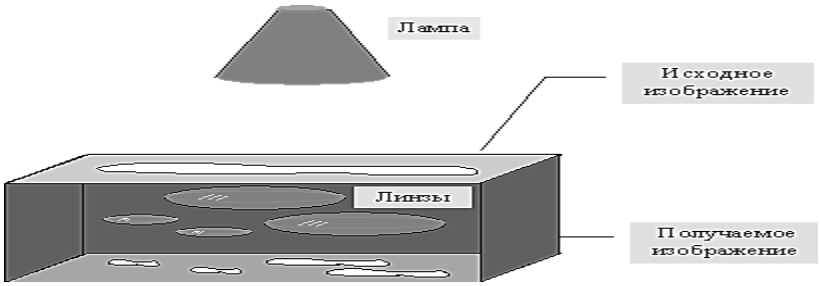

Рис. 1. Принцип линзы

Расставляя линзы и меняя их характеристики, мы можем управлять получаемым изображением. Одна итерация работы Машины заключается в том, что по исходному

изображению с помощью проектирования строится новое, после чего новое берется в качестве исходного. Утверждается, что в процессе итераций мы получим изображение, которое перестанет изменяться. Оно будет зависеть только от расположения и характеристик линз, и не будет зависеть от исходной картинки. Это изображение называется “неподвижной точкой” или аттрактором данной IFS. Соответствующая теория гарантирует наличие ровно одной неподвижной точки для каждой IFS.

Поскольку отображение линз является сжимающим, каждая линза в явном виде задает самоподобные области в нашем изображении. Благодаря самоподобию мы получаем сложную структуру изображения при любом увеличении. Таким образом, интуитивно понятно, что система итерируемых функций задает фрактал (нестрого — самоподобный математический объект).

Наиболее известны два изображения, полученных с помощью IFS: “треугольник Серпинского” и “папоротник Барнсли”. “Треугольник Серпинского” задается тремя, а “папоротник Барнсли” четырьмя аффинными преобразованиями (или, в нашей терминологии, “линзами”). Каждое преобразование кодируется буквально считанными байтами, в то время как изображение, построенное с их помощью, может занимать и несколько мегабайт.

Рис. 2. Изображение, построенное фрактальным методом.

Фактически, фрактальная компрессия — это поиск самоподобных областей в изображении и определение для них параметров аффинных преобразований.

В худшем случае, если не будет применяться оптимизирующий алгоритм, потребуется перебор и сравнение всех возможных фрагментов изображения разного размера. Даже для небольших изображений при учете дискретности мы получим астрономическое число перебираемых вариантов. Причем, даже резкое сужение классов преобразований, например, за счет масштабирования только в определенное количество раз, не дает заметного выигрыша во времени. Кроме того, при этом теряется качество изображения. Подавляющее большинство исследований в области фрактальной компрессии сейчас направлены на уменьшение времени архивации, необходимого для получения качественного изображения.

Определение. Преобразование $w: R^2 \to R^2$, представимое в виде

$$w(\bar{x}) = w\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \bullet \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

где a, b, c, d, e, f действительные числа и $(x \quad y) \in R^2$ называется двумерным аффинным преобразованием.

Определение. Преобразование $w: R^3 \to R^3$, представимое в виде

$$w(\bar{x}) = w\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a & b & t \\ c & d & u \\ r & s & p \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ q \end{pmatrix}$$

где a, b, c, d, e, f, p, q, r, s, t, u действительные числа и $(x \quad y \quad z) \in R^3$ называется трехмерным аффинным преобразованием.

Определение. Пусть $f: X \to X$ — преобразование в пространстве X. Точка $x_f \in X$ такая, что $f(x_f) = x_f$ называется неподвижной точкой (аттрактором) преобразования.

Определение. Преобразование $f: X \to X$ в метрическом пространстве (X, d) называется сжимающим, если существует число s: $0 \le s < 1$, такое, что

$d(f(x), f(y)) \le s \cdot d(x, y) \quad \forall x, y \in X$

Замечание: Формально мы можем использовать любое сжимающее отображение при фрактальной компрессии, но реально используются лишь трехмерные аффинные преобразования с достаточно сильными ограничениями на коэффициенты.

Теорема. (О сжимающем преобразовании)

Пусть $f: X \to X$ в полном метрическом пространстве (X, d). Тогда существует в точности одна неподвижная точка $x_f \in X$ этого преобразования, и для любой точки $x \in X$ последовательность $\{ f^n(x): \quad n = 0,1,2... \}$ сходится к x_f.

Более общая формулировка этой теоремы гарантирует нам сходимость.

Определение. Изображением называется функция S, определенная на единичном квадрате и принимающая значения от 0 до 1 или $S(x, y) \in [0...1] \quad \forall x, y \in [0...1]$

Пусть трехмерное аффинное преобразование $w_i: R^3 \to R^3$, записано в виде

$$w_i(\bar{x}) = w_i\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & p \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ q \end{pmatrix}$$

и определено на компактном подмножестве D_i декартова квадрата [0..1]x[0..1]. Тогда оно переведет часть поверхности S в область R_i, расположенную со сдвигом (e,f) и поворотом, заданным матрицей

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

При этом, если интерпретировать значение S как яркость соответствующих точек, она уменьшится в p раз (преобразование обязано быть сжимающим) и изменится на сдвиг q.

Определение. Конечная совокупность W сжимающих трехмерных аффинных преобразований w_i, определенных на областях D_i, таких, что $w_i(D_i) = R_i$ и $R_i \cap R_j = \phi \quad \forall i \ne j$, называется системой итерируемых функций (IFS).

Системе итерируемых функций однозначно сопоставляется неподвижная точка — изображение. Таким образом, процесс компрессии заключается в поиске коэффициентов системы, а процесс декомпрессии — в проведении итераций системы до стабилизации полученного изображения (неподвижной точки IFS). На практике бывает достаточно 7-16

итераций. Области R_i в дальнейшем будут именоваться ранговыми, а области D_i — доменными.

Построение алгоритма

Основной задачей при компрессии фрактальным алгоритмом является нахождение соответствующих аффинных преобразований. В самом общем случае мы можем переводить любые по размеру и форме области изображения, однако в этом случае получается астрономическое число перебираемых вариантов разных фрагментов, которое невозможно обработать на текущий момент даже на суперкомпьютере.

Все области являются квадратами со сторонами, параллельными сторонам изображения. Это ограничение достаточно жесткое. Фактически мы собираемся аппроксимировать все многообразие геометрических фигур лишь квадратами.

При переводе доменной области в ранговую уменьшение размеров производится ровно в два раза. Это существенно упрощает как компрессор, так и декомпрессор, т.к. задача масштабирования небольших областей является нетривиальной.

Все доменные блоки — квадраты и имеют фиксированный размер. Изображение равномерной сеткой разбивается на набор доменных блоков.

Доменные области берутся “через точку” и по X, и по Y, что сразу уменьшает перебор в 4 раза.

При переводе доменной области в ранговую поворот куба возможен только на 00, 900, 1800 или 2700. Также допускается зеркальное отражение. Общее число возможных преобразований (считая пустое) — 8.

Масштабирование (сжатие) по вертикали (яркости) осуществляется в фиксированное число раз — в 0,75.

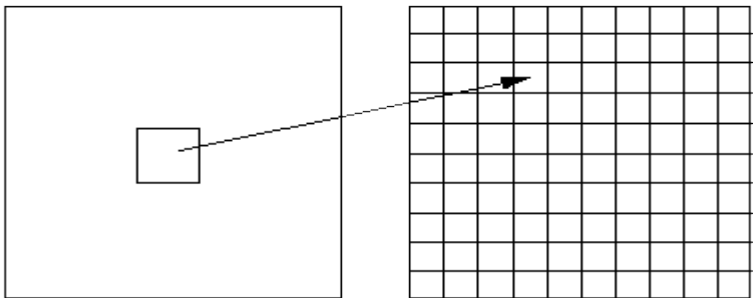

Рис. 3. Масштабирование изображения

Эти ограничения позволяют:

Построить алгоритм, для которого требуется сравнительно малое число операций даже на достаточно больших изображениях.

Очень компактно представить данные для записи в файл. Нам требуется на каждое аффинное преобразование в IFS:

два числа для того, чтобы задать смещение доменного блока. Если мы ограничим входные изображения размером 512x512, то достаточно будет по 8 бит на каждое число.

три бита для того, чтобы задать преобразование симметрии при переводе доменного блока в ранговый.

7-9 бит для того, чтобы задать сдвиг по яркости при переводе.

Информацию о размере блоков можно хранить в заголовке файла. Таким образом, мы затратили менее 4 байт на одно аффинное преобразование. В зависимости от того, каков размер блока, можно высчитать, сколько блоков будет в изображении. Таким образом, мы можем получить оценку степени компрессии.

Например, для файла в градациях серого 256 цветов 512x512 пикселей при размере блока 8 пикселей аффинных преобразований будет 4096 (512/8x512/8). На каждое потребуется 3.5

байта. Следовательно, если исходный файл занимал 262144 (512x512) байт (без учета заголовка), то файл с коэффициентами будет занимать 14336 байт. Коэффициент архивации — 18 раз. При этом мы не учитываем, что файл с коэффициентами тоже может обладать избыточностью и архивироваться методом архивации без потерь, например LZW.

Для каждого рангового блока делаем его проверку со всеми возможными доменными блоками (в том числе с прошедшими преобразование симметрии), находим вариант с наименьшей мерой L2 (наименьшим среднеквадратичным отклонением) и сохраняем коэффициенты этого преобразования в файл. Коэффициенты — это (1) координаты найденного блока, (2) число от 0 до 7, характеризующее преобразование симметрии (поворот, отражение блока), и (3) сдвиг по яркости для этой пары блоков. Сдвиг по яркости вычисляется как:

$$q = \left[\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} - \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij} \right] / n^2 ,$$

где rij — значения пикселей рангового блока (R), а dij — значения пикселей доменного блока (D). При этом мера считается как:

$$d(R,D) = \sum_{i=1}^{n} \sum_{j=1}^{n} (0.75 r_{ij} + q - d_{ij})^2 .$$

Мы не вычисляем квадратного корня из L2 меры и не делим ее на n, поскольку данные преобразования монотонны и не помешают нам найти экстремум, однако мы сможем выполнять на две операции меньше для каждого блока.

Схема алгоритма декомпрессии изображений

Декомпрессия алгоритма фрактального сжатия чрезвычайно проста. Необходимо провести несколько итераций трехмерных аффинных преобразований, коэффициенты которых были получены на этапе компрессии.

В качестве начального может быть взято абсолютно любое изображение (например, абсолютно черное), поскольку соответствующий математический аппарат гарантирует нам сходимость последовательности изображений, получаемых в ходе итераций IFS, к неподвижному изображению (близкому к исходному). Обычно для этого достаточно 16 итераций.

Поскольку мы записывали коэффициенты для блоков Rij (которые, как мы оговорили, в нашем частном случае являются квадратами одинакового размера) последовательно, то получается, что мы последовательно заполняем изображение по квадратам сетки разбиения использованием аффинного преобразования.

Как можно подсчитать, количество операций на один пиксель изображения в градациях серого при восстановлении необычайно мало (N операций “+”, 1 операций “* ”, где N — количество итераций, т.е. 7-16). Благодаря этому, декомпрессия изображений для фрактального алгоритма проходит быстрее декомпрессии, например, для алгоритма JPEG, в котором на точку приходится (при оптимальной реализации операций обратного ДКП и квантования) 64 операции “+” и 64 операции “? ” (без учета шагов RLE и кодирования по Хаффману!). При этом для фрактального алгоритма умножение происходит на рациональное число, одно для каждого блока. Это означает, что мы можем, во-первых, использовать целочисленную рациональную арифметику, которая существенно быстрее арифметики с плавающей точкой. Во-вторых, умножение вектора на число — более простая и быстрая операция, часто закладываемая в архитектуру процессора (процессоры SGI, Intel MMX...), чем скалярное произведение двух векторов, необходимое в случае JPEG. Для полноцветного изображения ситуация качественно не изменяется, поскольку перевод в другое цветовое пространство используют оба алгоритма.

Оценка потерь и способы их регулирования

При кратком изложении упрощенного варианта алгоритма были пропущены многие важные вопросы. Например, что делать, если алгоритм не может подобрать для какого-либо фрагмента изображения подобный ему? Достаточно очевидное решение — разбить этот фрагмент на более мелкие, и попытаться поискать для них. В то же время понятно, что эту процедуру нельзя повторять до бесконечности, иначе количество необходимых преобразований станет так велико, что алгоритм перестанет быть алгоритмом компрессии. Следовательно, мы допускаем потери в какой-то части изображения.

Для фрактального алгоритма компрессии, как и для других алгоритмов сжатия с потерями, очень важны механизмы, с помощью которых можно будет регулировать степень сжатия и степень потерь. К настоящему времени разработан достаточно большой набор таких методов. Во-первых, можно ограничить количество аффинных преобразований, заведомо обеспечив степень сжатия не ниже фиксированной величины. Во-вторых, можно потребовать, чтобы в ситуации, когда разница между обрабатываемым фрагментом и наилучшим его приближением будет выше определенного порогового значения, этот фрагмент дробился обязательно (для него обязательно заводится несколько “линз”). В-третьих, можно запретить дробить фрагменты размером меньше, допустим, четырех точек. Изменяя пороговые значения и приоритет этих условий, мы будем очень гибко управлять коэффициентом компрессии изображения в диапазоне от побитового соответствия до любой степени сжатия. Заметим, что эта гибкость будет гораздо выше, чем у ближайшего “конкурента” — алгоритма JPEG.

Характеристики фрактального алгоритма

Коэффициенты компрессии: 2-2000 (Задается пользователем). Полноцветные 24 битные изображения или изображения в градациях серого без резких переходов цветов (фотографии). Желательно, чтобы области большей значимости (для восприятия) были более контрастными и резкими, а области меньшей значимости — неконтрастными и размытыми. Может свободно масштабировать изображение при разархивации, увеличивая его в 2-4 раза без появления “лестничного эффекта”. При увеличении степени компрессии появляется “блочный” эффект на границах блоков в изображении.

Waveet

Английское название рекурсивного сжатия — wavelet. На русский язык оно переводится как волновое сжатие, и как сжатие с использованием всплесков. Этот вид архивации известен довольно давно и напрямую исходит из идеи использования когерентности областей. Ориентирован алгоритм на цветные и черно-белые изображения с плавными переходами. Идеален для картинок типа рентгеновских снимков. Коэффициент сжатия задается и варьируется в пределах 5-100. При попытке задать больший коэффициент на резких границах, особенно проходящих по диагонали, проявляется “лестничный эффект” — ступеньки разной яркости размером в несколько пикселей.

Идея алгоритма заключается в том, что мы сохраняем в файл разницу — число между средними значениями соседних блоков в изображении, которая обычно принимает значения, близкие к 0.

Так два числа a2i и a2i+1 всегда можно представить в виде b1i=(a2i+a2i+1)/2 и b2i=(a2i-a2i+1)/2. Аналогично последовательность ai может быть попарно переведена в последовательность b1,2i.

Разберем конкретный пример: пусть мы сжимаем строку из 8 значений яркости пикселей (ai): (220, 211, 212, 218, 217, 214, 210, 202). Мы получим следующие последовательности b1i, и b2i: (215.5, 215, 215.5, 206) и (4.5, -3, 1.5, 4). Заметим, что значения b2i достаточно близки к 0. Повторим операцию, рассматривая b1i как ai. Данное действие выполняется как бы рекурсивно, откуда и название алгоритма. Мы получим из (215.5, 215, 215.5, 206): (215.25,

210.75) (0.25, 4.75). Полученные коэффициенты, округлив до целых и сжав, например, с помощью алгоритма Хаффмана с фиксированными таблицами, мы можем поместить в файл.

Заметим, что мы применяли наше преобразование к цепочке только два раза. Реально мы можем позволить себе применение wavelet- преобразования 4-6 раз. Более того, дополнительное сжатие можно получить, используя таблицы алгоритма Хаффмана с неравномерным шагом (т.е. нам придется сохранять код Хаффмана для ближайшего в таблице значения). Эти приемы позволяют достичь заметных коэффициентов сжатия.

Алгоритм для двумерных данных реализуется аналогично. Если у нас есть квадрат из 4 точек с яркостями a2i,2j, a2i+1, 2j, a2i, 2j+1, и a2i+1, 2j+1, то

$$b^1_{i,j} = (a_{2i,2j} + a_{2i+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1}) / 4$$

$$b^2_{i,j} = (a_{2i,2j} + a_{2i+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1}) / 4$$

$$b^3_{i,j} = (a_{2i,2j} - a_{2i+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1}) / 4$$

$$b^4_{i,j} = (a_{2i,2j} - a_{2i+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1}) / 4$$

Используя эти формулы, мы для изображения 512x512 пикселей получим после первого преобразования 4 матрицы размером 256x256 элементов: Рассмотрим таблицы, в которых сводятся воедино параметры различных алгоритмов сжатия изображений, рассмотренных нами выше.

Таблица 1. Сравнение алгоритмов.

Алгоритм	Особенности изображения, за счет которых происходит сжатие
RLE	Подряд идущие одинаковые цвета: 2 2 2 2 2 2 15 15 15
LZW	Одинаковые подцепочки: 2 3 15 40 2 3 15 40
Хаффмана	Разная частота появления цвета: 2 2 3 2 2 4 3 2 2 2 4
CCITT-3	Преобладание белого цвета в изображении, большие области, заполненные одним цветом
Рекурсивный	Плавные переходы цветов и отсутствие резких границ
JPEG	Отсутствие резких границ
Фрактальный	Подобие между элементами изображения

Таблица 2. Параметры алгоритмов сжатия

Алгоритм	Сжатия	Симметричность по времени	На что ориентирован	Потери	Размерность
RLE	2, 2, 0.5	1	3,4-х битные	нет	1D
LZW	000, 4, 5/7	1.2-3	1-8 битные	нет	1D
Хаффмана	1.5, 1	1-1.5	8 битные	нет	1D
CCITT-3	13(3), 5, 0.25	~1	1-битные	нет	1D
JBIG	30 раз	~1	1-битные	нет	2D
Lossless JPEG	2раза	~1	24-битные, серые	нет	2D
JPEG	20 раз	~1	24-битные, серые	да	2D
Рекурсивное сжатие	2-200 раз	1.5	24-битные, серые	да	2D
Фрактальный	2-2000 раз	1000-10000	24-битные, серые	да	2.5D

В приведенной таблице отчетливо видны тенденции развития алгоритмов графики последних лет:

- ориентация на фотореалистичные изображения с 16 миллионами цветов (24 бита);

- использование сжатия с потерями, возможность за счет потерь регулировать качество изображений;
- использование избыточности изображений в двух измерениях;
- появление существенно несимметричных алгоритмов;
- увеличивающаяся степень сжатия изображений.

ЛИТЕРАТУРА

1. Джефф Просис. Фракталы и сжатие данных // PC Magazine, November 8, 1994, p. 289
2. Уэлстид С. Фракталы и вейвлеты для сжатия изображений в действии. - М.:Издательство Триумф, 2003. – 320 с
3. Д. Сэломон. Сжатие данных, изображения и звука. — М.: Техносфера, 2004.