Усовершенствованный алгоритм сжатия изображений на основе ИФС

Владимир Гребеник
Московский Государственный Университет имени М.В, Ломоносова
Москва, Россия

Аннотация

Одной из главных проблем в компрессии изображений на основе теории итерированных функциональных систем (ИФС) является недопустимо большое время кодирования. В данной статье представлен усовершенствованный алгоритм ИФС-сжатия (или *фрактального* сжатия) цветных изображений. Предлагаемый алгоритм позволяет значительно снизить время построения фрактального кода, а также улучшить ряд других характеристик существующих алгоритмов фрактального сжатия. Помимо стандартного фрактального кодирования используется метод векторного квантования и оригинальный способ восстановления гладких областей, сохраняющие независимость полученного кода от разрешения исходного образа. Проведены исследования влияния изометрических преобразований на качество генерируемого кода.

Ключевые слова: *сжатие изображений, фрактал, ИФС.*

1. ВВЕДЕНИЕ

Идея сжатия изображений, основанного на теории ИФС (*итерированных функциональных систем*), была впервые высказана М. Барнсли в 1987 г. Впоследствии предложения Барнсли были модифицированы и дополнены, и возможности нового метода сжатия, называемого также *фрактальным*, широко обсуждались в научных кругах. Суть фрактального метода состоит в аппроксимации исходного образа *неподвижной точкой* некоторого сжимающего отображения. В теории фракталов эта неподвижная точка называется *аттрактором*. В отличие от традиционных способов компрессии (например, JPEG), использующих локальные зависимости на отдельных участках изображения, кодирование с помощью ИФС основано на глобальной взаимосвязи фрагментов изображения, точнее, на его *самоподобии*. Первый шаг в автоматизации процедуры поиска фрактального кода (т.е. сжимающего отображения) был сделан А. Жакином [1], который предложил разбить исходный образ на небольшие фрагменты (называемые *ранговыми*), и, используя набор простых преобразований, сопоставлять этим фрагментам подобные им *области-домены*, также взятые из исходного изображения. Если все преобразования, используемые при отображении доменов в ранговые области, обладают свойством *сжимаемости*, то полученный набор данных преобразований (фрактальный код), будучи примененным к произвольному начальному приближению, сойдется к исходному изображению. Этот подход позволил в принципе реализовать автоматическое ИФС-кодирование, но на практике время создания такого кода все еще слишком велико. Предлагаемый алгоритм позволяет значительно снизить время построения фрактального кода, а также улучшить ряд других характеристик существующих алгоритмов фрактального сжатия.

2. МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ МЕТОДА ФРАКТАЛЬНОГО СЖАТИЯ

Пусть $\mathbf{x} = (x_1, x_2, \ldots x_n)^T \in \mathbf{R}^n$ - исходное изображение (сигнал). Рассмотрим аффинное преобразование

$$\mathrm{W}: \mathbf{x} \to \mathrm{W}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}, \quad (1)$$

состоящее из матрицы **A** и вектора смещения **b**. Будем рассматривать только W, обладающие свойством *сжимаемости*:

$$\exists\, s \in [0,1): \; d(\mathrm{W}(\mathbf{x}), \mathrm{W}(\mathbf{y})) < sd(\mathbf{x}, \mathbf{y}) \;\; \forall \mathbf{x}, \mathbf{y} \in \mathbf{R}^n \quad (2)$$

Тогда существует единственная неподвижная точка отображения W:

$$\exists\, \mathbf{x}_f = \mathrm{W}(\mathbf{x}_f) \quad (3)$$

Процесс кодирования состоит в определении **A** и **b** таких, что расстояние $d(\mathbf{x}_0, \mathbf{x}_f)$ между исходным сигналом $\mathbf{x}_0$ и неподвижной точкой $\mathbf{x}_f$ отображения W минимально, а сами **A** и **b** имеют сравнительно простое представление. По полученному ИФС-коду можно восстановить его неподвижную точку (а, следовательно, и исходный сигнал), решая итеративным образом уравнение $\mathbf{x}_f = \mathrm{W}(\mathbf{x}_f)$. Начиная с произвольного начального приближения $\mathbf{x}_0 \in \mathbf{R}^n$, последовательность итераций $\mathbf{x}_k = \mathrm{W}^k(\mathbf{x}_0)$ сходится к искомой неподвижной точке $\mathbf{x}_f = \lim_{k \to \infty} \mathbf{x}_k$. Таким образом, если выполнены все указанные выше требования, то с помощью ИФС можно итерационным путем получить сколь угодно хорошую аппроксимацию исходного изображения из произвольного начального приближения. Но на практике решение обратной задачи построения ИФС-кода требует проведения слишком большого объема вычислений. *Теорема о коллаже*, доказанная М. Барнсли, позволила значительно упростить процедуру построения фрактального кода. Согласно этой теореме, вместо

минимизации расстояния $d(\mathbf{x}_0, \mathbf{x}_f)$ между исходным образом и неподвижной точкой ИФС, можно минимизировать расстояние $d(\mathbf{x}_0, \mathrm{W}(\mathbf{x}_0))$ между исходным образом и его "коллажем", полученым с помощью данной ИФС. Хотя построенный таким образом фрактальный код является субоптимальным, этот подход является единственным применяемым на практике.

3. УСОВЕРШЕНСТВОВАННЫЙ АЛГОРИТМ ФРАКТАЛЬНОГО СЖАТИЯ

3.1 Описание алгоритма

Первым шагом работы компрессора является создание всех возможных доменов и приведение их к "нормированному" виду с динамическим рядом (разницей между максимальным и минимальным цветами) 255, так как в данной реализации для простоты под цвет отведено 8 бит, и средним цветом 128

$$\mathbf{D}^1[i] = \alpha \mathbf{D}^0[i] + \beta$$
$$\alpha = \frac{C_{max} - C_{min}}{\max\limits_{j=0,N} \mathbf{D}^0[j] - \min\limits_{k=0,N} \mathbf{D}^0[k]}$$
$$\beta = C_{avg} - \frac{\alpha}{N} \sum_{l=0,N} \mathbf{D}^0[l]$$
$$i = 0..N,\ C_{max} = 255,\ C_{min} = 0,\ C_{avg} = 128$$

Ссылки на "нормированные" домены помещаются в дерево доменов (см. 3.2). Затем каждая ранговая область в зависимости от величины своего динамического ряда помещается в один из трех классов (см. [2]). Возможны 3 случая

1. Небольшой динамический ряд (*гладкая* область). В этом случае осуществляется попытка аппроксимации рангового блока одноцветным фрагментом (см. 3.3). Если аппроксимация неудовлетворительна, область переходит в класс 3.

2. Очень большой динамический ряд. В этом случае осуществляется попытка аппроксимации рангового блока *вектором* (см. 3.4). Если аппроксимация неудовлетворительна, область переходит в класс 3.

3. Средний динамический ряд. Такие области, как и домены, приводятся к виду с единым (максимальным) динамическим рядом и средним цветом, а затем для них осуществляется поиск подходящего домена и преобразования. $\mathbf{R}_i$ аппроксимируется $\mathbf{D}_i^j$ так, чтобы

$$d(\mathbf{R}_i, \mathbf{R}_i^D) \to \min,$$
$$\mathbf{R}_i^D[l] = \alpha I_k^i(\mathbf{D}_i^j[l]) + \beta$$
$$\alpha = \frac{\max\limits_{p=0,N} \mathbf{R}_i[p] - \min\limits_{q=0,N} \mathbf{R}_i[q]}{\max\limits_{r=0,N} \mathbf{D}_i^j[r] - \min\limits_{t=0,N} \mathbf{D}_i^j[t]}$$
$$\beta = \tfrac{1}{N} \sum_{u=0,N} \mathbf{R}_i[u] - \tfrac{\alpha}{N} \sum_{v=0,N} \mathbf{D}_i^j[v]$$

3.2 Структура данных

Для поиска подходящих пар "домен-ранговая область" используется структура данных - дерево, - в котором хранится информация обо всех доменах. Перед началом кодирования происходит генерация всех возможных доменов, они "нормируются", затем разбиваются на четыре одинаковых квадратных области, вычисляются и запоминаются средние значения по этим областям. Дерево имеет пять уровней. На первом уровне хранятся все имеющиеся варианты значений среднего цвета по верхнему левому квадранту (в силу нормировки их не более 256). Каждому значению соответствует ссылка на следующий уровень, на котором хранятся все имеющиеся варианты значений среднего цвета по верхнему правому квадранту, которые, в свою очередь, ссылаются на список структур данных, содержащих информацию о средних цветах остальных квадрантов и номера доменов с таким распределением цветов. Таким образом, процесс поиска сводится к следующему:

- взять ранговую область, преобразовать ее к "нормированному" виду и посчитать средние значения по квадрантам;
- для каждого преобразования подобия осуществить следующие операции:
  - найти элемент первого уровня дерева, максимально соответствующий значению первого квадранта. Поскольку значения упорядочены, это быстрая операция. Найденному значению обязательно соответствует ссылка на следующий уровень дерева;
  - получить список элементов, содержащих варианты значений вторых квадрантов и повторить операцию поиска;
  - получить список элементов, содержащих варианты значений третьих квадрантов и повторить операцию поиска;
  - получить список элементов, содержащих варианты значений четвертых квадрантов и повторить операцию поиска;
- среди списка доменов выбрать набор доменов с подходящим динамическим рядом, если такой существует (это определяется сразу, так как первым в списке доменов стоит домен с максимальным динамическим рядом);
- для всех выбранных доменов и для всех преобразований подобия вычислить коэффициент сжатия и сдвиг яркости, выбрать домен и преобразование, дающие минимальную ошибку.

3.3 Кодирование гладких областей

Области с небольшим динамическим рядом часто могут быть без существенных потерь аппроксимированы одноцветными областями (с нулевым динамическим рядом). Но в случае, если на значительном фрагменте изображения имеет место не очень большой, но достаточно стабильный градиент,

цвет соседних блоков может отличаться весьма заметно. Чтобы избежать эффекта "блочности", а именно, по возможности делать плавным переход между границами соседних ранговых областей, при восстановлении изображения применяется специальный алгоритм. Рассмотрим гладкую ранговую область $\mathbf{R}_A$ на Рис. 1.

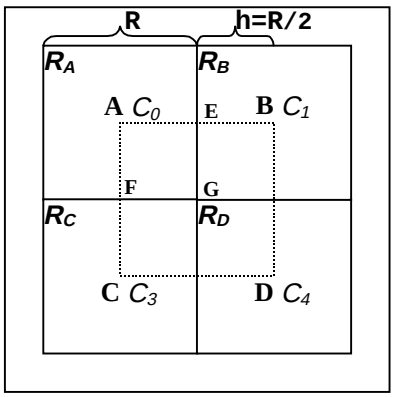

Рис.1

Каждой гладкой области в построенном фрактальном коде соответствует одна константа, обозначающая основной цвет данной области (для $\mathbf{R}_A$ это C_0). Алгоритм сглаживания при восстановлении предполагает, что данный цвет будет иметь центр области (точка A). Каждая ранговая область делится на четыре квадранта, и для каждого квадранта сглаживание производится независимо. При этом выделяются три соседних гладких ранговых области (на Рис. 1 это $\mathbf{R}_B, \mathbf{R}_C, \mathbf{R}_D$ для правого нижнего квадранта $\mathbf{R}_A$). Если один из соседних ранговых фрагментов отсутствует (текущая область находится на границе) или не является гладкой, то производится сглаживание с "виртуальной" соседней областью, имеющей тот же основной цвет, что и сглаживаемая. После определения соседних фрагментов и их основных цветов производится линейное сглаживание:

$$\mathbf{R}_A[i,j] = \frac{C_1 - C_0}{h} i + \frac{C_2 - C_0}{h} j + \frac{C_0 - C_1 - C_2 + C_3}{h^2} ij + C_0$$

Таким образом, в центральных точках всех сглаживаемых фрагментов всегда будут их "основные цвета", в углах фрагментов будут усредненные значения всех граничащих ранговых областей (значение цвета в точке G равно $(C_0 + C_1 + C_2 + C_3)/4$), а на середине границ между фрагментами будет полусумма "основных" цветов граничащих областей (значение цвета в точке E равно $(C_0 + C_1)/2$), что обеспечит корректный переход между разными сглаживаемыми квадрантами

3.4 Интеграция с методом векторного квантования

Для ранговых областей с большим динамическим рядом бывает сложно подобрать домен с еще большим динамическим рядом, поэтому для кодирования таких областей используется метод векторного квантования. Суть метода заключается в том, что для данной ранговой области подбирается наиболее похожий элемент из заранее определенного множества векторов, и в качестве идентификатора этой области хранится лишь номер данного элемента. В данной реализации была разработана система 16 двухцветных векторов (цвета кодируются отдельно как параметры преобразования), подобранных таким образом, чтобы они достаточно хорошо представляли резкие грани между фрагментами изображения, поскольку именно в таких случаях могут возникнуть проблемы с поиском подходящего домена. Такая ситуация обычно возникает из-за ограничений на динамический ряд искомого домена, который, как правило, должен быть больше динамического ряда ранговой области. На участках с резкими границами динамический ряд очень велик, и часто оказывается, что он достигает своего максимального значения. В этом случае не существует домена со строго большим динамическим рядом, и ранговая область замещается наиболее похожим вектором. Вектора выбраны так, что их структура не зависит от разрешения, и это дает ему дополнительное преимущество в сочетании с фрактальным методом.

3.5 Программная реализация

Описанный алгоритм был реализован программно, и все предложенные решения были практически опробованы. В качестве входных данных компрессора и выходных данных декомпрессора были взяты высококачественные файлы формата Targa True Color, 24 бита на пиксел. Сгенерированный фрактальный код сохранялся в файле специального формата, после чего избыточная информация удалялась методом ZIP. Программа управляется с помощью конфигурационного файла, содержащего определенный набор параметров компрессора и декомпрессора. Кроме этих параметров (их более 20), на качество получаемого изображения влияет классификация доменов и ранговых областей, также регулируемая некоторыми коэффициентами, и внутренние параметры, влияющие на логику и ход работы кодировщика. Поскольку программная реализация носила исследовательский характер, была проведена лишь самая общая оптимизация алгоритма по скорости. Испытания проводились при наличии большого объема выводимых отладочных и статистических данных, служащих для оценки эффективности самого алгоритма и влияния входных параметров. Путем удаления модулей и фрагментов программного кода, использовавшихся исключительно для тестирования, и проведения более детальной настройки быстродействия, возможно достичь значительного ускорения работы компрессора и декомпрессора.

3.6 Результаты применения предложенного алгоритма

Реализация предложенного алгоритма была протестирована на различных реальных изображениях при разных значениях параметров (расчеты велись на ПК с процессором Intel Pentium 166). Коэффициент сжатия (отношение размера исходного файла к размеру сжатого файла) в среднем составлял 15-30 раз при отношении сигнал/шум (PSNR) около 31 db. Практические исследования показали, что три компоненты комбинированного фрактального метода хорошо дополняют друг друга: на областях с малым динамическим рядом есть возможность сэкономить время кодирования и использовать меньше параметров при хранении; для областей с большим динамическим рядом часто не существует домена, удовлетворяющего условию сжимаемости и поэтому метод векторного квантования предоставляет единственную возможность для их кодирования. Все три метода не зависят от разрешения, что позволяет увеличивать его при восстановлении изображения. На изображении, полученном фрактальным методом, отсутствует блочность, вызванная повторением пикселов при обычном увеличении исходного образа, хотя полностью реалистичной гладкости добиться не удается. При кодировании полутонового изображения размером 600×600 (с **R**=4 и расстоянием между соседними доменами в 8 пикселов) полным перебором необходимо было бы осуществить около 1.2×10^8 сравнений домен-ранговая область. Если учесть, что необходимо сравнить все точки областей, получим около 1.9×10^9 "элементарных" операций сравнения. При использовании предложенной структуры данных осуществляется в среднем около 4.8×10^6 "элементарных" операций сравнения, при этом количество собственно сравнений искомых пар и количество сравнений, произведенных при обхода дерева примерно совпадают. Для построения фрактального кода цветного изображения аналогичных размеров (и осуществления утроенного числа операций) уходит около 60 секунд. В [3] описан оптимизированный алгоритм, также использующий структуру "дерево" для хранения и подбора доменов. Для сравнения, максимальный результат скорости, объявленный авторами, составляет 52 секунды при полутоновом изображении 256×256. Количество используемых доменов не является параметром, заметно влияющим на качество полученного кода. Приемлемое качество можно получить, используя уже $0.1N_R$ доменов, где N_R - число ранговых областей, причем увеличение числа доменов не приведет к какому-либо улучшению восстанавливаемого изображения. Совсем небольшие изменения в качестве произойдут, если использовать всего $0.015N_R$ доменов. В то же время важным фактором, влияющим на качество, являются 8 изометрий, используемых для расширения множества доменов. При их отсутствии восстанавливаемое изображение имеет ярко выраженную анизотропную структуру. Хотя в [4] утверждается, что изометрии используются крайне редко и могут быть отброшены без потери качества восстанавливаемого изображения (по крайней мере при кодировании методом полного перебора), предлагаемый алгоритм использует все изометрии практически с одинаковой частотой.

4. ЗАКЛЮЧЕНИЕ

Предложен новый алгоритм фрактального сжатия цветных изображений, позволяющий значительно ускорить процесс кодирования. Дано детальное описание структуры алгоритма, его основных частей и особенностей. Представлены результаты практических испытаний алгоритма применительно к цветным изображениям высокого разрешения.

5. ЛИТЕРАТУРА

[1] A. E. Jacquin, Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations, IEEE Transactions on Image Processing 1992 Vol. 1 No.1.

[2] B. Ramamurthi, A. Gersho, Classified Vector Quantization of Images, IEEE Transactions on Communication 1986, Vol. 34..

[3] B. Bani-Eqbal, Speeding Up Fractal Image Compression, Technical report, University of Manchester, 1994.

[4] D. Saupe, The Futility Of Square Isometries In Fractal Image Compression, IEEE International Conference on Image processing, 1996

Author

Vladimir Grebenik, Moscow State University.
E-mail: grebenik@tlsoft.ru

Improved Algorithm of Image Compression Based on IFS

Abstract

One of the biggest problems in the IFS-based image compression is very long encoding time. In this paper we present new improved algorithm of IFS-compression (also called *fractal compression*) of color images. The algorithm we propose allows to reduce significantly the time of building of the fractal code and enhance some other characteristics of existing fractal image compression algorithm. Along with “standard” fractal encoding in our algorithm we use also Vector Quantisation and original method of restoration of smooth blocks. This combined method keeps resolution independence of usual fractal encoders. The influence of isometries on the quality of IFS-code was studied.

Keywords: image compression, fractal, IFS