УДК 004:681.3

Фрактальные алгоритмы сжатия, некоторый опыт их сравнения

Бубличенко А.В., Беловодский В.Н.
Донецкий национальный технический университет
alexander@bliqo.com

Abstract

Bublichenko A.V., Belovodskiy V.N. "Fractal Compression Algorithms, Some Experience Comparing Them" *The results of computational experiments and comparison analysis of two basic fractal image compression algorithms are described. Their modifications targeting further efficiency improvements are given.*
Key words: fractal, image compression algorithm, comparison, modification.

Введение

Идея фрактального метода сжатия изображений была сформулирована в начале 1980-х годов [1] и концептуально заключается в следующем. Для заданного изображения по специальным правилам строится система сжимающих отображений, для которых оно является аттрактором. Тогда, в силу теоремы о неподвижной точке, итерационный процесс, сформированный на базе этих отображений, сходится к своему аттрактору при любом выборе начального изображения. Эффект сжатия достигается за счет того, что вместо исходного изображения для его восстановления необходимо хранить лишь коэффициенты преобразования.

К числу отличительных можно отнести ряд особенностей, обуславливающих интерес к данному методу сжатия на протяжении более двух десятилетий, а именно:

– высокий коэффициент сжатия (т.е. отношение объема исходной информации к объему кода). Так, для изображений, обладающих самоподобием, он достигает 2000;

– независимость качества декодированного изображения от масштаба его просмотра;

– существенная асимметричность, т.е. время кодирования заметно больше времени декодирования.

Одной из главных проблем фрактального метода сжатия изображений остается высокое время кодирования.

Постановка задачи

Привлекательность самой идеи фрактального метода сжатия и его достоинства притягивает исследователей к развитию и совершенствованию алгоритмов его реализующих. Предлагаемые модификации касаются практически всех его сторон. В большинстве опубликованных работ по данной теме предлагаются различные методы, направленные на снижение объема перебора доменных блоков и, как следствие, – сокращение времени кодирования. Так, в работе [2] затрагивается способ разбиения доменов и предлагается ограничиваться теми, которые окаймляют ранги. В работе [3] используется дискретное косинусоидальное преобразование для классификации всех доменных блоков на некоторое количество классов, в работе [4] для этого применяются R-деревья, а в работе [5] – самоорганизующаяся нейронная сеть. В работах [6] и [7] авторы используют генетический алгоритм для поиска оптимальных доменов. В работе [8] поиск ограничивается посредством меры информационной энтропии доменов. Другой, менее разработанной областью исследований, является применение нелинейных сжимающих отображений (способов аппроксимации ранговых блоков доменными блоками) [2].

Однако, возможно в силу коммерческой составляющей работ данного направления и, как следствие, отсутствия удовлетворительных описаний рассматриваемых алгоритмов, предлагаемые модификации воспринимаются, скорее, как результат интуитивных и спонтанных импровизаций, нежели обоснованный итог некоторой серии вычислительных экспериментов. Вместе с тем, каждый исследователь, начинающий путь в этой области, нуждается в наличии тщательного сопоставительного анализа основных алгоритмов. Проведение и последующее изложение такого анализа и является основной целью данной работы.

Сравниваемые алгоритмы

Рассматриваемые алгоритмы подробно описаны в литературе [1], их можно отнести к числу основных. Приведем их краткое описание.

Базовый алгоритм. Условно включает следующие этапы.

Шаг 1. В изображении f выделяют множество доменных блоков. Они могут перекрываться, а величина перекрытия

определяется специально предусмотренным параметром.

Шаг 2. Изображение разбивают на не перекрывающиеся ранговые блоки $\{R_k\}$. Они могут не быть одинакового размера, т.к. используется адаптивное разбиение методом квадро-дерева с переменным размером блоков. Это позволяет плотно заполнить ранговыми блоками маленького размера части изображения, содержащие мелкие детали.

Шаг 3. Для каждого рангового блока перебирают доменные блоки. При этом предусматривают изменение ориентации доменов (3 варианта вращения, 2 варианта вращения с отражением, 2 варианта отражения и 8-ой вариант – исходная ориентация без изменений). При каждом из вариантов ориентации домен сжимают до размеров рангового блока и вычисляют оптимальные значения коэффициентов a_{ij} и b_{ij} преобразования

$$w_{ij}(d_{xy}) = a_{ij} \cdot d_{xy} + b_{ij}$$

методом наименьших квадратов и по формуле

$$L_{kij} = L(R_k, D'_{ij}) = \left(\sum_{R_k} \left| w_{ij}(d_{xy}) - r_{xy} \right| \right) \Big/ \left(256 \cdot N_{R_k} \right)$$

вычисляют нормированное значение параметра $L(R_k , D'_{ij})$, который характеризует соответствие преобразованного сжатого i-го доменного блока $w_{ij}(D'_{ij})$ в его j-ой ориентации ранговому блоку R_k. Здесь $r_{xy} \in R_k$, $d_{xy} \in D'_{ij}$, D'_{ij} – i-ый доменный блок в j-ой ориентации, сжатый до размеров рангового блока, N_{R_k} – количество пикселей в ранговом блоке. На этом шаге алгоритм работает в одном из двух режимах, выбираемом пользователем, – с поиском и без поиска наилучшего домена. В *режиме поиска наилучшего домена* для каждого ранга перебираются *все* домены, и выбирается тот i-ый домен и его j-ая ориентация, значение L_{kij} которого является минимальным среди всех остальных. В *режиме без поиска наилучшего домена* полный перебор доменов останавливают, как только обнаруживается такой i-ый домен и его j-ая ориентация, что его значение L_{kij} не превышает заданной допустимой погрешности (например, $L_{kij} \le 0.05$). В обоих режимах, если после перебора всех доменных блоков не нашлось таких, значения L_{kij} которых не превышают значение заданной допустимой погрешности, то делается проверка – находится ли рассматриваемый ранговый блок на максимально допустимом уровне разбиения. Если нет, то этот ранговый блок разбивают на меньшие блоки и проводят данный шаг алгоритма для них. Если да, то из всех доменов выбирают тот домен и его вариант ориентации D_{ij}, для которого значение L_{kij} является минимальным среди всех остальных, и считают рассматриваемый ранговый блок покрытым этим доменом.

Шаг 3 требует наибольших вычислений, т.к. для каждого рангового блока R_k алгоритм перебирает все (или многие, в зависимости от режима работы) доменные блоки и их варианты ориентации, проводя над каждым занимающие много машинного времени попиксельные операции изменения ориентации и нахождения коэффициентов преобразования. Хорошее сжатие зависит от возможности найти хорошее соответствие между доменными и ранговыми блоками без необходимости глубокого разбиения ранговых блоков. Чрезмерно глубокое разбиение рангов приводит к слишком большому их количеству, что ухудшает коэффициент сжатия, а недостаточно глубокое – к плохому качеству закодированного изображения.

FE-алгоритм. Сравнение рангов с доменами в базовом алгоритме требует значительных вычислительных ресурсов. С целью их снижения в FE-алгоритме (Feature Extraction – выделение особенностей) выделяется пять характеристик, описывающих доменные и ранговые блоки. И первоначально, проводится именно их сравнение, что значительно сокращает объем вычислений. Эти характеристики следующие:

а) стандартное отклонение

$$\sigma = \sqrt{\frac{\sum_I \left(p_{x,y} - \mu \right)^2}{N_I}},$$

где I – сегмент изображения, N_I – количество пикселей в сегменте I, $p_{x,y}$ – значение пикселя в точке (x, y), μ – среднее значение пикселя в сегменте I;

б) асимметрия

$$a = \frac{\sum_I \left(p_{x,y} - \mu \right)^3}{N_I \cdot \sigma^3};$$

в) межпиксельная контрастность

$$c = \frac{\sum_I \left(\left| p_{x,y} - p_{x-1,y} \right| + \left| p_{x,y} - p_{x,y-1} \right| \right)}{N_I};$$

г) коэффициент бета, характеризующий отличие значений пикселей от значения центрального пикселя

$$\beta = \frac{\sum_{x=1}^{I_W} \sum_{y=1}^{I_H} \left(\omega - \sqrt{\left(x - \frac{I_W}{2} \right)^2 + \left(y - \frac{I_H}{2} \right)^2} \right) \cdot \left(p_{x,y} - \mu \right)}{\sum_{x=1}^{I_W} \sum_{y=1}^{I_H} \left(\omega - \sqrt{\left(x - \frac{I_W}{2} \right)^2 + \left(y - \frac{I_H}{2} \right)^2} \right)^2},$$

где $\omega = \dfrac{\sum_{x=1}^{I_W} \sum_{y=1}^{I_H} \sqrt{\left(x - \frac{I_W}{2} \right)^2 + \left(y - \frac{I_H}{2} \right)^2}}{N_I}$;

д) максимальный градиент (g) – максимум из горизонтального (h) и вертикального (v) градиентов

$$h = \frac{\sum_{x=1}^{I_W}\sum_{y=1}^{I_H}\left(x - \frac{I_W}{2}\right)\cdot\left(p_{x,y} - \mu\right)}{\sum_{x=1}^{I_W}\sum_{y=1}^{I_H}\left(x - \frac{I_W}{2}\right)^2},$$

$$v = \frac{\sum_{x=1}^{I_W}\sum_{y=1}^{I_H}\left(y - \frac{I_H}{2}\right)\cdot\left(p_{x,y} - \mu\right)}{\sum_{x=1}^{I_W}\sum_{y=1}^{I_H}\left(y - \frac{I_H}{2}\right)^2},\ g = \max(v, h),$$

где I_W, I_H – соответственно ширина и высота (в пикселях) сегмента изображения.

Сам алгоритм включает следующие этапы.

Шаг 1. Аналогично шагу 1 базового алгоритма.

Шаг 2. Дополняя шаг 2 базового алгоритма, производится вычисление и хранение значений вектора характеристик для каждого доменного блока.

Шаг 3. При обработке рангового блока сначала вычисляют его вектор характеристик, затем вычисляют расстояния между вектором характеристик данного ранга и вектором характеристик каждого домена по формуле

$$d = \sum_{j=1}^{5}\left|f_j^R - f_j^D\right|,$$

где f_j^R и f_j^D – это j-ые характеристики рангового и доменного блоков соответственно. Для последующего полного сравнения отбирается только заданный q процент доменов (например, q = 2%) с минимальными значениями расстояний между векторами характеристик. Последующие действия аналогичны тем, которые выполняются в шаге 3 базового алгоритма, но с той лишь разницей, что при переборе доменов рассматриваются только выбранные q % и наилучший выбирается из них.

Отметим, что такая процедура отбора доменов является своеобразным фильтром, который существенно ограничивает их перебираемое количество.

Сравнительный анализ, выводы

Для проведения анализа обоих алгоритмов было разработано программное обеспечение на языке C# с использованием Microsoft .NET 2.0. Оно позволяет кодировать и декодировать изображения рассмотренными выше алгоритмами. Пользователь имеет возможность задавать настройки, регулирующие качество кодируемых изображений, кроме этого, предусмотрены также следующие средства анализа: отображение структуры доменов и рангов, как в виде таблицы со значениями их свойств, так и визуально на изображении; вычисление средней пиксельной ошибки между исходным и закодированным изображением; подсчет количества обработанных и использованных при кодировании рангов и доменов; определение времени кодирования и декодирования.

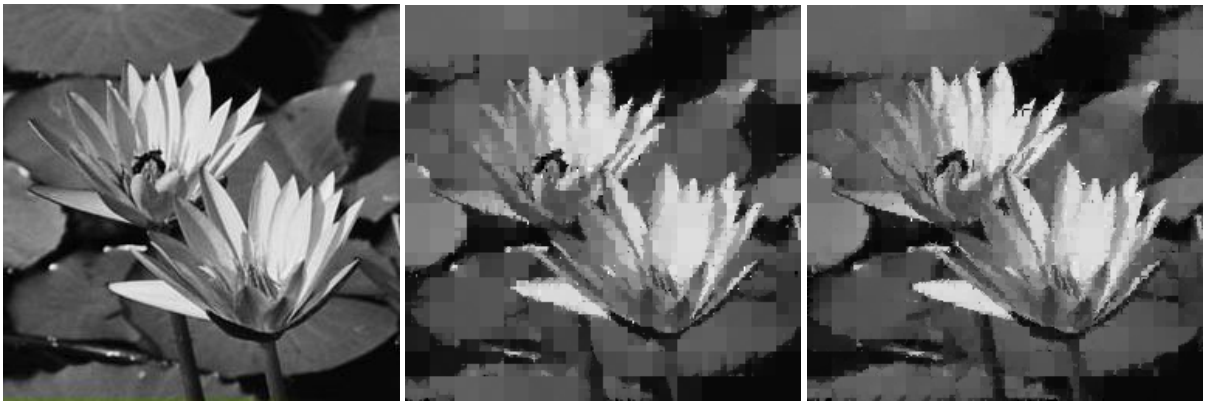
С использованием этого программного обеспечения была выполнена серия экспериментов, в которых использовалось произвольно выбранное изображение размером 256×256 пикселей, представленное на рис. 1. Исходные данные, параметры настроек и результаты экспериментов приведены в табл. 1. В процессе экспериментов время кодирования в обоих алгоритмах не ограничивалось. Каждый последующий эксперимент отличался от предыдущего более высокими значениями параметров, определяющими качество кодируемого изображения. Опишем некоторые из этих параметров. Так, начальный и максимальный уровни разбиения доменов регулируют их количество и размеры. Коэффициент перекрытия определяет, насколько сильно соседние домены перекрываются (при минимальном значении 0 – они не перекрываются вообще, а при максимальном значении 1 – перекрываются так, что остается неперекрытой область шириной ровно в один пиксель). Количество доменов зависит исключительно от этих параметров. Начальный уровень разбиения рангов задает начальное минимальное их количество и максимально возможный их размер, а максимальный уровень разбиения – их максимально допустимое количество и минимально возможный размер. Эти уровни, по сути, являются уровнями разбиения рангов методом квадро-дерева. Количество рангов по окончании кодирования зависит не только от этих параметров, но и от допустимой погрешности (см. описание алгоритмов). Средняя пиксельная ошибка показывает, насколько декодированное изображение отличается от исходного, и определяется по формуле

$$e = 100\cdot\left(\sum_{x=1}^{I_W}\sum_{y=1}^{I_H}\left|p_{xy} - p'_{xy}\right|\right)\Big/\left(256\cdot N_I\right),$$

где $p_{x,y}$, $p'_{x,y}$ – значение пикселя в точке *(x, y)* исходного и декодированного изображений соответственно, I_W, I_H – соответственно ширина и высота (в пикселях) изображения, N_I – количество пикселей в изображении.

Отметим, что параметры, определяющие количество доменов, непосредственно влияют на время кодирования и качество декодированного изображения, и не отражаются на коэффициенте сжатия.

Параметры же, регулирующие количество и размер рангов, влияют как на время кодирования и декодирования, так и на качество декодированного изображения. А окончательное число рангов однозначно определяет коэффициент сжатия.



а) б) в)

Рисунок 1 – Исходное (а) и декодированные изображения, предварительно закодированные базовым алгоритмом (б) и FE-алгоритмом (в)

Таблица 1 – Результаты экспериментов

Параметры	**№ эксперимента**			
	1	2	3	4
FE-алгоритм				
Количество доменов	225	841	2977	4562
нач.ур.дом.	3	3	3	3
макс.ур.дом.	3	3	4	4
Коэфф. перекрытия доменов	0.5	0.25	0.3	0.25
Количество рангов	1651	1504	1264	1228
Нач. уровень разбиения рангов	4	4	4	4
Макс. уровень разбиения рангов	6	6	6	6
Допуст.погр.	0.05	0.05	0.05	0.05
Искать наилучший домен	–	–	–	–
Ср. пикс. ошибка, %	3.79	3.92	3.73	3.69
Коэфф.сжатия	10	11	13	13
Время кодирования, (с)	4.58	12.86	30.25	49.39
Время декодирования, (с)	1.91	1.62	1.41	1.25
Базовый алгоритм				
Количество доменов	225	841	2977	4562
нач.ур.дом.	3	3	3	3
макс.ур.дом.	3	3	4	4
коэф. перекрытия доменов	0.5	0.25	0.3	0.25
Количество рангов	1150	1075	940	928
Нач. уровень разбиения рангов	4	4	4	4
Макс. уровень разбиения рангов	6	6	6	6
Допуст.погр.	0.05	0.05	0.05	0.05
Искать наилучший домен	Нет	Нет	Нет	Нет
Ср. пикс. ошибка, %	4.01	4.18	4.37	4.29
Коэфф.сжатия	14	15	17	18
Время кодирования, (с)	154.81	508.69	735.28	1068
Время декодирования, (с)	1.15	1.05	0.97	0.95

Анализ полученных результатов позволяет сделать следующие выводы:

1. FE-алгоритм, по сравнению с базо-вым, позволяет сжимать изображения в десятки раз меньшее время при одинаковых параметрах настроек алгоритмов (см. рис. 2).

2. Различаются средние пиксельные ошибки алгоритмов (см. рис. 3). В зависимости от настроек, различие находится в пределах 5-16%.

3. В результате кодирования базовым алгоритмом во всех случаях достигается в среднем на 30-40% более высокий коэффициент сжатия (см. рис. 4).

4. Визуальное качество декодированно-го изображения (см. рис. 1) заметно лучше при использовании FE-алгоритма.

5. Время декодирования закодирован-ного изображения при различных настройках кодирования меньше при использовании базового алгоритма во всех экспериментах (см. рис. 5).

Некоторые из этих отличий вполне предсказуемы. Действительно, в FE-алгоритме, вместо сравнения блоков изображения происходит сравнение их векторов характеристик, что безусловно отражается на быстродействии. Различие пиксельных ошибок и коэффициентов сжатия скорее всего объясняется тем, что при использовании FE-алгоритма в результате кодирования получается бóльшее количество рангов и они меньшего размера, что обеспечивает более высокую детализацию изображения, то есть – лучшее визуальное качество. Это подтверждает и рис. 6, на котором представлено закодированное обоими алгоритмами изображение с сеткой рангов. Заметим, что хотя при базовом алгоритме и получается на 5-16% худшее качество изображения, но при этом, достигается на 30-40% более высокий коэффициент сжатия.

Можно предположить, что причина выявленных различий заключается в том, что в FE-алгоритме при переборе доменов, отобранных по значениям их характеристик, не находится такой, который обеспечивал бы принятую допустимую погрешность соответствия ранговому блоку, в результате чего ранговый блок разбивается на более малые по размеру части. В то же время при кодировании базовым алгоритмом для тех же рангов, вполне возможно, такие домены находятся.

Для проверки этой гипотезы был разработан дополнительный программный модуль, позволяющий подсчитывать число совпадений в обоих алгоритмах при выборе домена, изображение оставалось тем же. Настройки кодирования соответствуют тем, которые использовались в четвертом эксперименте табл. 1, с той лишь разницей, что базовый алгоритм работал в режиме поиска наилучшего домена. Это означает, что при обработке каждого рангового блока осуществлялся перебор всех доменных блоков, без остановки при достижении допустимой погрешности, и выбирался действительно наилучший. Это, на наш взгляд, позволяет провести корректное сравнение.

В результате экспериментов было установлено, что для 99.73 % ранговых блоков FE-алгоритм выбрал другие домены, т.е. – не наилучшие. Таким образом, по крайней мере, для данного изображения можно утверждать, что процедура отбора доменов, принятая в FE-алгоритме, не вполне отражает близость сравниваемых блоков.

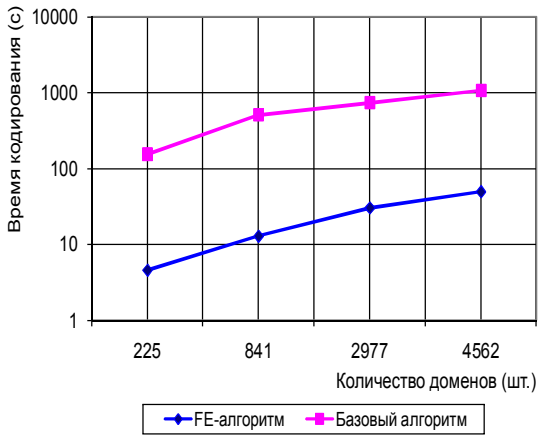


Рисунок 2 – Зависимость времени кодирования от количества доменов

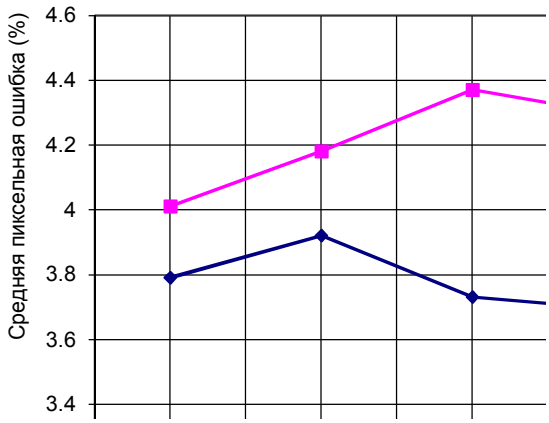


Рисунок 3 – Средняя пиксельная ошибка закодированного изображения при различных настройках (см. табл. 1) кодирования

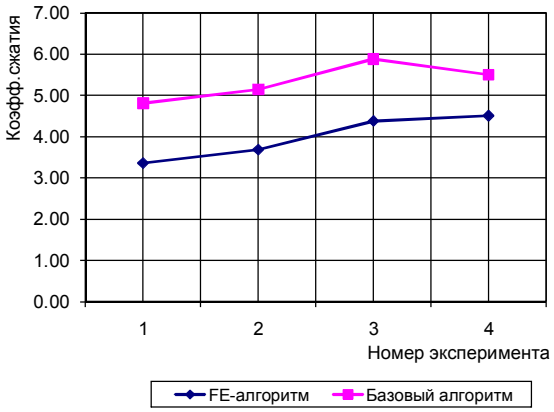


Рисунок 4 – Коэффициент сжатия изображения при различных настройках (см. табл. 1) кодирования

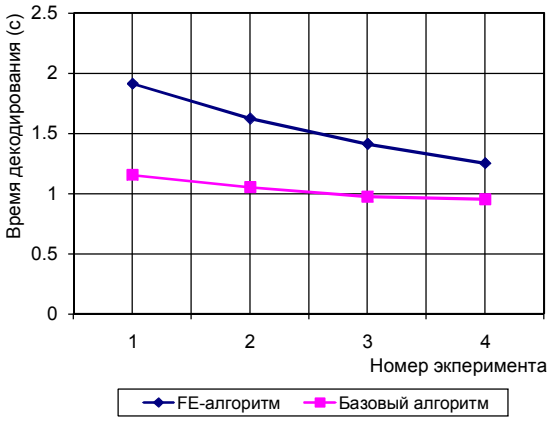
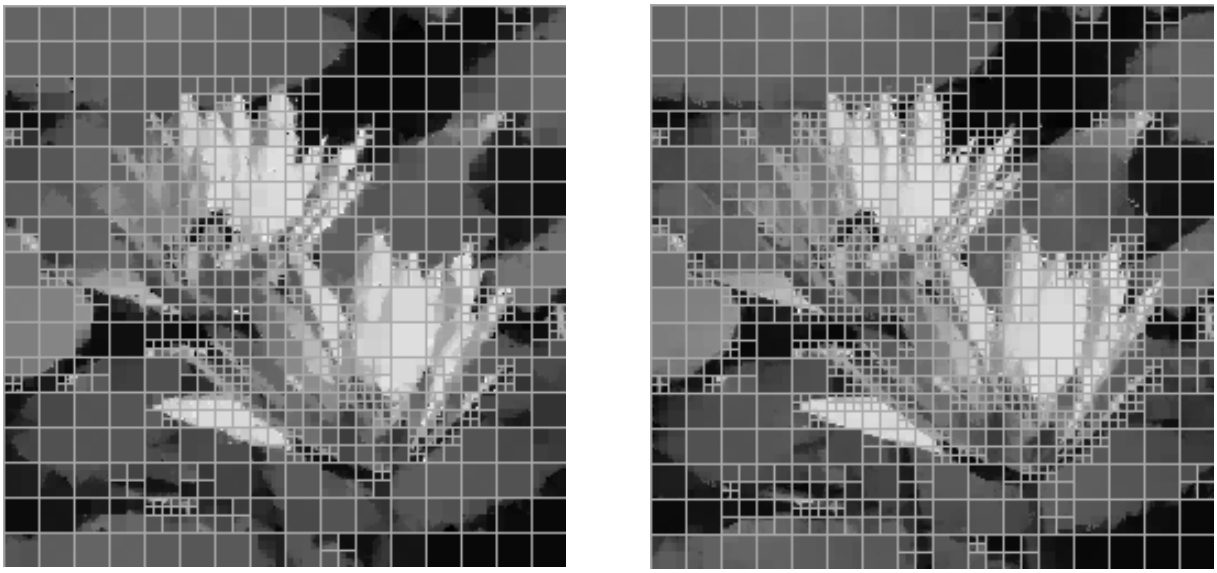


Рисунок 5 – Время декодирования закодированного изображения при различных настройках (см. табл. 1) кодирования



а) б)

Рисунок 6 – Изображение с сеткой рангов, закодированное базовым алгоритмом (а) и FE-алгоритмом (б) при одинаковых настройках

Предложения, заключения

Сформулируем некоторые соображения, сформировавшиеся в результате проведенных экспериментов.

1. Представляется бесспорным, что эффективность сжатия в немалой степени зависит от начального разбиения исходного изображения. Действительно, невключение удачного домена или его ошибочное отбрасывание сказывается как на степени сжатия, так и на качестве декодирования. Априори представляется правдоподобным, что в обычных, а не специально созданных изображениях различие соседних близких пикселей, в целом, менее разительно, чем в областях, выбираемых случайным образом. По этим соображениям, включение окаймляющих доменов в первоначально формируемое их множество представляется целесообразным.

2. Идея введения упрощенного набора критериев с целью существенного снижения объема вычислений является привлекательной, но, как показывает эксперимент, в большинстве случаев сравнения доменов с рангами этот набор не позволяет выявить оптимальный домен. Другими словами, данный набор является не вполне адекватным основному критерию. Поэтому представляется целесообразным вместо этого набора использовать непосредственно основной критерий, но применять его к уменьшенным копиям рассматриваемых пар домен-ранг. Это может быть реализовано алгоритмом, который аналогичен базовому, за исключением 2-го и 3-го шагов. На шаге 2 при генерации доменных

блоков для каждого из них строятся его уменьшенные копии. Первая копия имеет размер $k \times k$ пикселей, вторая – $2k \times 2k$ пикселей и т.д. до исходного размера домена, где k выбирается экспериментально. На шаге 3 перед началом обработки очередного рангового блока сначала строятся его уменьшенные копии аналогичным образом. Затем проводится многоуровневый отбор наилучшего доменного блока для данного ранга. На 0-м уровне сравниваются уменьшенные копии размером $k \times k$ пикселей данного ранга и домена. При сравнении запоминаются величина L_{kij} (средняя пиксельная ошибка) и наилучшая ориентация для каждого доменного блока. Для последующего сравнения отбираются только m (например, m=20) доменных блоков с наименьшим значением величины L_{kij}. На всех последующих уровнях сравниваются уменьшенные копии соответствующего размера рангового и доменных блоков, но уже только в сохраненной на первом уровне наилучшей ориентации каждого отдельного домена. Этот процесс продолжается до последнего уровня уменьшенных копий рангового блока. На последнем уровне сравнения отбирается только один доменный блок с наименьшей величиной L_{kij}. Преимущество такой модификации алгоритма заключается в сохранении основного критерия сравнения и его применении последовательно к уменьшенным копиям пар ранг-домен, что позволит на первых же уровнях сравнения отбрасывать неоптимальные домены и оставлять только перспективные.

3. Напрашивается также соображение, непосредственно вытекающее из теоретических основ фрактального метода сжатия. Так, из теоремы коллажа [9] следует, что степень отличия аттрактора системы сжимающих отображений от исходного изображения не в последнюю очередь определяется качественными характеристиками формируемых отображений. Их повышения естественно ожидать при переходе от простейших, грубых, линейных моделей, используемых в базовых алгоритмах, к более сложным – нелинейным. В этой связи попытка использования таких моделей нам представляется актуальной.

Литература

1. С. Уэлстид. Фракталы и вейвлеты для сжатия изображений в действии. Учебное пособ. – М.: Издательство «Триумф», 2003, с. 77-119.
2. А.Н. Авлеева. Фрактальное сжатие изображений. Решение задач сжатия изображений с использованием систем итерированных функций. Магистерская диссертация, ДонНТУ, 2006.
3. Д.С. Ватолин. Использование ДКП для ускорения фрактального сжатия изображений. Журнал «Программирование» №3 1999, с. 51-57.
4. J. Kominek. Algorithm for Fast Fractal Image Compression. Department of Computer Science, University of Waterloo, Canada, 1995.
5. S. Welstead. Self-Organizing Neural Network Domain Classification for Fractal Image Coding. Proc. of the IASTED International Conference “Artificial Intelligence and Soft Computing”. Banff, Canada, 1997, pp. 248-251.
6. L. Vences, I. Rudomin. Genetic Algorithms for Fractal Image and Image Sequence Compression. Manuscript. Institute of Technology, University of Monterrey, 1997.
7. L. Xi, L. Zhang. A Study of Fractal Image Compression Based on an Improved Genetic Algorithm. International Journal of Nonlinear Science. Vol.3, No. 2, 2007, pp. 116-124.
8. M. Hassaballah, M.M. Makky, Youssef B. Mahdy. A Fast Fractal Image Compression Method Based Entropy. Electronic Letters on Computer Vision and Image Analysis 5(1), 2005, pp. 30-40.
9. Р. М. Кроновер. Фракталы и хаос в динамических системах. Основы теории. – М.: Постмаркет, 2000.