УДК 004.62

В.К. Гулаков, С.Б. Клепинин

ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ФРАКТАЛЬНОГО КОДИРОВАНИЯ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ ПЕРЦЕПТИВНЫХ ХЕШ-ФУНКЦИЙ

Рассмотрен вопрос фрактального кодирования изображения для задач распознавания образов. Проанализированы требования к вычислительным ресурсам классической реализации алгоритма фрактального кодирования. Выдвинуто и реализовано предложение по её улучшению. Представлены практические результаты, а также сравнительный анализ двух имплементаций.

Ключевые слова: фракталы, системы итерируемых функций, распознавание изображений, фрактальный код изображения, перцептивное хеширование, фрактальное кодирование.

Фракталы как математические объекты с интересными свойствами (самоподобие, дробная размерность, отсутствие четкой характеристической меры) были известны таким математикам, как Кантор, Пуанкаре, Гилберт, еще в конце XIX - начале XX века. Однако основоположником фрактальной математики считается Бенуа Мандельброт. Теория итерируемых функций, введенная Джоном Хатчинсоном, стала вторым шагом на пути практического использования фракталов. В дальнейшем эта теория была использована Михаилом Барнсли для определения теоремы коллажа, которая описывает, каким образом должна быть построена система итерируемых функций для получения *фрактального изображения*. ЭрнеЖаквин, один из студентов Барнсли, создал алгоритм, позволяющий преобразовать изображение в систему частично итерируемых функций [1].Эта реализация по сей день является основой большинства систем фрактального кодирования.

С одной стороны, геометрические фракталы обладают очень высокой визуальной сложностью, с другой стороны, эта сложность имеет маленький информационный вес, так как фракталы могут быть описаны и сгенерированы несложным алгоритмом. Также фракталы являются избыточными объектами в том смысле, что они, как правило, составлены из трансформированных копий себя или какой-то части себя. Эту избыточность и пытаются эксплуатировать фрактальные алгоритмы сжатия. В течение многих лет фрактальное кодирование использовалось только для целей компрессии изображений, было предложено множество подходов и реализаций [2]. Значительно позже фракталы стали использоваться и для задач распознавания образов. Системы строятся как на основе непосредственного сравнения векторов фрактальных кодов (наборов параметров сжимающих преобразований), так и с использованием математической основы фрактального кодирования (например, теоремы коллажа, теоремы о фиксированной точке) [3].

Фрактальный код изображения представляет собой набор сжимающих преобразований, трансформирующих доменные блоки в ранговые. Распределение соответствующих друг другу пар (доменный блок - ранговый блок) зависит от содержания изображения, а также от используемого алгоритма. Некоторые методы используют лучшее совпадение, другие - первое, удовлетворяющее определенному пороговому значению. Доменные блоки покрывают кодируемое изображение полностью и могут перекрываться. Ранговые блоки организованы в квадродерево[5] и пересекаться не могут.

Рассмотрим подробнее типичный алгоритм фрактального кодирования:

Входные данные:

- кодируемое изображение;
- начальный размер ранговых блоков в квадродереве;
- максимальная глубина квадродерева.

I. Создание пула доменных блоков:

1. Кодируемое изображение покрывается несколькими слоями доменных блоков квадратной формы. Слой – это множество доменных блоков одинакового размера. Количество слоев равно глубине квадродерева ранговых блоков. Доменный блок должен быть больше рангового блока, с которым он будет сравниваться в дальнейшем (в нашей реализации используется коэффициент 2). Устанавливается четкое соответствие между доменным слоем и уровнем квадродерева: все ранговые блоки этого уровня сравниваются только с блоками определенного доменного слоя. Доменные блоки в слое могут накладываться друг на друга, соответственно шаг размещения доменных блоков может быть фиксированным или относительным (в нашей реализации используется относительный, равный четвертой части размера доменного блока). Доменные блоки, полученные на этом шаге, будем называть *оригинальными* доменными блоками.

2. Каждый оригинальный доменный блок сжимается до размера рангового блока в соответствующем уровне квадродерева. Полученные блоки помещаются в соответствующий слой.

3. К полученным на предыдущем шаге доменным блокам применяются аффинные трансформации, такие как:

- поворот на 90 градусов;
- поворот на 180 градусов;
- поворот на 270 градусов;
- отражение относительно вертикальной оси;
- отражение относительно горизонтальной оси;
- отражение относительно главной диагонали;
- отражение относительно второстепенной диагонали.

Полученные доменные блоки помещаются в соответствующий слой. Блоки, полученные на шагах 2 и 3 данного алгоритма, будем называть *производными* доменными блоками.

II. Создание квадродерева. Кодируемое изображение покрывается сеткой непересекающихся ранговых блоков максимально допустимого размера (он передается в качестве входного параметра алгоритма). Для каждого выполняются следующие шаги:

1. Ранговый блок сравнивается с каждым доменным блоком из соответствующего доменного слоя (соответствие устанавливается так, чтобы размер рангового блока совпадал с размером доменных блоков в слое):

1.1. Подбираются значения контраста (*s*) и яркости (*o*), обеспечивающие минимальное различие рангового и доменных блоков:

$$Err = \sum_{i=1}^{n}(sd_i + o - r_i)^2. \quad (1)$$

Минимум достигается, когда частные производные функции выше по *s*и *o*равны 0.Это возможно при [3]

$$s = \frac{\alpha}{\beta};\ o = \bar{r} - \left(\frac{\alpha}{\beta}\right)\bar{d},$$

Где $\bar{d} = \frac{1}{n}\cdot\sum_{i=1}^{n} d_i; \bar{r} = \frac{1}{n}\cdot\sum_{i=1}^{n} r_i;\ \alpha = \sum_{i=1}^{n}(d_i - \bar{d})\cdot(r_i - \bar{r});\ \beta = \sum_{i=1}^{n}(d_i - \bar{d})^2$; d_i –i-я точка доменного блока; r_i–i-я точка рангового блока.

1.2. Параметры контраста и яркости, полученные на предыдущем шаге,применяются к доменному блоку и вычисляется расстояние между ранговым и доменным блоками по формуле (1).

2. Запоминается наиболее близкий доменный блок и расстояние до него.

3. Если текущий ранговый блок расположен не на максимальном уровне квадродерева, то выполняются следующие шаги:

3.1. Блок разбивается на 4 дочерних ранговых блока, для каждого из которых повторяются шаги 1–3.

3.2. Суммируется расстояние до наиболее близких доменных блоков дочерних ранговых блоков и сравнивается с аналогичным показателем родительского рангового блока. Если последний меньше, то дочерние узлы удаляются, текущий ранговый блок становится листовой вершиной квадродерева.

III. Сохранение фрактального кода. Совершается последовательный обход квадродерева, для каждой листовой вершины сохраняются (например, в CSV-файл) следующие параметры:

- координаты рангового блока;
- размер рангового блока;
- координаты оригинального доменного блока;
- размер оригинального доменного блока;
- набор трансформаций, примененных к оригинальному доменному блоку.

Если внимательно присмотреться к данному алгоритму, то можно заметить, что он подразумевает размещение в памяти доменного пула на все время кодирования. Какого же размера может достигать доменный пул? Допустим, мы кодируем изображение в градациях серого, на хранение каждого пикселя используется 4 байта (стандартная длина типа данных intв наиболее распространённых языках программирования). Обозначим через S_{DP} размер доменного пула в байтах, через S_{DL}^i - размер i-го слоя доменного пула в байтах.Тогда

$$S_{DP} = \sum_{i=1}^{n} S_{DL}^i. \quad (2)$$

Здесь *n* – максимальная глубина квадродерева;

$$S_{DL}^i = N_{DL}^i \left(S_R^i\right)^2 4, \quad (3)$$

где N_{DL}^i – количество доменных блоков в i-м слое доменного пула; S_R^i – размер рангового блока на i-й глубине квадродерева (равен размеру доменного блока в i-м слое доменного пула).

$$N_{DL}^i = N_0^i (N_T + 1), \quad (4)$$

где N_T – количество используемых трансформаций; N_0^i – количество оригинальных доменов для i-го слоя доменного пула.

$$N_0^i = \left(floor\left(\frac{S_I - 2S_R^i}{\Delta} + 1\right)\right)^2, \quad (5)$$

где $floor(x)$–функция округленияx до ближайшего целого в меньшую сторону;S_I – исходный размер кодируемого изображения;Δ – шаг размещения доменных блоков.

На основе формул (2 - 5) был рассчитан размер доменного пула для изображения размером 512 x 512 (таблица).

Таблица

Размер памяти, занимаемой под доменный пул, для изображения 512x512

i	S_I	Δ	S_R^i	N_T	N_0^i	N_{DL}^i	S_{DL}^i
1	512	8	32	7	3,249	25,992	106,463,232
2		4	16		14,641	117,128	119,939,072
3		2	8		62,001	496,008	126,978,048
Итого					79,891	639,128	353,380,352

Также был рассчитан размер доменного пула для изображений размером от 128x128 до 512x512.Получившийся тренд представлен на рис. 1.

Из представленных данных легко сделать вывод, что для кодирования одного изображения размером 512x512 потребуется ~ 350 МВ оперативной памяти, что очень много даже по современным меркам.

С другой стороны, шаги 1.1 и 1.2 представленного алгоритма являются очень тяжелыми в вычислительном смысле. Так, кодирование изображения размером 256x256 с максимальным размером рангового блока 32 пикселя и максимальной глубиной квадродерева 3 (на компьютере с процессором IntelCorei3 (2 ядра, 2.40GHz) и объемом оперативной памяти 4GB) заняло 1442767 миллисекунд (~24 минуты) (испытуемая реализация по максимуму использовала кеширование, никакое значение не вычислялось два раза).

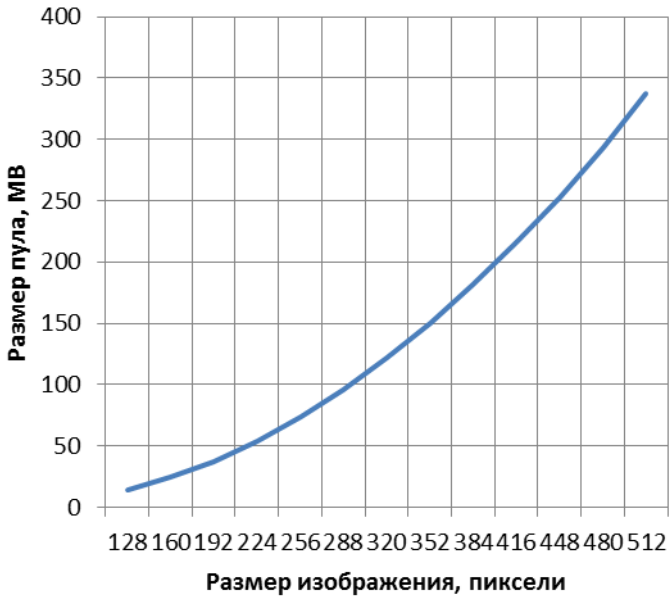

Рис. 1. Зависимость размера доменного пула от размера кодируемого изображения

Итак, как показано выше, классический алгоритм фрактального кодирования требует большого объема оперативной памяти и выполняется неприемлемо долго. Для решения этих двух проблем предлагается использовать перцептивное хеширование [4]. Идея заключается в следующем: вместо того чтобы хранить куски изображений в доменных и ранговых блоках, достаточно хранить короткий *хеш* блока и осуществлять над ним в процессе кодирования операцию сравнения для выяснения близости доменных и ранговых блоков. Для сравнения можно использовать расстояние Хэмминга. Если хеш будет достаточно коротким (например, 64 бита), это очевидным образом сократит объем занимаемой памяти и уменьшит время кодирования.

Рассмотрим один из возможных алгоритмов получения перцептивного хеша для доменного/рангового блока изображения:

1. Размер блока уменьшается до размера 8x8.Можно использовать обычный алгоритм, когда вместо четырех соседних пикселей получается один усредненной интенсивности.

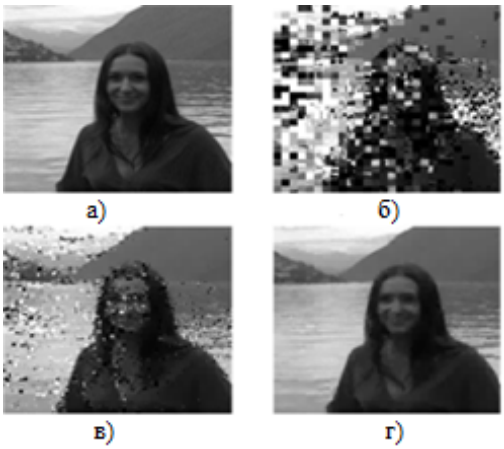

Рис. 2. Декодирование изображения, полученного с применением перцептивной хеш-функции,на основе фрактального кода: а - кодируемое изображение; б – 1-я итерация декодирования; в - 2-я итерация декодирования; г - 5-я итерация декодирования

2. Вычисляется среднее значение интенсивности для блока 8x8.

3. Каждому пикселю блока сопоставляется бит: если интенсивность пикселя больше средней – 1, в противном случае - 0.

4. Из полученных на шаге 3 значений создается 64-битный хеш.

Алгоритм достаточно простой (но не единственно возможный), что должно обеспечить приемлемое время кодирования. Сначала проводились эксперименты по проверке корректности алгоритма: было взято изображение размером 256x256 в градациях серого, закодировано, а затем декодировано. Результат проверки алгоритма представлен на рис. 2. Возможно, для применения в области сжатия изображений точность

недостаточная, но для задач распознавания вполне пригодная, так как здесь более важно, чтобы фрактальный код верно передавал структуру кодируемого изображения. Далее проводились сравнительные замеры времени кодирования одних и тех же изображений с помощью классического алгоритма и хеш-функции (использовались одни и те же максимальный размер рангового блока, максимальная глубина квадродерева). Результаты представлены на рис. 3. Так, кодирование изображения 256x256 заняло всего 84 секунды, что примерно в 17 раз лучше показателей классического алгоритма.

Если посмотреть на график, то можно увидеть, что с увеличением размера изображения эффективность алгоритма на основе перцептивных хешей, по сравнению с классическим, только увеличивается.

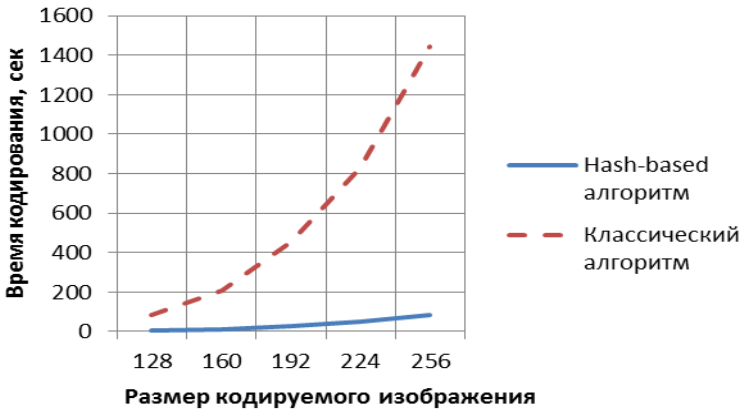

Рис. 3. Сравнение времени кодирования изображений классическим методом и с использованием перцептивных хешей

Таким образом, классический алгоритм фрактального кодирования изображений требует большого количества вычислительных ресурсов - как процессорного времени, так и оперативной памяти. Обойти данную проблему можно, отказавшись от сравнения непосредственно каждого пикселя ранговых и доменных блоков и перейдя на использование в этих целях перцептивных хешей. Представленные результаты экспериментальных замеров подтверждают сделанное предположение, показывая прирост в производительности на 1-2 порядка.

СПИСОК ЛИТЕРАТУРЫ

1. Jacquin, A.E.Fractal image coding: A review / A.E. Jacquin // Proceeding of the IEEE. – 1993. - Vol. 81.-№. 10.-P. 1451-1456.
2. Welstead, S. Fractal and wavelet image compression techniques / S. Welstead // SPIE Press.–1999.
3. Гулаков, В.К. Методы распознавания изображений на основе фрактального кодирования / В.К. Гулков, С.Б. Клепинин // Вестн.Брян. гос. техн. ун-та. – 2012. - №4(36). – С. 54 – 61.
4. Zauner, C. Implementation and Benchmarking of Perceptual Image Hash Functions / C.Zauner// Hagenberg. – 2010.
5. Гулаков, В.К. Многомерные структуры данных: монография / В.К. Гулаков, А.О. Трубаков. – Брянск: БГТУ, 2010. – 387 с.

Материал поступил в редколлегию 11.08.14.