Breaking the Time Complexity of Fractal Image Compression

Dietmar Saupe[1]
Universität Freiburg

Abstract: In fractal image compression the encoding step is computationally expensive. A large number of sequential searches through a list of domains (portions of the image) are carried out while trying to find a best match for another image portion. We show that this step can be replaced by multi-dimensional nearest neighbor search which runs in logarithmic time instead of linear time required for the common sequential search.

Keywords: Image compression, multi-dimensional search, nearest neighbors, fractals.

1 Introduction

With the ever increasing demand for images, sound, video sequences, computer animations and volume visualization, data compression remains a critical issue regarding the cost of data storage and transmission times. While JPEG currently provides the industry standard for still image compression there is ongoing research in alternative methods. Fractal image compression is one of them.

JPEG can be termed *symmetric* in the sense that the encoding and decoding phases require about the same number of operations. On the contrary, fractal image compression is *asymmetric*, requiring long encoding times while allowing fast decoding. Thus, the method seems to be most applicable where a large number of pre-encoded images have to be decoded quickly. The encoding may be done using special hardware or larger computers while the decoding and the display may be carried out in software on small machines.

The merits and drawbacks of fractal image compression in comparison to JPEG and other methods are topics of interest in the current literature, e.g., [1, 7].

Fractal image compression allows fast decoding but suffers from long encoding times. This paper introduces a new twist for the encoding process. What is the current state-of-the-art? During encoding a large pool of image subsets, called domains, has to be searched repeatedly many times, which by far dominates all other computations in the encoding process. If the number of domains in the pool is N, then the time spent for each search is *linear* in N, $O(N)$. Previous attempts to reduce the computation times employ *classification schemes* for the domains based on image features such as edges or bright

[1]Address: Institut für Informatik, Universität Freiburg, Rheinstr. 10-12, 79104 Freiburg, Germany. Email: saupe@informatik.uni-freiburg.de. This paper is available as a Technical Report of the Institut für Informatik, also via anonymous ftp at ftp.informatik.uni-freiburg.de:papers/fractals. A condensed version of this paper with the title *From classification to multi-dimensional keys* will be published in: *Fractal Encoding — Theory and Applications to Digital Images,* Y. Fisher (ed.), Springer-Verlag, New York, 1994.

spots. Thus, in each search only domains from a particular class need to be examined. However, this approach reduces only the factor of proportionality in the $O(N)$ complexity.

The main contribution of this paper is the following. We suggest to replace the domain classification by a small set of real-valued keys for each domain. These keys are carefully constructed such that searching in the domain pool can be restricted to the *nearest neighbors* of a query point. Thus, we may substitute the sequential search in the domain pool (or in one of its classes) by multi-dimensional nearest neighbor searching. There are well known data structures and algorithms for this task which operate in *logarithmic* time, $O(\log N)$, a definite advantage over the $O(N)$ complexity of the sequential search. These time savings may provide a considerable acceleration of the encoding and, moreover, an enlargement of the domain pool potentially yielding improved image fidelity.

The remainder of this paper is organized as follows. In the next section we give an overview of fractal image compression providing an orientation about where in the process the classification scheme is replaced by multi-dimensional search. In section 3 we present the definition of the multi-dimensional keys, a theorem that establishes the mathematical foundation, and some practical comments on the implications.

Figure 1: Elements of the fractal code. Left: partitioning of the image region (a square) into ranges. Center: some of the corresponding domains. Right: the affine transformation v_k for the k-th domain-range pair. For each domain-range pair (D_k, R_k) there is an (invertible) geometric transformation $u_k : D_k \to R_k$. The function f evaluates image intensities. For a point $x \in R_k$ we compute its preimage $u_k^{-1}(x)$ in the corresponding domain D_k, look up the image intensity, $f(u_k^{-1}(x))$, and finally apply the affine transformation, v_k, obtaining $Tf(x) = v_k f u_k^{-1}(x)$ for $x \in R_k$. Iteration of the image operator T produces a sequence of images that converges to an approximation of the encoded image (see figure 2).

2 Outline of Fractal Image Compression

Basically, a fractal code consists of three ingredients: a partitioning of the image region into portions R_k, called *ranges,* an equal number of other image regions D_k, called *domains* (which may overlap), and for each domain-range pair two transformations, a geometric one, $u_k : D_k \to R_k$, which maps the domain to the range, and an affine transformation, v_k, that adjusts the intensity values in the domain to those in the range (see figure 1).

The collection of transformations may act on an arbitrary image, g, producing an output image, Tg, which is like a collage of modified copies of the domains of the image g. The iteration of the image operator T is the decoding step, i.e., it yields a sequence of images which converges to an approximation of the encoded image (see figure 2).

Figure 2: Encoding and decoding for a one-dimensional example. Let $f : [0,1] \to [0,1]$ denote the 'image' to be encoded (left). We choose the same domain $[1/2, 1]$ for all ranges shown on the left. Note that the graph of f in R_1 as well as in R_3 is just a scaled down copy of the graph in the domain. In R_2 we can reproduce f by a vertical flip of the same copy, in R_4 we use the copy with an additional offset of $1/2$. The code thus employs the geometric transformations $u_k(x) = x/2 + (k-2)/4$ and the affine transformations $v_1(f) = v_3(f) = f/2$, $v_2(f) = (1-f)/2$, and $v_4(f) = (1+f)/2$. When reconstructing the original from the code we may start with an arbitrary 'image', g, and repeatedly apply the image operator T. The first two iterations Tg and T^2g are shown for the choice $g(x) = x$.

The time consuming part of the encoding step is the search for an appropriate domain for each range. The number of possible domains that theoretically may serve as candidates is prohibitively large. For example, the number of arbitrarily sized square subregions in an image of size N by N pixels is of order $O(N^3)$. Thus, one must impose certain restrictions in the specification of the allowable domains. In a simple implementation one might consider as domains, e.g., only sub-squares of a limited number of sizes and positions. This defines the so-called *domain pool*. Now for each range in the partition of the original image all elements of the domain pool are inspected: for a given range R_k and a domain D the transformations u_k and v_k are constructed such that when the domain image portion is mapped into the range the result $v_k f u_k^{-1}(x)$ for $x \in R_k$ matches the original image f as much as possible. This step uses the well-known least squares method. From all domains in the pool we select the best one, i.e., the domain D_k that yields the best least squares approximation of the original image in the range. In other words, fractal image coding consists in approximating the image as a collage of transformed pieces of itself, which can be viewed as a collection of self-similarity properties. The better the collage fits the given image the higher the fidelity of the resulting decoded image.

In this short article we cannot explain any further details and variations of fractal image compression. For introductory texts or reviews see, for example, [3, 6, 7, 11]. For a comprehensive bibliographic survey of the field of fractal image compression see our paper [13].

3 From Classification to Multi-Dimensional Keys

The classification of ranges and domains serves the purpose of reducing the number of domains in the domain pool which need to be considered as a partner for a given range. Just like in 'real life', birds of a feather flock together. For example, if the original image contains an edge running through a range, then domains which contain only 'flat' pieces of the image can be safely discarded when searching for a good match for that range. In previous schemes only a few classes were used in this way (from 3 or 4 in [10] up to 72 in [8]). Jacquin [10, 11] sorts ranges and domains into three classes (shade blocks, edge blocks, and midrange blocks) following a classification, well-known in image processing. The classification of Fisher, Jacobs, and Boss [8, 7] is made with a clever design of a variable number of classes (4–12–72) taking into account not only intensity values but also intensity variance across a domain. More recently, attempts have been made to design the set of classes adaptively, i.e. depending upon the target images. Lepsøy and Øien [12] propose an adaptive codebook clustering algorithm and Boss and Jacobs [5] consider an archetype classification based on a set of training images. In addition to the complexity reduction by using these classification schemes Bedford, Dekking, and Keane [4] suggest to use inner products with Rademacher functions to further exclude certains domains from consideration for a partner to a given range.

The innovation here is that we replace the classification concept by the assignment of multi-dimensional keys to ranges and domains. These keys form a metric space in which we have to search for closest neighbors of the range key. The approach presented here is really new. In contrast to the state-of-the-art classification schemes we may operate with the *entire* domain pool, yet we will have to deal with only (logarithmically) few domains in the actual minimization.

To simplify we present the results for a special one-dimensional case and generalize later. We consider a set of N vectors $x^{(1)}, \ldots, x^{(N)} \in \mathbf{R}^d$ (representing d pixel values in each of the N domains of the pool) and a point $z \in \mathbf{R}^d$ (range with d pixels). We let $E(x^{(i)}, z)$ denote the smallest possible least squares error of an approximation of the range data z by an affine transformation of the domain data $x^{(i)}$. In terms of a formula, this is $E(x^{(i)}, z) = \min_{a,b\in\mathbf{R}} ||z - (ae + bx^{(i)})||^2$, where $e = \frac{1}{\sqrt{d}}(1, \ldots, 1) \in \mathbf{R}^d$ is just a unit length vector with equal components. Computing the optimal a, b and the error $E(x^{(i)}, z)$ is a costly procedure, which we have to perform for all of the domain vectors $x^{(1)}, \ldots, x^{(N)}$ in order to arrive at the minimum error, given by $\min_{1\leq i\leq N} E(x^{(i)}, z)$. This staggered minimization problem needs to be solved for many query points z in the encoding process (i.e., for all ranges). The goal is to show that this search can be done in logarithmic time $O(\log N)$ in place of linear time, $O(N)$, valid for the sequential search. The following theorem provides the mathematical foundation for the solution. We use the notation $d(\cdot,\cdot)$ for the Euclidean distance and $\langle\cdot,\cdot\rangle$ for the inner product in $\mathbf{R}^d$, thus, $||x|| = d(x, 0) = \sqrt{\langle x, x\rangle}$.

Theorem. *Let* $d \geq 2$, $e = \frac{1}{\sqrt{d}}(1, \ldots, 1) \in \mathbf{R}^d$ *and* $X = \mathbf{R}^d \backslash \{re \mid r \in \mathbf{R}\}$. *Define the normalized projection operator* $\phi : X \to X$ *and the function* $D : X \times X \to [0, \sqrt{2}]$ *by*

$$\phi(x) = \frac{x - \langle x, e \rangle e}{||x - \langle x, e \rangle e||} \quad \textit{and} \quad D(x,z) = \min(d(\phi(x), \phi(z)), d(-\phi(x), \phi(z))).$$

For $x, z \in X$ *the least squares error* $E(x,z) = \min_{a,b \in \mathbf{R}} ||z - (ae + bx)||^2$ *is given by*

$$E(x,z) = \langle z, \phi(z) \rangle^2 g(D(x,z)) \quad \textit{where} \quad g(D) = D^2(1 - D^2/4).$$

Figure 3: Illustration of the geometry underlying the theorem.

Proof. The least squares approximation of z by a vector of the form $ae + bx$ (resp. $ae + b\phi(x)$) is given by the projection

$$\mathrm{Proj}(z) = \langle z, e \rangle e + \langle z, \phi(x) \rangle \phi(x) = \langle z, e \rangle e + \langle z, \phi(z) \rangle \langle \phi(x), \phi(z) \rangle \phi(x),$$

where the last equation is derived from $z = \langle z, e \rangle e + \langle z, \phi(z) \rangle \phi(z)$ (see figure 3). The least squares error in this approximation is calculated as

$$E(x,z) = ||z - \mathrm{Proj}(z)||^2 = \langle z, \phi(z) \rangle^2 (1 - \langle \phi(x), \phi(z) \rangle^2).$$

Since $d(\pm\phi(x), \phi(z)) = \sqrt{2(1 \mp \langle \phi(x), \phi(z) \rangle)}$ we have $D(x,z) = \sqrt{2(1 - |\langle \phi(x), \phi(z) \rangle|)}$. Solving for $|\langle \phi(x), \phi(z) \rangle|$ and inserting the square of the result in the formula for $E(x,z)$ completes the proof.

For an interpretation we note that for given $x \in \mathbf{R}^d$ all vectors of the form $ae + b\phi(x)$ can be represented exactly as a linear combination of x and e (with zero least squares error). These vectors form a two-dimensional subspace of $\mathbf{R}^d$, an orthonormal basis of which is given by $\phi(x)$ and e. For all x in this space $\phi(x)$ is unique up to choice of the sign. Thus, $\phi(x)$ may serve as a 'representative' of this space and will become the multi-dimensional key for searching.

The theorem states that the least squares error $E(x,z)$ is proportional to the simple function g of the Euclidean distance D between the projections $\phi(x)$ and $\phi(z)$ (or $-\phi(x)$

and $\phi(z)$). The value of the result is not in terms of a speed-up of the calculation of the least squares error, but of a more fundamental nature. Since $g(D)$ is a monotonically increasing function for $0 \leq D \leq \sqrt{2}$ we conclude that *the minimization of the least squares errors $E(x^{(i)}, z)$ for $i = 1, \ldots, N$ is equivalent to the minimization of the distance expressions $D(x^{(i)}, z)$!* Thus, we may replace the computation and minimization of N least squares errors $E(x^{(i)}, z)$ by the search for the nearest neighbor of $\phi(z) \in \mathbf{R}^d$ in the set of $2N$ vectors $\pm\phi(x^{(i)}) \in \mathbf{R}^d$.

The problem of finding closest neighbors in Euclidean spaces has been thoroughly studied in computer science. For example, a method using k-d trees that runs in expected logarithmic time is presented in [9] together with pseudo code. After a preprocessing step to set up the required k-d tree, which takes $O(N \log N)$ steps, any search for the nearest neighbors of query points can be completed in expected logarithmic time, $O(\log N)$. However, as the dimension d increases, the performance may suffer. A method that is more efficient in that respect, presented in [2], produces a list of so-called approximate nearest neighbors. The above theorem provides the link to these algorithms from computational geometry.

We conclude with some remarks on generalizations and implications.

1. In the above we have made the ideal assumption that the number of components in all vectors is the same, namely d. This is not the case in practice, where small and large ranges need to be covered by domains that can be of a variety of sizes. To cope with this difficulty, we settle for a compromise and proceed as follows. We *down-filter* all ranges and domains to some prescribed dimension of moderate size, e.g., $d = 8$. This allows the processing of arbitrary domains and ranges, however, with the implication that the formula of the theorem is no longer exact but only approximate.

2. For a given range not all domains from the pool are *admissible*. There are restrictions on the resulting number b, the contraction factor of the affine transformation (e.g., $|b| < 1$ in some implementations). This is necessary in order to ensure convergence of the iteration in the image decoding. Also one imposes bounds on the size of the domain (for a given range a suitable domain should be larger than the range, but not too large). To take that into consideration, we search the *entire* domain pool not only for the nearest neighbor of the given query point but also for, say, the next 10 or 20 nearest neighbors (this is called the all-nearest-neighbors problem and can still be solved in logarithmic time using a priority queue). From this set of neighbors the non-admissible domains are discarded and the remaining domains are compared using the ordinary least squares approach. This also takes care of the problem from the previous remark, namely that the estimate by the proposition is only approximate. Some first empirical tests (using a 256×256 image and the quicksort algorithm for computing nearest neighbors) have shown that although the best domain for a given range often was not the first entry in the priority queue it usually was among the first six or so. Thus, the result was the same as if the exhaustive search through the *entire* domain pool had been carried out.

3. Our technique for encoding one-dimensional image data readily carries over to *(2D) images*. There are two possible approaches. The first is simply to represent an image by a one-dimensional scan, e.g., by the common scan-line method or (better) using a Hilbert-

or Peano-scan. However, in the literature image domains and ranges are usually squares, rectangles or triangles. Image data from a rectangle, e.g., can first be transformed to a square, then down-filtered to, say, an array of 3 by 3 or 4 by 4 intensity values leading to $d = 9$ or 16. After applying the normalized projection operator ϕ we can use the result as a multi-dimensional key in the exact same fashion as described before.
4. We make two technical remarks concerning *memory requirements* for the k-d tree. Firstly, it is not necessary to create the tree for the full set of $2N$ keys in the domain pool. We need to keep only one multi-dimensional key per domain if we require that this key has a non-negative first component (multiply key by -1 if necessary). In this set-up a k-d tree of all $2N$ vectors has two symmetric main branches (separated by a coordinate hyperplane), thus, it suffices to store only one of them. Secondly, there is some freedom in the choice of the *geometric transformation* that maps a domain onto a range. In the one-dimensional setting we may reverse the 'direction of time'. In two dimensions a square, e.g., may undergo any of the 8 transformations of its symmetry group (rotations by multiples of 90 degrees and a flip). This will create 7 additional entries in the k-d tree, enlarging the size of the tree. However, we can get away without this tree expansion. To see this, just note that we may instead consider the 8 transformations of the *range* (or just some of them) and search the original tree for nearest neighbors of each one of them.
5. The *preprocessing time,* $O(N \log N)$, to create the data structure for the multi-dimensional search is not a limitation of the method. To see that, observe that the number of the ranges to be considered is of the same order as the number of domains, $O(N)$. Thus, the sum of all search times, including the preprocessing, is $O(N \log N)$, to be compared with $O(N^2)$ for the method using sequential search.

4 Summary

We have introduced a new technique at the core of fractal image compression which can be integrated into existing implementations. It reduces the time complexity of the encoding step thereby creating the potential for better and faster fractal image compression. Clearly, the approach proposed in this paper should be tested empirically, the results of which will be presented in a forthcoming paper.

Acknowledgements. The author thanks Klaus Bayer, Raouf Hamzaoui, Thomas Ottmann, and Sven Schuierer for fruitful discussions. Amitava Datta pointed out the relevance of the paper of Arya et al [2]. This work had been started last year while the author was at the Center for Complex Systems and Visualization at the University of Bremen.

References

[1] Anson, L. F., *Fractal image compression,* BYTE, October 1993, 195–202.

[2] Arya, S., Mount, M. M., Netanyahu, N. S., Silverman, R., Wu, A., *An optimal algorithm for approximate nearest neighbor searching,* Proc. 5th Annual ACM-SIAM

Symposium on Discrete Algorithms (1994) 573–582.

[3] Barnsley, M., Hurd, L., *Fractal Image Compression*, AK Peters, Wellesley, 1993.

[4] Bedford, T. J., Dekking, F. M., Keane, M. S., *Fractal image coding techniques and contraction operators,* Nieuw Arch. Wisk. (4), 10, no. 3 (1992) 185–218.

[5] Boss, R. D., Jacobs, E. W., *Archetype classification in an iterated transformation image compression algorithm,* in: *Fractal Encoding — Theory and Applications to Digital Images,* Y. Fisher (ed.), Springer-Verlag, New York, 1994.

[6] Fisher, Y., *A discussion of fractal image compression,* in: H.-O. Peitgen, H. Jürgens, D. Saupe, *Chaos and Fractals,* Springer-Verlag, New York, 1992.

[7] Fisher, Y., *Fractal Encoding — Theory and Applications to Digital Images,* Springer-Verlag, New York, 1994.

[8] Fisher, Y., Jacobs, E. W., Boss, R. D., *Fractal image compression using iterated transforms,* in: J. A. Storer (ed.), *Image and Text Compression,* Kluwer Academic Publishers, Boston, 1992.

[9] Friedman, J. H., Bentley, J. L., Finkel, R. A., *An algorithm for finding best matches in logarithmic expected time,* ACM Trans. Math. Software 3,3 (1977) 209–226.

[10] Jacquin, A., *Image coding based on a fractal theory of iterated contractive Markov operators, Part II: Construction of fractal codes for digital images,* Report Math. 91389-017, Georgia Institute of Technology, 1989.

[11] Jacquin, A. E., *Fractal image coding: A review,* Proceedings of the IEEE 81,10 (1993) 1451–1465.

[12] Lepsøy, S., Øien, G. E., *Fast attractor image encoding by adaptive codebook clustering,* in: *Fractal Encoding — Theory and Applications to Digital Images,* Y. Fisher (ed.), Springer-Verlag, New York, 1994.

[13] Saupe, D., Hamzaoui, R., *A guided tour of the fractal image compression literature,* in: ACM SIGGRAPH'94 Course Notes, Course 13, *New Directions for Fractal Modeling in Computer Graphics,* J. Hart (ed.), 1994. Also Technical Report, Institut für Informatik, Universität Freiburg, 1994.