AN APPLICATION OF THE FIXED POINT THEOREM TO IMAGE COMPRESSION

by

CHOKRI CHERIF

A dissertation submitted to the Graduate Faculty in Mathematics in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1999

This manuscript has been read and accepted for Graduate Faculty in Mathematics in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

9/8/99
Date

Chair of Examining Committee

9/8/99
Date

Executive Officer

Professor Burton Randol

Professor George Wolberg

Professor Robert Sibner
Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

AN APPLICATION OF THE FIXED POINT THEOREM TO IMAGE COMPRESSION

by

CHOKRI CHERIF

Adviser: Professor ROBERT SIBNER

Let W be a contraction on the complete metric space (E, d) with contractivity factor s and fixed point f. Let g be in E. Then $d(f, g) \leq (1/(1-s))d(g, W(g))$. Thus, by minimizing the distance between g and $W(g)$ (the collage of the image), we hope to minimize the distance between the fixed point f and the given image g. Of course, if the value of s is close to 1, nothing ensures that this method provides a good approximation. In this work we present a new approach of construction of the operator W that guarantees a better fidelity due to the better bound we put on $d(f, g)$, without any cost on compression.

Preface

The mathematical principle behind fractal image compression: As model for the space of monochrome images we choose a space E of bounded continuous functions $f : X \longrightarrow G$ for the simplicity of its mathematical description. The set X taken for example as the unit square represents the set of the spatial co-ordinates of the image. While the set G taken as the interval $[0, 1]$ represents the set of intensity values of the image. However, for practical applications suitable for computer processing one can prefer a discrete framework in which a spatially digitized image is modeled as a point of a finite dimensional space. Thus, a discrete grey-tone image of size $n \times n$ pixels is thought of as a point in $R^{n \times n}$. After a distance d is constructed such that (E, d) is a complete metric space, the fractal (or attractor) coding of the image f is seen as the following optimization problem: Find a contractive operator W on (E, d) whose fixed point $g = W(g)$ is the best possible approximation of f (the contraction mapping principle ensures that a fixed point g=W(g) exists and is unique). This optimization problem will be approached by means of the collage theorem

Collage Theorem. Let W be a contraction on the complete metric space (E, d) with contractivity factor s and fixed point f. Let g be in E. Then $d(f, g) \leq (1/(1-s))d(g, W(g))$. Thus, by minimizing the distance between g and $W(g)$ (the collage of the image), we hope to minimize the distance between the fixed point f and the given image g. Of course, if the value of s is close to 1, nothing ensures that this method provides a good approximation. Yet this was the original idea of Barnsley and most of the fractal based algorithms rely on the same approach. In this work we present a new approach of construction of the operator W that guarantees a better fidelity due to the better bound we put on $d(f, g)$, without any cost on compression. A detailed description of the operator as well as a complete practical algorithm are presented. In the same work we also present a classical construction based on interpolation techniques [MN], to construct W. Description of the PIFS (Partitioned Iterated Function System) and it's associated algorithm are presented where we propose to encode range blocks using linear combinations of domain blocks when a good match for a range block is not found. Computation shows that the construction is numerically stable.

ACKNOWLEDGMENTS

I wish to thank my adviser Professor Robert Sibner for his support in my mathematical achievement, and decisive help in many critical times.

I am greatly indebted to my mom. She always encouraged, helped, and supported my work in mathematics for more than ten years.

I am thankful to my brothers and sisters. Without their love, care and encouragement this work would not have existed. Finally, i want to express my appreciation to many professors of GSUC-CUNY who have influenced my study and research. For all my sincere thank you.

Manhattan, NY - May 1999 Chokri Cherif

To my mother.

Contents

0.1 Introduction

There are important image applications where some processing(such as subtraction, filtering contrast enhancement, etc...) should be applied to archived or transmitted images. In those cases lossy compression methods may destroy some of the information required during processing, or add artifacts which lead to erroneous interpretations. Quite frequently the user of those applications wants to have total control of the precision in which the image pixels are represented and prefers to have the image compressed with a lossless method. Lossless compression is also indicated for images obtained at great cost, such as space and medical images when it is unwise to discard any information that may be useful later. Nevertheless, images are frequently visually inspected, and it is interesting to have a compression scheme that simultaneously allows fast inspection and, only when necessary, exact recovery of the image. Traditionally, the user had to choose different coding methods depending whether the highest compression, the fast inspection or the fidelity is desired. In this work a special focus on fidelity is presented without affecting compression.

0.2 Basic definitions

Definition 0.2.1 *A metric space X is complete if every Cauchy sequence in X converges to a limit point in X.*

Definition 0.2.2 *Let X be a metric space, let $H(X) = \{S \subset X | S \text{ is compact}\}$. If $A \in H(X)$, then we write $A_{d(\epsilon)} = \{x | d(x,y) \leq \epsilon \text{ for some } y \in A\}$; $A_{d(\epsilon)}$ is the set of points that are in ϵ-neighborhood from A. We define the Hausdorff distance between two elements A, $B \in H(X)$ to be*

$$h_d(A,B) = max\{inf\{\epsilon | B \subset A_{d(\epsilon)}\}, inf\{\epsilon | A \subset B_{d(\epsilon)}\}\}$$

Theorem 0.2.1 *Let X be a complete metric space with metric d, then $H(X)$ with the Hausdorff metric h_d is a complete metric space.*

Note: The compactness condition in the definition of $H(X)$ is necessary in order for theorem 1 to hold.

Definition 0.2.3 *Let X be a metric space with metric d. A map $W : X \longrightarrow X$ is Lipschitz with Lipschistz factor s if there exists a positive real value s such that*

$$d(W(x), W(y)) \leq sd(x,y)$$

for all x, $y \in X$. If $s < 1$, then W is said to be contractive with contractivity s.

Theorem 0.2.2 *If $w_i : R^2 \longrightarrow R^2$ is contractive with contractivity s_i for $i = 1, 2, \ldots, n$, then*

$$W(S) = \bigcup_{i=1}^{n} w_i(S)$$

where S is a set in $H(R^2)$ is contractive in the Hausdorff metric with contractivity $s = max_{i=1,\ldots,n}\{s_i\}$

Theorem 0.2.3 *(The Contractive Mapping Fixed-Point Theorem) Let X be a complete metric space and $f : X \longrightarrow X$ be a contrative mapping. Then there exit a unique point $x_f \in X$ such that for any point $x \in X$*

$$x_f = f(x_f) = \lim_{n\to\infty} f^n(x).$$

such a point is called a fixed point or the attractor of the mapping f.

Corollary 0.2.1 *(Collage Theorem) With the hypothesis of the Contractive Mapping Fixed Point Theorem,*

$$d(x, x_f) \leq \frac{1}{1-s} d(x, f(x)).$$

Definition 0.2.4 *Let f be a Lipschitz function. If there is a number n such that $f^{on}(x)$ is contractive, then we call f eventually contractive. We call n the exponent of eventual contractivity.*

Corollary 0.2.2 *Let f be eventually contractive with exponent n; then there exists a unique fixed point $x_f \in X$ such that for any $x \in X$*

$$x_f = f(x_f) = lim_{k\to\infty} f^k(x).$$

in this case,

$$d(x, x_f) \leq \frac{1}{1-s} \frac{1-\sigma^n}{1-\sigma} d(x, f(x)),$$

where s is the contractivity of f^k and σ is the Lipschitz factor of f.

The following theorem is another version of the Collage Theorem.

Theorem 0.2.4 *Let (X, d) be a complete metric space. Given a $\mu_0 \in X$, suppose there exists a map W which is contractive with contractivity $0 \leq s < 1$, so that $d(\mu_0, W(\mu_0)) < \epsilon$. If μ_w is the fixed point of W, i.e. $W(\mu_w) = \mu_w$, then $d(\mu_0, \mu_w) < \frac{\epsilon}{1-s}$.*

In other words, this applies to the case in which a "target" μ_0 that we wish to approximate with a fixed point μ_w of an unknown mapping W. The inverse problem reduces to finding a W which minimizes the "Collage distance" $d(\mu_0, W(\mu_0))$.

Iterated function system (IFS):

Definition 0.2.5 *An N-map contractive IFS on a compact metric space (x, d) is a set of contraction maps, $W = \{w_1, w_2, \ldots, w_N\}$, where $w_i : X \longrightarrow X$. Associated with the IFS W is a set-valued mapping $\hat{W}$ which acts on nonempty compact subsets of X.*

Definition 0.2.6 *An image will be represented by a function $u : [0,1]^2 \longrightarrow R_g$, where $R_g \subseteq R^+$, is the gray level range typically in this work will be $[0, 255]$. The value $u(x)$ at a point or pixel $x \in [0,1]^2$ may be interpreted as a nonnegative gray level or brightness.*

0.3 Model Space

In these various IFS-type schemes, an image or target is represented as a point y in an appropriate complete metric space (Y, d_Y). The spaces employed by various IFS-type methods are listed below:

IFS $M(X)$, the set of nonempty compact subsets of X. **IFZS** $F^*(X)$, the set of all functions $u : X \longrightarrow [0,1]$ which are (1) upper semi continuous on (X,d) and (2) normalized, i.e. for each $u \in F^*(X)$ there exists an $x_0 \in X$ for which $u(x_0) = 1$.

Fractal Functions which also includes "fractal interpolation functions", the space of k-times continuously differentiable functions $C^k(X)$.

IFSM $L^p(X,\mu)$, the space of p-integrable function with respect to a measure μ, $1 \leq p \leq \infty$. Fractal Transforms are a special case of IFSM as are Fractal Functions. The Bath Fractal Transformation is an IFSM with place-dependent gray level maps.

IFSP $M(X)$, the set of probability measures on $B(X)$, the σ-algebra of Borel subsets of X.

In what follows, we outline a formulation of generalized fractal transformations of image functions. The mathematical setting is as follows:

1. The base space: denoted, as above by (X,d). The space representing the pixel; a compact subset of $\mathbf{R}^n$. Without loss of generality, $X = [0,1]^n$ with Euclidean metric.

2. The IFS component: For an $n \in N$, let $w = w_1, w_2, \ldots, w_n$. In many cases we can relax the condition that the w_i be contraction maps on X. Note that the sets $w_i(X)$ may overlap.

3. The image function space: $F(X) = \{u : X \longrightarrow R_g \subseteq \mathbf{R}\}$, the functions which will represent our images.

4. The gray level range R_g will denote the range of a particular class of image functions used in a given fractal transform method. (In particular applications, R_g is a bounded subset of $\mathbf{R}^+$.)

5. The gray level component: Associated with the IFS maps w will be a vector of N function $\Phi = \{\phi_1, \phi_2, \ldots, \phi_N\}$, $\phi_i : R_g \longrightarrow R_g$. We may also consider $\phi_i : R_g \times X \longrightarrow R_g$, i.e. "place-dependent" gray level maps.

6. The Fractal component of u will be given by $f_i : X \longrightarrow R_g$, $1 \leq i \leq N$, where

$$f(x) = \begin{cases} \phi_i(u(w_i^{-1}(x))), & x \in w_i(X) \\ 0, & x \notin w_i(X) \end{cases}$$

In other words, the fractal component $f_i(x)$ represents a modified value of the gray level of u at the i-th preimage of x (if it exists).

In the following section we present a method described in [BDS]. This method could be used to minimize the function presented in the main algorithm on page 30.

0.4 Penalty functions

A technique for constrained minimization involves applying a penalty to the objective function at non-feasible points, so that an unconstrained minimization technique finds these points unattractive. A typical formulation for inequality constraints

$$c_i(x) \geq 0, \quad i = 1, 2, \ldots, m, \tag{0.1}$$

would be to minimize

$$F(x) = f(x) + \sum k_i [c_i(x)]^2$$

where the summation extends over only those of the m constraints which are violated at x, and the weights k_i are positive. The penalty is therefore seen to be equal to the weighted sum of squares of the amounts by which the constraints are violated, and obviously the greater the violation, the greater the penalty, which for feasible points the objective function is not modified. Some theoretical work has been done giving conditions under which it can be proved that this formulation converges to the constrained minimum as the k_i's, tend to infinity. A scheme for calculating and updating the k_i's is described in Leitmann (1962), p. 213.

The method works reasonably well, although it creates steep valleys at the constraints. In introducing discontinuities into the second derivatives of the

modified objective function F(x), some difficulty might be anticipated when certain minimization methods, particularly gradient methods, are used. A further disadvantage of the method is that the computation of the function at non-feasible points is allowed, which can result in program failures, such as trying to compute the logarithm or square root of a negative number. A somewhat different application of the penalty function concept has proven rather more successful in practice. In this, the search for the minimum is restricted to feasible points, while the modification of the function is negligible or non-existent away from the constraints, but results in a rapid increase in the modified function as the constraints are approached. In this method therefore, the introduced steep valleys lie within the constraints, and all non-feasible function evaluations are avoided. Two methods of this type, due to Rosenbrock (1960) and Carroll (1961), which are known to have performed successfully in many instances, will now be described.

0.4.1 Rosenbrock's Method

Rosenbrock had made provisions for the treatment of constraints of the type

$$l_i \leq x_i \leq u_i, \quad i = 1, 2, \ldots, m, \ (m \geq n)$$

where the implicit variables $x_{n+1}, \ldots, x_m$, if any, are functions of the explicit variable $x_1, \ldots, x_n$ and the l_i and u_i are either constants or also functions of $x_i, \ldots, x_n$.

Rosenbrock introduces boundary zones

$$l_i \leq x_i \leq l_i + 10^{-4}(u_i - l_i)$$

$$u_i \geq x_i \geq u_i - 10^{-4}(u_i - l_i)$$

for $i = 1, 2, \ldots, m$ at the edges of the range of each variable (explicit or implicit). The method requires that an initial point be provided which not only satisfies all the constraints, but also does not lie within any boundary zone. We denote by f_0 the least function value yet found during the search for which the corresponding x_i, $i = 1, 2, \ldots, m$, satisfy their respective constraints, and by f^*, the least value yet found for f_0 for which the x_i, $i = 1, 2, \ldots, m$, have the additional property that they do not lie within any of the m boundary zones. Before commencing the minimization, f_0 and f^* are set equal to the initial function value.

The search for the minimum is performed as for the unconstrained case, except that after each function evaluation, the following steps are carried out.

(i) Set $i = 1$ and $k = 0$.

(ii) If either $f > f_0$, or x_i is non-feasible, the trial is a failure, and we return to the basic f as search procedure.

(iii) If x_i lies within either of its boundary zones, we modify the objective function f as described below and set $k = 1$.

(iv) If $i \neq m$, i is increased by one and we continue from (ii).

(v) If $k = 0$, we set $f^* = f_0$, the old best f value. The trial is a success and we return to the basic search procedure.

The manner in which the objective function is modified within a boundary region is as follows. (A number of different ways by which this might be achieved have been considered by Rosenbrock.) We suppose that x_j lies within its lower boundary zone and compute the fractional depth of penetration of this boundary zone

$$\lambda = \frac{\text{amount by which boundary zone is entered}}{\text{width of boundary zone}} = \frac{l_j + 10^{-4}(u_j - l_j) - x_j}{10^{-4}(u_j - l_j)}.$$

(For upper bound boundary region an analogous formula would be used.)

The computed function value f is now replaced by f', given by

$$f' = f - (f - f^*)(3\lambda - 4\lambda^2 + 2\lambda^3).$$

It is stressed that the computed value of f has been destroyed and the computation continues as if this value had been f', i.e. the function is indeed modified within the boundary zones. The function is further modified if additional boundary regions are entered, in accordance with the algorithm set out above.

For full details of the properties of the boundary regions, the reader is

referred to Rosenbrock (1960). Here we note that the following: (i) at $\lambda = 0$ (i.e. the inner edge of the boundary region), the function value is unaltered, $f' = f$;

(ii) at $\lambda = 1$ (i.e. at the constraint) $f' = f^*$, so that the function value is replaced by the least function value yet obtained for which no boundary zone was entered;

(iii) for a function which decreases steadily as the constraint is approached, the modified function has a minimum in the boundary zone that is, a valley representing an unconstrained path along the constraint has been introduced, and as Rosenbrock shows, this valley slopes back into the interior of the region.

This technique of applying constraints has been found not to be generally effective when applied to other optimization methods.

0.4.2 Carroll's Created Response Surface Technique

In this method, Carroll (1961) proposes that a created response surface

$$P(x,k) = f(x) + k\sum_{i=1}^{m}\frac{w_i}{c_i(x)} \tag{0.2}$$

be defined, where the constraints are assumed to be inequalities of the form $c_i(x) > 0, i = 1,2,\ldots,m$ and k and $w_i, i = 1,2,\ldots,m$, are positive constants. The minimum of this surface is then found and this point is used as a starting point for the minimization of a further response surface corre-

sponding to a reduced value of k. This whole procedure is then repeated for a sequence of response surfaces corresponding to successively smaller values of k and a final minimization is performed with k = 0. In all these minimizations, non-feasible points are excluded. It can be seen that the essence of the method is that a single constrained optimization has been replaced by a sequence of unconstrained ones. The explanation of this apparent anomaly, that an unconstrained minimization procedure is to be used but the search is to be restricted to feasible points, is as follows. Whenever a non-feasible point is proposed, a minimum in the current search direction can always be found by interpolation, so that the unconstrained search procedure will, with this slight guidance, converge to a feasible point.

Fiacco and McCormick (1964) have suggested that the $w_i, i = 1, 2, \ldots, m$, be set to unity, and have proven that, under certain conditions, the minima of this sequence of response surfaces converge to the required constrained minimum. The method has however, been successfully applied to problems for which it is not possible to prove that convergence is assured, for example problems with non-convex feasible regions. The choice of the initial value of k presents some difficulty. If too small a value is used, the created response function P(x, k) will approximate too closely to $f(x)$ before the constrained minimum has been approximately located and in this situation the smoothing property of the method is lost and convergence to each response surface minimum can be very slow. If however, too large a value of k is used, the

minima of the first few response surfaces will be forced well away from all the constraints, and will depend only slightly on the objective function. The usefulness of a good initial approximation to the constrained minimum would therefore, be lost. It is therefore, desirable that the minimum of the first response surface should correspond to a smaller value of the objective function than does the initial Point.

Fiacco and McCormick (1963) have suggested two methods for computing the initial value of k, but as these methods make use of derivatives of the objective function and the penalty term, they are time-consuming, and since they are not completely reliable, [BDS] have preferred to use an empirical approach. [BDS] observe that a necessary condition (certainly for a concave function) for the minimum of the first response function to correspond to a smaller value of the objective function than does the initial point, is that an acute angle exists between the gradient directions of the objective function and the created response function at the initial point. If this condition does not hold, k is too large, while if it does hold, k may be too small. [BDS] suggestion that k be multiplied or divided by ten until this angle changes from acute to obtuse or vice versa and then to use the largest of the k values considered for which this angle is acute. A further rule, which has been found advisable for handling initial points close to constraints, is that a lower limit $k = 0.0001$ be applied. Fiacco and McCormick (1963) have found that the rate at which k is reduced for successive response surfaces has little ef-

fect on the total effort involved in finding the constrained minimum, for the larger the reduction factor, the smaller the number of response surfaces that need be minimized becomes, but the longer each of these minimization takes. [BDS] recommendation is that a constant reduction factor of ten be used for successive k's. [BDS] have found that a larger reduction factor of fifty will give faster convergence, but this sometimes causes a slight loss of accuracy. [BDS] have also found that the convergence of the created response surface technique is both more rapid and more consistent when all the variables and constraints are scaled so as to be of the same order of magnitude, as far as this is possible. In effect, this says that better convergence will be achieved if the w_i are not all set to unity. This opinion has been endorsed privately by McCormick, but no really satisfactory algorithm exists for the generation of the w_i automatically. It can be assumed though that if a constraint is frequently being violated, then the corresponding w_i is too small. An empirical algorithm can be derived from this and much improved convergence has been achieved on process control applications by [BDS] and others. Another observation is that the method is more likely to find the global minimum if the squares of the constraint contributions are used in equation (2). A further opinion which [BDS] have formed is that only constraints which have been violated need be incorporated into the created response function, thereby saving some computing effort.

Finally, [BDS] point out that Fiacco and McCormick (1966) have ex-

tended the method to apply to equality constraints of the form

$$e_j(x_1, x_2, \ldots, x_n) = 0, \text{ for } j = 1, 2, \ldots, s,$$

by adding to equation (equa 1.1) the term

$$k^{\frac{-1}{2}} \sum_{j=1} s e_j^2(x). \tag{0.1}$$

This modification, which Fiacco and McCormick (1966) have proven converges to the constrained minimum under certain conditions, has been found to work satisfactorily, although of course k cannot now be set to zero for the final minimization. When using squared contributions of the inequality constraints, it has been pointed out by W. Murray that the appropriate power of k in (equa 1.2) is $\frac{-1}{3}$.

In the following section we describe an algorithm that can be used to make the triangular partitions of the image into domain and range blocks. This algorithm will construct the first step of the algorithm presented on page 21.

0.5 The Radial Sweep Algorithm

The radial sweep algorithm was first published by [MW82] and later mentioned by [PK90]. The input data has arbitrary distributed points with x, y and z coordinates. The algorithm is described as follows:

1) choose a point near the center of the points, and calculate the distance and bearing from this central point to all the other points. The points have to be sorted and ordered by bearing. The radiating line is established from the center point to all the other points (fig. a)

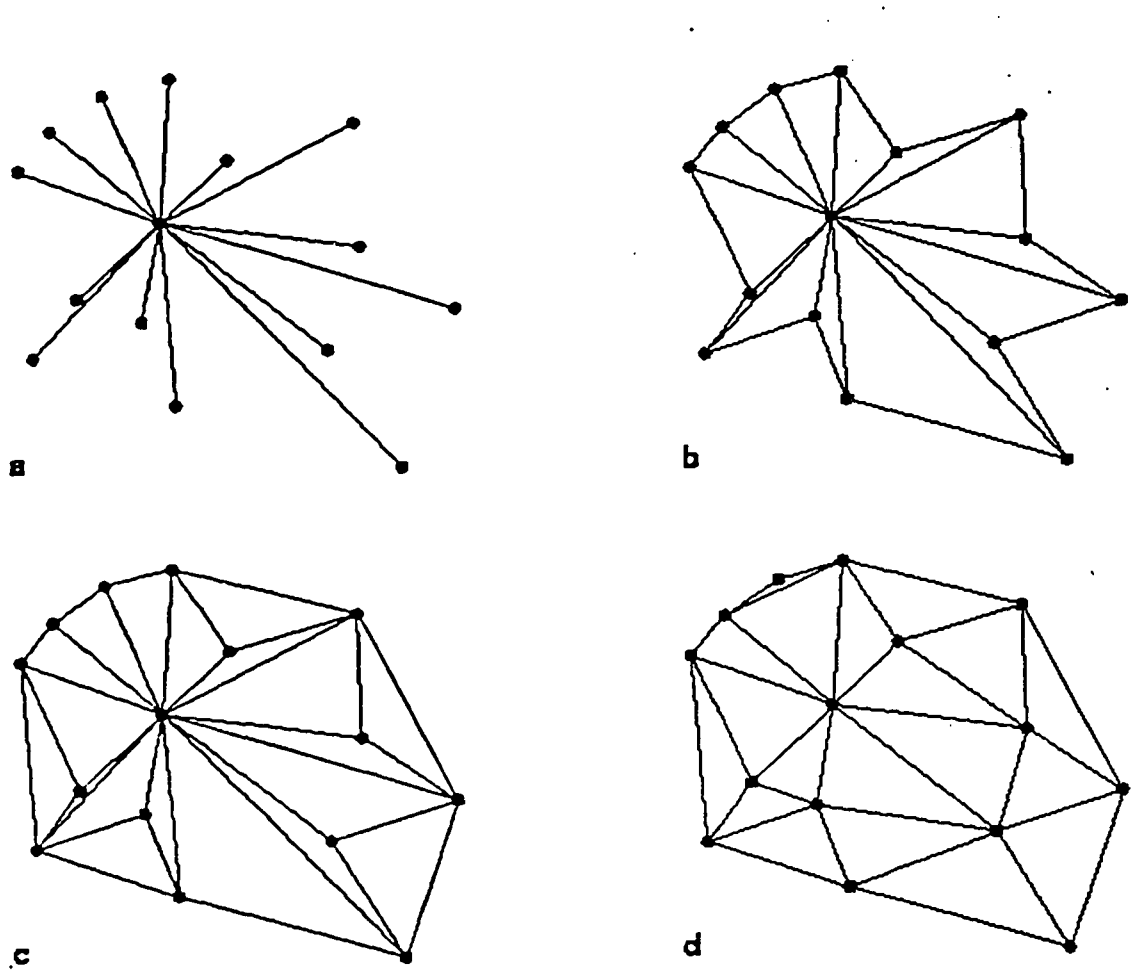
2) Neighboring lines are constructed at the opposite end of the central point. If two points have got identical bearing, their neighboring line will coincide with the radiating line (fig. b).

3) Every point on the boundary is listed in a boundary list. Each of these points is combined with the next two nodes on the list. If these points form a triangle outside the existing network, the triangle is included in the lattice and the boundary list is updated. This procedure goes on until no more triangles can be created. The boundary of the area will now be a convex hull (fig. c).

4) The triangulation from step 3 consists of non-overlapping triangles. However, the shapes of the triangles are less satisfactory (long and thin). In the network, two neighboring triangles make one quadrilateral. The shortest distance between two opposite vertices in the quadrilateral is chosen as the diagonal to improve the geometry. If the "wrong" diagonal is present, this diagonal is swapped to become the shortest diagonal. The diagonal swapping process is repeated until no more alterations appear (fig. d).

Running time: The complexity of the radial sweep algorithm for the sorting of the bearings in the initial step is of order $O(n\log(n))$. Further on the edge swap operations, in worst case, seems to have a running time of $O(n^2)$. But this worst case is extremely unlikely.

Radial sweep algorithm



Radial sweep algorithm. (a) Radiating lines from the central point. (b) The radiating lines are connected. (c) The convex hull is covered by triangles. (d) Final result of the triangulation.

Definition 0.5.1 *The peak signal-to-noise ratio is defined by*

$$\mathbf{PSNR} = 20\log_{10}(\frac{b}{rms})$$

where b is the largest possible value of the signal (typically 255), and rms is the rms difference between two images.

In practice, the PSNR is given in decibel units (dB) which measure the ratio of the peak signal and the difference between two images. An increase of 20 dB corresponds to a ten-fold decrease in the rms difference between two images. There are many versions of signal-to-noise ratios, but the PSNR is very common in image processing, probably because it gives better-sounding numbers than other measures.

The following algorithm will describe a classical encoding technique based on the first iteration.

Algorithm 0.5.1 *The program will perform the following steps:*

1) Use the Radial sweep algorithm to construct three different triangulations of the image support. First and second constructions return the pool P_1 *of domain blocks* D_j*'s . The size of* P_1 *will be increased by considering the six possible rotations and flips of each triangle. The third construction returns the pool* P_2 *of range blocks* R_i*'s (The mean area of any triangle in* P_1 *must be greater or equal than the mean area of any triangle in* P_2*)*

2) Selection of a set of allowed transformations W_{ji}*'s.*
W_{ji} *will be the map that sends the domain block* D_j *onto the range block* R_i

3)Fix the range block R_i *to be encoded,* $i = 1, 2, \ldots N$

4) Fix a tolerance error ϵ

5)From all possible pairs of (D_j, W_{ji}) *select the pair that satisfies the error condition,* $d(W_{j,i}(D_j), R_i) < \epsilon$*, the first.*

6) If step 5 is not satisfied, select the domain D_j *that gives the best match*

then compute $R_i^+ = R_i - D_j$ *and* $R_i^- = D_j - R_i$.

7) Repeat step 5 to R_i^+ *and* R_i^-, *then repeat 6 if it's needed .*

8) Store the parameters that define W_{ji}*'s and their domains. That would be the code of the range block* R_i.

9) Repeat the above steps 4 ,5 , 6 and 7 for all range blocks until all the image is encoded

Note: if step 5 is not satisfied then R_i *will be approximated by:*

$$w_{j,i}(D_j) + w_{j^+,i}(D_j^+) - w_{j^-,i}(D_j^-)$$

Since

$$d(R_i, w_{j,i}(D_j) + w_{j^+,i}(D_j^+) - w_{j^-,i}(D_j^-) \leq d(R_i, w_{j,i}(D_j))$$

0.5.1 Construction of the maps

We start by selecting a range block, say $R_i \subseteq R^3$ with support $K_i \subseteq R^2$, see [MN], let the corner points of K_i be q_{ia}, q_{ib} and q_{ic} where i.e $q_{ia} = (x_{ia}, y_{ia})$.

Let p_{ia} , p_{ib} and p_{ic} be such that i.e $p_{ia} = (q_{ia}, g_{ia}) = ((x_{ia}, y_{ia}), g_{ia})$ where g_{ia} represents the gray level at the position (x_{ia}, y_{ia}).

Now we select a domain block say $D_j \subseteq R^3$ with support $I_j \subseteq R^2$, let the corner points of I_j be q_{jA}, q_{jB} and q_{jC} where $q_{jA} = (x_{jA}, y_{jA})$.

Let p_{jA}, p_{jB} and p_{jC} be such that i.e $p_{jA} = (q_{jA}, g_{jA}) = ((x_{jA}, y_{jA}), g_{jA})$ where g_{jA} represents the gray level at the position (x_{jA}, y_{jA}).

We choose affine maps W_{ji}'s to transform D_j 's into a best approximation of R_i 's

$$A_{ji} = \begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ u_1 & u_2 & u_3 \end{pmatrix}$$

$$T_{ji} = \begin{pmatrix} b_1 \\ b_2 \\ c \end{pmatrix}$$

$W_{ji}(P) = A_{ji} * P + T_{ji}$

Where P is a point in the domain block D_j with coordinates (x_j, y_j, g_j). We need to find the 10 parameters that define the map W_{ji} under the conditions that W_{ji} maps the corner of D_j to the corner of R_i and the support I_j of D_j onto the support K_i of the range block R_i .

0.5.2 Computation of the parameters

For each transformation W_{ji} compute the inverse matrix M^{-1} of the matrix

$$M = \begin{pmatrix} x_{jA} & y_{jA} & 1 \\ x_{jB} & y_{jB} & 1 \\ x_{jC} & y_{jC} & 1 \end{pmatrix}$$

Clearly, M is invertible by construction .

$$\det(M) = \begin{pmatrix} x_{jA} - x_{jC} & y_{jA} - y_{jC} & 0 \\ x_{jB} - x_{jC} & y_{jB} - y_{jC} & 0 \\ x_{jC} & y_{jC} & 1 \end{pmatrix}$$

$\det(M) = 0$ only if the three spatial coordinates live on the same line in R^2.

The vector $V_1 = (a_{11}, a_{12}, b_1)^T$ is computed as follows:

$V_1 = (a_{11}, a_{12}, b_1)^T = M^{-1} * X_{R_i}$

Where the vector X_{R_i} represents the 3 x-coordinates of the corner points of the range block.

$$X_{R_i} = (x_{ia}, x_{ib}, x_{ic})^T$$

The vector $V_2 = (a_{21}, a_{22}, b_2)^T$ is computed as follows:

$V_2 = (a_{21}, a_{22}, b_2)^T = M^{-1} * Y_{R_i}$

Where the vector Y_{R_i} represents the 3 y-coordinates of the corner points of the range block.

$$Y_{R_i} = (y_{ia}, y_{ib}, y_{ic})^T$$

The vector $V_3 = (u_1, u_2, c)^T$ is computed as follows:

$$V_3 = (u_1, u_2, c)^T = M^{-1} * (Z_{R_i} - u_3 * Z_{D_j})$$

Where the vector Z_{R_i} represents the 3 z-coordinates of the corner points of the range block.

$$Z_{R_i} = (g_{ia}, g_{ib}, g_{ic})^T$$

Where the vector Z_{D_j} represents the 3 z-coordinates of the corner points of the domain block.

$$Z_{D_j} = (g_{jA}, g_{jB}, g_{jC})^T$$

0.5.3 Computation of the value of u_3

To compute the value of u_3, we first compute the distance between the range block R_i and the image of the domain block D_j as a function of u_3.

$d(R_i, W_{ji}(D_j)) = (1/n)(\sum_{\forall(x,y)\in K_i}(F(x,y) - G(x,y))^2)^{1/2}$

Where :

*) n is the number of pixel values in the support K_i of the range block R_i.

*) $F(x,y)$ is the grayscale value at the position (x,y) inside the support K_i of R_i.

*) $G(x,y)$ is the grayscale value of the transformed block $W_{ji}(D_j)$ at the position (x,y). W_{ji} must be surjective. G(x,y) will depend on the pixel value at the preimage of (x,y) .

Since W_{ij} is composed of a spatial component function and a gray level component function, we can write :

$W_{ji}((x,y),g) = (S_{ji}(x,y), G_{ji}((x,y,g))$

Where :

$$S_{ji} : I_j \longrightarrow K_i$$

and $S_{ji}(x_d, y_d) = (x_r, y_r)$ here (x_d, y_d) represents the coordinate of a point in the support, I_j of the domain block D_j.

(x_r, y_r) will be in the support, K_i of the range block R_i.

Therefore, $(x_d, y_d) = S_{ji}^{-1}(x_r, y_r)$

Once (x_d, y_d) are found we can compute : $G_{ji} : D_j \longrightarrow R^+$ as

$$G_{ji}(x_d, y_d, g_d) = G_{ji}(S_{ji}^{-1}(x_r, y_r), g_d)$$

According to the general formula of W_{ji}

$$G_{ji}(x_d, y_d, g_d) = u_1 * x_d + u_2 * y_d + u_3 * g_d + c$$

$$G_{ji}(x_d, y_d, g_d) = (x_d, y_d, 1) * (u_1, u_2, c)^T + u_3 * g_d$$

$$G_{ij}(x_d, y_d, g_d) = (x_d, y_d, 1) * V_3 + u_3 * g_d$$

$$G_{ji}(x_d, y_d, g_d) = (x_d, y_d, 1) * M^{-1} * Z_{R_i} + u_3(g_d - (x_d, y_d, 1) * M^{-1} * Z_{D_j})$$

Let

$$f(x_d, y_d) = (x_d, y_d, 1) * M^{-1} * Z_{R_i}$$

Let

$$f'(x_d, y_d, g_d) = (g_d - (x_d, y_d, 1) * M^{-1} * Z_{D_j})$$

So G_{ij} can be written as :

$$G_{ji}(x_d, y_d, g_d) = f(x_d, y_d) + u_3 * f'(x_d, y_d, g_d)$$

If we substitute the value of $G(x, y)$ in the expression of the L_2 distance

$$d(R_i, W_{ji}(D_j)) = (1/n)(\sum_{\forall(x,y)\in K_i} (F(x, y) - G(x, y))^2)^{1/2}$$

$$d(R_i, W_{ji}(D_j)) = (1/n)(\sum_{\forall(x,y)\in K_i} [F(x, y) - (f(x_d, y_d) - u_3.f'((x_d, y_d), g_d))]^2)^{1/2}$$

And we let

$$A_1 = \sum_{\forall(x,y)\in K_i} (F(x, y) - f(x_d, y_d)^2$$

$$A_2 = \sum_{\forall(x,y)\in K_i} (F(x, y) - f(x_d, y_d)).f'((x_d, y_d), g_d)$$

$$A_3 = \sum_{\forall(x,y)\in K_i} (f'((x_d, y_d), g))^2$$

when we differentiate the distance with respect to u_3 the minimum holds at $u_3 = A_2/A_3$. Therefore, we have to find the value of u_3 that minimizes

$$d(R_i, W_{ji}(D_j)) = (1/n).\sqrt{(A_1 * A_3 - A_2^2)/A_3}$$

where i is fixed and $j = 1, 2, \ldots, M$. This computation should be done to each range block in order to encode the total image.

Now we formulate the major algorithm.

A second Iteration Collaging

Let W be a contractive map on some complete metric space X. Therefore, by the fixed point theorem W has a unique fixed point say x_w in X. Furthermore,

$$x_w = \lim_{n \to \infty} W^n(x_0)$$

for any x_0 in X.

Corollary 0.5.1

$$d(x_w, w^2(x_0)) \leq d(x_w, w(x_0))$$

for any x_0 in X.

Proof:

By the fixed point theorem the more we iterate W the closer we get to x_w.

The following algorithm will describe a blockwise construction of a contractive map W on the space of digital images which means the construction of $W = \bigcup w_i$ such that $d(\mu_0, W^2(\mu_0)) < \epsilon$ where μ_0 is the image to be encoded. The difficulty here is to find a PIFS that forms W subject to constraints on W^2. In the following algorithm the construction of W will satisfy the Second Iteration Collaging and as a direct consequence the fidelity will be improved.

Now we represent the main algorithm where we describe a step by step construction of the operator W based on the second iteration.

Algorithm 0.5.2 *The program should perform the following steps.*

1) Partition P_1 of the original image into N blocks of size 8×8 each. Range blocks should not overlap and the total number of the range blocks must cover the image.

2) Partition P_2 of the original image into M domain blocks of size 16×8 each. Every two consecutive domain blocks should intersect in exactly one range block except if both of them are edge domains, in that case their intersection is empty. This constraint on the position of the domain blocks is required in order for the PIFS to be well defined.

3) Create the collection C of all possible domain pairs (there are M^2 pairs).

Now we are ready to start the encoding procedure .

4) Fix a range block, say R_i and a domain block, say D_j.

5)Map in a systematic way pairs of C as follows: Map the first component of the pair onto the left half of D_j and the second component of the pair onto

the right half of D_j then map D_j onto $R_i(j = 1, 2, \ldots M.; i = 1, 2, \ldots N)$.

6) Compute the following:

$$\sum_{i=1}^{i=N/2} a_i\,,\ \sum_{i=N/2+1}^{i=N} a_i\,,\ \sum_{i=1}^{i=N/2} a_i^2\,,\ \sum_{i=N/2+1}^{i=N} a_i^2$$

$$\sum_{i=1}^{i=N/2} b_i\,,\ \sum_{i=N/2+1}^{i=N} b_i\,,\ \sum_{i=1}^{i=N} b_i\,,\ \sum_{i=1}^{i=N} b_i^2$$

$$\sum_{i=1}^{i=N/2} a_i b_i\,,\ \sum_{i=N/2+1}^{i=N} a_i b_i$$

$\sum_{i=1}^{i=N/2} a_i$, *This will be the sum of the average of the pixel values of the first domain block in the pair mapped to the left of D_j.*

$\sum_{i=1}^{i=N/2} a_i$, *This will be the sum of the average of the pixel values of the first domain block in the pair mapped to the left of D_j.*

$\sum_{i=N/2+1}^{i=N} a_i$, *This will be the sum of the average of the pixel values of the second domain block (D_l) of the pair mapped to the right of D_j.*

$\sum_{i=1}^{i=N/2} a_i^2$, *This will be the sum of the square of the average of the pixel values of the first domain block (D_k) of the pair mapped to the left of D_j.*

$\sum_{i=N/2+1}^{i=N} a_i^2$, *This will be the square of the sum of the average of the pixel values of the second domain block* (D_l) *of the pair mapped to to the right of* D_j.

$\sum_{i=1}^{i=N/2} b_i$, *This will be the sum of the pixel values of the first left half of the range block* R_i.

$\sum_{i=N/2+1}^{i=N} b_i$, *This will be the sum of the pixel values of the first right half of the range block* R_i.

$\sum_{i=1}^{i=N} b_i$, *This will be the total sum of the pixel values in the range block* R_i.

$\sum_{i=1}^{i=N} b_i^2$, *This will be the total sum of the square of the pixel values in the range block* R_i.

$$\sum_{i=1}^{i=N/2} a_i b_i \,, \sum_{i=N/2+1}^{i=N} a_i b_i$$

In the last two sums only elements a and b in the same pixel position are summed.

These sums will be used to compute $s_1^k, s_2^k, s_3^k, o_1^k, o_2^k, and\, o_3^k$ *that minimize the formula*

$$\sum_{i=1}^{i=N/2} (s_1^k(s_2^k * ai + o_2^k) + o_1^k - b_i)^2 + \sum_{i=N/2+1}^{i=N} (s_1^k(s_3^k * ai + o_3^k) + o_1^k - b_i)^2$$

Where k is an index that keeps track of the specific range in P_1. The a_i's in the first sum represent the average of the average pixel values of the first domain component of the pair mapped onto the left of D_j and the a_i's in the second sum represent the average of the average pixel values of the second domain component of the pair mapped onto the right of D_j. The b_i's are the pixel values of R_i.

Note: The above formula will be minimized using the penalty function techniques. Other techniques could be used as well. Such as geometric programming or gradient projection methods.

7) If the error is not accepted take the next pair in the collection C, map it onto D_j as in step 5 and do step 6 again.

8) Repeat step 6 until the error is satisfied else do:

9) Keep R_i fixed and change D_j by D_{j+1} and repeat steps 4, 5 and 6 until the error is satisfied else repeat step 8 again.

10) If the error is satisfied store temporarily the six parameters of the three

maps as well as the coordinates of the lower left corner of their domains. Else store temporarily the six parameters and the three lower left corners of the block domains of the maps that give the minimum error.

11) Now we need to check whether the transformation mapping onto the left of D_j and the transformation mapping onto the right of D_j are good candidates to encode the two range blocks formed by D_j.

12) Fix the Range block formed by the left half of D_j and repeat steps 5, 6, 7 and 8 considering that three maps are known. Fix the Range block formed by the right half of D_j and repeat steps 5, 6, 7 and 8 considering that three maps are known.

13) If new maps are generated by step 12 (at least 1, at most 4 for the second search) they will be also stored temporarily and repeat step 12 to the new range blocks involved.

14) If step 8 fails at any level of search (chances of failure are very small) all the temporary storage will be cancelled and we repeat the search starting with different range block. Else

15) If no new maps are generated all the temporary storing will be final and we repeat all the above cycle to new range block until the total image is encoded.

0.6 Analytic Representation of the algorithm:

Formulation of the distance to be minimized. Let

$$d : R^6 \longrightarrow R^+,$$

$$d^2(\mathbf{x}) = \sum_{i=1}^{i=N/2} (s_1^k(s_2^k * ai + o_2^k) + o_1^k - b_i)^2 + \sum_{i=N/2+1}^{i=N} (s_1^k(s_3^k * ai + o_3^k) + o_1^k - b_i)^2$$

where $\mathbf{x} = (s_1^k, s_2^k, s_3^k, o_1^k, o_2^k, o_3^k)^T$. Find $\mathbf{x}$ that minimizes d subject to

$0 \leq s_1^k < 1 \quad 0 \leq o_1^k \leq 255$

$0 \leq s_2^k < 1 \quad 0 \leq o_2^k \leq 255$

$0 \leq s_3^k < 1 \quad 0 \leq o_3^k \leq 255.$

The explicit formula of the error d is

$d^2(R_i, W(D_{ji})) = \sum_{i=N/2+1}^{i=N} a_i^2 s_1^2 s_2^2 + (N/2) s_1^2 o_2^2 + 2 \sum_{i=1}^{i=N/2} a_i s_1^2 s_2 o_2 +$
$2(N/2) o_1^2 - 2 \sum_{i=1}^{i=N} b_i o_1 + 2 \sum_{i=1}^{i=N/2} a_i s_1 s_2 o_1 + 2(N/2) s_1 o_2 o_1 - 2 \sum_{i=1}^{i=N/2} a_i b_i s_1 s_2 -$
$2 \sum_{i=1}^{i=N/2} b_i s_1 o_1 + \sum_{i=N/2+1}^{i=N} a_i s_1^2 s_3^2 + 2(N/2) s_1^2 o_3^2 + 2 \sum_{i=N/2+1}^{i=N} a_i s_1^2 s_3 o_3$
$+ 2 \sum_{i=N/2+1}^{i=N} a_i s_1 s_3 o_1 + 2(N/2) s_1 o_1 o_3 - 2 \sum_{i=N/2+1}^{i=N} a_i b_i s_1 s_3 -$
$2 \sum_{i=N/2+1}^{i=N} b_i s_1 o_3 + \sum_{i=1}^{i=N} b_i^2$

Note: For reasons of simplicity we omitted the index k from the above expression. For example S_1^2 should be $(S_1^k)^2$ and S_1 should be S_1^k.

Construction of the maps:

$$\begin{array}{ccccc} C & \longrightarrow & P_2 & \longrightarrow & P_1 \\ (D_k, D_l) & \overset{w_{k,lj},\, w_{l,Rj}}{\longrightarrow} & D_j & \overset{w_{ji}}{\longrightarrow} & R_i \end{array}$$

where C contains M^2 pairs, P_2 contains M domain blocks and P_1 contains N range blocks. W_{k,l_j} would mean the map that sends the domain block D_k to the left half side of the domain block D_j which is an element of P_1.

W_{l,R_j} would mean the map that sends the domain block D_l to the right half side of the domain block D_j which is an element of P_1.

$$W_{k,l_j}(x_k, y_k, g_k) = \begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & S_k^{lj} \end{pmatrix} \begin{pmatrix} x_k \\ y_k \\ g_k \end{pmatrix} + \begin{pmatrix} a_1 \\ a_2 \\ O_k^{lj} \end{pmatrix} = \begin{pmatrix} \Phi_{k,l_j}(x_k, y_k) \\ G_{k,l_j}(\Phi_{k,l_j}^{-1}(x_{l_j}, y_{l_j}), g_k) \end{pmatrix}$$

$$\Phi_{k,l_j} : I_k \longrightarrow K_{lj}$$

$$G_{k,l_j} : D_k \longrightarrow R_{l_j}$$

where $(x_k, y_k) = \Phi_{k,l_j}^{-1}(x_{l_j}, y_{l_j})$, (x_k, y_k) represents the spatial coordinates of the pixel value g_k in the domain block D_k and (x_{l_j}, y_{l_j}) represents the spatial coordinates of some pixel value g_{l_j} in the left half of D_j.

$$W_{l,R_j}(x_l, y_l, g_l) = \begin{pmatrix} b_{11} & b_{12} & 0 \\ b_{21} & b_{22} & 0 \\ 0 & 0 & S_l^{R_j} \end{pmatrix} \begin{pmatrix} x_l \\ y_l \\ g_l \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \\ O_l^{R_j} \end{pmatrix} = \begin{pmatrix} \Phi_{l,R_j}(x_l, y_l) \\ G_{l,R_j}(\Phi_{l,R_j}^{-1}(x_{R_j}, y_{R_j}), g_l) \end{pmatrix}$$

$$\Phi_{l,R_j} : I_l \longrightarrow K_{Rj}$$

$$G_{l,R_j} : D_l \longrightarrow R_{R_j}$$

where $(x_l, y_l) = \Phi^{-1}_{l,R_j}(x_{R_j}, y_{R_j})$, (x_l, y_l) represents the spatial coordinates of the pixel value g_l in the domain block D_l and (x_{R_j}, y_{R_j}) represents the spatial coordinates of the some pixel value g_{l_j} in the right half of D_j.

$$W_{j,i}(x_j, y_j, g_j) = \begin{pmatrix} c_{11} & c_{12} & 0 \\ c_{21} & c_{22} & 0 \\ 0 & 0 & S^i_j \end{pmatrix} \begin{pmatrix} x_j \\ y_j \\ g_j \end{pmatrix} + \begin{pmatrix} d_1 \\ d_2 \\ O^i_j \end{pmatrix} = \begin{pmatrix} \Phi_{j,i}(x_j, y_j) \\ G_{j,i}(\Phi^{-1}_{j,i}(x_i, y_i), g_j) \end{pmatrix}$$

$$\Phi_{j,i} : I_j \longrightarrow K_i$$

$$G_{j,i} : D_j \longrightarrow R_i$$

where $(x_j, y_j) = \Phi^{-1}_{j,i}(x_i, y_i)$, (x_j, y_j) represents the spatial coordinates of the pixel value g_j where

$$g_j = \begin{cases} S^{l_j}_k g_k + O^{l_j}_k & if\ (x_j, y_j) \in LD_j \\ S^{R_j}_l g_l + O^{R_j}_l & if\ (x_j, y_j) \in RD_j \end{cases}$$

in the domain block D_j and (x_i, y_i) represents the spatial coordinates of some pixel value g_i in the range block R_i where

$$\Phi^{-1}_{j,i}(x_i, y_i) = \begin{cases} \Phi_{k,l_j}(x_k, y_k)) & if\ (x_i, y_i) \in LR_i \\ \Phi_{l,R_j}(x_l, y_l) & if\ (x_i, y_i) \in RR_i \end{cases}$$

Note : All position maps are invertible by construction. LR_i is the left half of the range block R_i and RR_i is the right half of the range block R_i.

In order to put the major algorithm in practical use we need to minimize the following distance by means of penalty functions techniques. But other techniques can be used as well such as gradient projection or geometric programming methods.

$$d = \frac{1}{n}[\sum(F(x_i, y_i, g_i) - G_{ji}(\Phi_{j,i}^{-1}(x_i, y_i), g_j))^2]^{1/2}$$

Where $F(x_i, y_i, g_i)$ will be the pixel value associated to the range block R_i at the position (x_i, y_i) that we wish to encode and n will be the total number of the pixel values in the support of R_i.
Explicitly in this algorithm d will be:

$$d^2(\mathbf{x}) = \sum_{i=1}^{i=N/2} (s_1^k(s_2^k * ai + o_2^k) + o_1^k - b_i)^2 + \sum_{i=N/2+1}^{i=N} (s_1^k(s_3^k * ai + o_3^k) + o_1^k - b_i)^2$$

Where $\mathbf{x} = (s_1^k, s_2^k, s_3^k, o_1^k, o_2^k, o_3^k)^T$

Also d^2 could be written as :

$$d^2 = \frac{1}{n}[\sum(F(x_i, y_i, g_i) - G_{ji}(G_{k,L_j}((x_k, y_k), g_k))^2 + \sum(F(x_i, y_i, g_i) - G_{ji}(G_{l,R_j}((x_l, y_l), g_l))^2]$$

Obviously minimizing this distance is much more complex than minimizing the distance used in classical encoding techniques, but this is the price we pay to get better fidelity.

0.7 Formulation of PIFS Coding/Decoding

In this section we discuss and show an example of a different formulation of PIFS coding and decoding. Also, we will refer to 1-dimensional signals, and we call them either vectors or blocks.

Encoding:

The task of finding the PIFS code of a vector μ_0 is the task of finding a contractive transformation W, such that its fixed point is as close as possible to μ_0. Formally, this task can be described as follows.

Consider the complete metric space $(\mathbf{R}^N, d^\infty)$, where:

1. $\mathbf{R}^N$ denotes the N-dimensional Cartesian product of the real numbers. Each point in $\mathbf{R}^N$ is a column-vector of size N of real numbers. Thus,

$$\mathbf{x} \in \mathbf{R}^N \text{ implies } \mathbf{x} = (x_1, x_2, \ldots, x_N)^T \tag{0.2}$$

2. d^∞ is the metric defined by:

$$\mathbf{x}, \mathbf{y} \in \mathbf{R}^N \quad d^\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1,\ldots,N} |x_i - y_i|. \tag{0.3}$$

The vector to be encoded is $\mu_0 \in \mathbf{R}^N$. We seek a transformation W, such that the following three requirements are fulfilled:

1. W maps the space into itself:

$$W : \mathbf{R}^N \longrightarrow \mathbf{R}^N$$

$$\mathbf{v} \longrightarrow \mathbf{u} = W(\mathbf{v}) \tag{0.4}$$

2. W is a contractive transformation:

$$\exists\ s \in [0,1) |\ \forall\ \mathbf{x},\ \mathbf{y} \in \mathbf{R}^N\ \ d^{\infty}(W(\mathbf{x}, \mathbf{W}(\mathbf{y}))) \leq \mathbf{s}\mathbf{d}^{\infty}(\mathbf{x}, \mathbf{y}). \tag{0.5}$$

These first two requirements define the set of allowed transformations, $W \in \Omega$.

3. Being a contraction in a complete metric space, W has a unique fixed point $\mathbf{f}_w \in \mathbf{R}^N$ such that $f_w = W(f_w)$. The third requirement is that W minimizes the distance between its fixed point $\mathbf{f}_W$ and μ_0:

$$d^{\infty}(\mu_0, f_w) = \min_{v \in \Omega}\ d^{\infty}(\mu_0, \mathbf{f}_v). \tag{0.6}$$

That is,

$$W = \arg\ \min_{v \in \Omega}\ d^{\infty}(\mu_0, \mathbf{f}_v). \tag{0.7}$$

Finding such a transformation W can be a very complex problem, since it involves a minimization over many transformations. An approach to making this problem solvable is to restrict the number of allowed transformations see [JF] and [J89]. The W found this way may be suboptimal, but this is a compromise that one has to make in order to solve the minimization problem.

Thus, we will restrict the allowed transformations W to be systems of M_R functions w_i. Each w_i is further restricted to be of the form:

$$w_i : \mathbf{R}^D \longrightarrow \mathbf{R}^B$$

$$d_{m_i} \longrightarrow r_i = w_i(\mathbf{d}_{m_i}) = a_i\phi(\mathbf{d}_{m_i}) + b_i\mathbf{1}_B, \tag{0.8}$$

where:

d- is a block of D consecutive elements extracted from **v**. It is called the *domain block*. $\mathbf{d}_{m_i}$ is thus the m_i-th domain block in an enumerated list of all such blocks in **v**. The use of the subscript m_i stresses the fact that the domain block $\mathbf{d}_{m_i}$ is mapped to $\mathbf{r}_i$. For now, no specific mechanism for extracting the blocks will be discussed.

r- is a block of $B < D$ consecutive elements of **v**. It is called the *range block*, and $\mathbf{r}_i$ is thus the i-th range block. $\mathbf{r}_i$ belongs to the image of W.

ϕ -is a scalar *scaling* factor,

$$a_i \in \mathbf{R}, \quad |a_i| < 1. \tag{0.9}$$

$\mathbf{b_i}$- is a scalar *offset* value, $b_i \in \mathbf{R}$.

$\mathbf{1}_B$ - is a vector of size B of all 1's.

The three parameters (a_i, b_i, m_i) are called the *transformation parameters.*

By loosely using the union notation to describe concatenation of blocks, we can write:

$$\mathbf{u} = W(\mathbf{v})$$

$$W(\mathbf{v}) = \bigcup_{\mathbf{i=1}}^{\mathbf{M_R}} \mathbf{w_i}(\mathbf{d_{m_i}}), \quad \mathbf{d_{m_i}} \in \mathbf{v}. \tag{0.10}$$

The length of $\mathbf{v}$, which is the result of concatenating M_R range blocks of size B each, is therefore:

$$N = M_R B. \tag{0.11}$$

Moreover, the concatenation of range blocks can also be written (using $\mathbf{r}_i(j)$) as:

$$\mathbf{u}((i-1).B + j) = \mathbf{r}_i(j); \quad i = 1, \ldots, B. \tag{0.12}$$

Now the mechanism of computing $\mathbf{u} = W(\mathbf{v})$, when all the parameters describing W are known, can be described by the following algorithm:

Algorithm a: for Computing the Transformation $\mathbf{u} = W(\mathbf{v})$.

1. For $i = 1$ to M_R:

(a) Extract the $\mathbf{d}_{m_i}$ block from the vector $\mathbf{v}$

(b) Compute

$$\mathbf{r}_i = w_i(\mathbf{d}_{m_i}) = a_i\phi(\mathbf{d}_{m_i}) + b_i\mathbf{1}_B. \tag{0.13}$$

2. *Concatenate* the range blocks thus obtained, $\mathbf{r}_i, i = 1, \ldots, M_R$, in the natural order, to get the new vector $\mathbf{u}$. The length of the vector $\mathbf{u}$ is $N = M_R \times B$.

The transformation W described above is called a blockwise transformation, the reason being evident from the computational algorithm. So far the discussion of the w_i's was quite general. In order to make the discussion both more practical and lucid at this stage, we will make further restrictions and assumptions about the different parameters, shown in step 1 below. The description of the PIFS code of a vector, namely, the parameters that define W, can now be summarized. The PIFS code is described by step 2 below. All other relevant parameters needed for decoding, such as $D = 2B$, $D_h = B$, $N = M_R B$, and others, are derived from the PIFS code using the previous assumptions. The process of encoding, namely, the process of finding the M_R triplets of transformation parameters (a_i, b_i, m_i), is shown in step 3 below. As stated, we seek to minimize $d^\infty(\mu_0, f)$, where μ_0 is the original vector, and $\mathbf{f}$ is the fixed point of the sought transformation. Since, by the Collage Theorem,

$$d^\infty(\mu_0, \mathbf{f}) \leq \frac{1}{1-s} d^\infty(\mu_0, W(\mu_0)), \tag{0.14}$$

one actually tries to minimize the upper bound on the right of Equation

(14), by minimizing $d^{\infty}(\mu_0, W(\mu_0))$ instead of $d^{\infty}(\mu_0, \mathbf{f})$. (Note that since s is the contraction factor of W, the factor $\frac{1}{1-s}$ depends on W. This factor, however, is not taken into account.) Though this method will not necessarily lead to a minimum of $d^{\infty}(\mu_0, \mathbf{f})$, it is at present the most practical way of doing the coding. See [O] section 4.4 for a statistical motivation for the minimization goal. Since W is a blockwise transformation, this minimization can be done in stages, as described below. The minimization process to be described consists of finding a W that satisfies $\mu_0, \cong W(\mu_0)$, Thus, μ_0, is approximately the fixed point of W. Since W uniquely defines its fixed point, storing W (by storing the parameters that define it) defines a lossy code for μ_0. Note that in this case both the operated-on vector and its image by the transformation W are assumed to be μ_0. Therefore, both $\mathbf{d}_{m_i} \in \mu_0$ and $\mathbf{r}_i \in \mu_0$. This formulation will be demonstrated with a numerical example below. By performing the transformation described by the PIFS on μ_0, one can verify that in this example the vector μ_0 is a fixed point of the transformation (namely, $\mu_0 = W(\mu_0)$, and thus the coding in this case is lossless. Indeed, as shown in the example below the result of computing the first code-line, namely, w_1, on μ_0 produces exactly $\mathbf{r}_1$.

Step 1: Parameters, restrictions, and assumptions.

1. N - The size of the vector μ_0 to be encoded is an integer power of 2.

2. $B = 2^l$ - The size of a range block. B is therefore also some integer power of 2.

3. $D = 2B$ - The size of a domain block.

4. $D_h = B$ - The value of D_h is defined to be the shift between consecutive domain blocks. Thus, the number of domain blocks is $N_D \equiv (\frac{N-D}{D_h} + 1)$, and each domain block is given by:

$$\mathbf{d}_{m_i}(j) = \mathbf{v}((m_i - 1)D_h + j), \quad m_i = 1, 2, \ldots, M_D; j = 1, 2, \ldots, D. \quad (0.15)$$

Note that the domain blocks are overlapping, since $D_h < D$.

5. $\phi(.)$ - The spatial contraction function is defined to be:

$$\phi(\mathbf{d}_{m_i})(j) \equiv \frac{1}{2}(\mathbf{d}_{m_i}(2j) + \mathbf{d}_{m_i}(2j-1)), \quad (0.16)$$

for $j = 1, 2, \ldots, B$. That is, $\phi(.)$ contracts blocks of size $D = 2B$ into blocks of size B by averaging pairs of adjacent elements in $\mathbf{d}_{m_i}$.

Step 2: PIFS code.

1. B - The size of the range blocks.
2. M_R - The number of range blocks.
3. M_R triplets of the transformation-parameters (a_i, b_i, m_i).

Step 3: PIFS Coding of μ_0.

1. Store B in the code file.

2. Store M_R in the code file, where $M_R = N/B$, and N is the length of μ_0.

3. Partition μ_0, into M_R range blocks, as described in Equation (12),

$$\mathbf{r}_i(j) = \mu_0((i-1)\ B + j),\ i = 1, \ldots, M_R,\ \ j = 1, \ldots, B. \tag{0.17}$$

4. Extract from μ_0 the $M_D = (\frac{N-D}{D_h} + 1)$ domain blocks, according to Equation (15)

$$\mathbf{d}_l(j) = \mu_0((l-1)D_h + j),\ l = 1, \ldots, M_D,\ \ j = 1, 2, \ldots, D. \tag{0.18}$$

5. For $i = 1$ to M_R

(a) Find the best parameters (a_i, b_i, m_i), such that

$$d^{\infty}(\mathbf{r}_i, a_i\phi(\mathbf{d}_{m_i}) + b_i\mathbf{1}_B) \tag{0.19}$$

is minimized.

(b) Store the parameters (a_i, b_i, m_i) in the code file.

0.8 Decoding

The process of decoding is straightforward, since it involves the finding of a fixed point of a contractive transformation W. This can be done by repeatedly iterating W on any initial vector until a desired proximity to the fixed

point is reached [4]. In the example below, the decoding of the PIFS code is demonstrated, starting from an initial vector of all 0's.

Looking at the example below, an important fact about the decoding (as well as the PIFS definition) can be observed. In the example, the PIFS code, with the prescribed $B = B_1 = 4$, resulted in a transformation $W : \mathbf{R}^{16} \longrightarrow \mathbf{R}^{16}$ with fixed point $\mathbf{f}^1 \in \mathbf{R}^{16}$.

Example:

Let v be the vector that we want to encode.

$v = (23, 21, 17, 19, 11, 9, 15, 13, 5, 7, 3, 1, 15, 13, 9, 11)$

Let the domain blocks be as follows:

$d_1 = (23, 21, 17, 19, 11, 9, 15, 13)$

$d_2 = (11, 9, 15, 13, 5, 7, 3, 1)$

$d_3 = (5, 7, 3, 1, 15, 13, 9, 11)$

Let the range blocks be as follows:

$r_1 = (23, 21, 17, 19)$

$r_2 = (11, 9, 15, 13)$

$r_3 = (5, 7, 3, 1)$

$r_4 = (15, 13, 9, 11)$

We can approximate the r_i's above for $i = 1, 2, 3, 4$ as follows:

$r_1 = 0.5\phi(\mathbf{d}_1) + 12$

$= 0.5\phi(23, 21, 17, 19, 11, 9, 15, 13) + 12$

$= 0.5[22, 18, 10, 14] + 12$

$= [23, 21, 17, 19]$

with the same procedure, we can get:

$r_2 = 0.5\phi(\mathbf{d}_3) + 8$

$r_3 = 0.5\phi(\mathbf{d}_2) + 0$

$r_4 = 0.5\phi(\mathbf{d}_1) + 4$

The decoding by iteration will be given by the following:

$v_0 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

The first iteration: $w(v_0) = (12, 12, 12, 12, 8, 8, 8, 8, 0, 0, 0, 0, 4, 4, 4, 4)$

The second iteration: $w^2(v_0) = (18, 18, 16, 16, 8, 8, 10, 10, 4, 4, 0, 0, 10, 10, 8, 8)$

The Third iteration: $w^3(v_0) = (21, 20, 16, 17, 10, 8, 13, 12, 4, 5, 2, 0, 13, 12, 8, 9)$

where v_0 is the initial vector. Notice that, the third iteration gives us a good approximation of v.

0.9 Compression

Computation of the space needed for storage. The receiver should know the following:

1) A description of the mapping.

2) A description of the position of the domain block for every range block.

Let M be the number of domain blocks. The three corners of the domain block D_j can be mapped onto the three corners of the range block R_i in 6 different ways

The mapping W_{ji} thus consists of a grayscale level Positioning map S_{ji} and a Gray level function G_{ji}. The positioning map is uniquely determined from:

1) The positional information for the domain block.

2. The index $j = 1, 2 \ldots, M$ of the chosen domain block.

3. The orientation of the chosen domain block (6 possibilities).

The tessellation of the image into triangular non-overlapping range blocks, performed using the radial sweep algorithm, has always a fixed set of range triangles. The positional information for the range blocks is thus known. If we assume that all domain blocks have the same chance to be chosen and that all six permutations of the corner points are also equally likely, then the information needed to determine the index and orientation is $\log_2 6M$ bits. The information required to represent the grayscale function G_{ji} can be computed from the following:

The grayscale function G_{ji} is determined from the parameters u_1, u_2, u_3 and c. From the equations above we note that these can be expressed by the parameters u_3, Z_{D_j}, Z_{R_i}. The vector Z_{D_i} contains the grayscale values of the corner points of the domain block. Since the domain blocks are constructed so that two adjacent domain blocks have two corner points in common, the

total number of grayscale values will be $M + 2$. From this set of grayscale values every Z_{D_j} can be produced. The vector Z_{R_i} contains the grayscale values of the corner points of the range blocks. If the total number of range blocks is N and the grayscale values are represented by k bits each, this will yield a total of $k(N + 2 + M + 2)$ bits of information to represent all vectors Z_{D_j} and Z_{R_i}.

Also, we have to represent the mapping coefficients u_3 for each W_{ji}.

Quantization of u_3

$$d^2(R_i, w_{ji}(D_j)) = A_1 + u_3^2 A_3 - 2u_3 A_2$$

$$\frac{\partial d^2}{\partial u_3} = 2u_3 A_3 - 2A_2$$

d^2 is minimal at $u_3 = \frac{A_2}{A_3}$. Therefore $d^2_{\min} = A_1 - \frac{A_2^2}{A_3}$.

Now a little perturbation of d^2 at u_3 yields:

$$d^2(\frac{A_2}{A_3} + \epsilon) = A_1 + (\frac{A_2}{A_3} + \epsilon)^2 A_3 - 2(\frac{A_2}{A_3} + \epsilon)A_2 = d^2_{\min} + \epsilon^2 A_3$$

where $A_3 = \sum_{(x,y)\in k_i}[f'(x_d, y_d, g_d)]^2$.

The maximum error due to quantization of $u_3 = \frac{A_2}{A_3}$ is bounded by $\epsilon \leq \frac{1}{2^{m+1}}$ where m is the number of bits to store u_3. Therefore, the maximum increase of d^2 caused by the quantization of u_3 is bounded by: $\frac{A_3}{2^{2m+2}}$

Suppose that the grayscale values are between 0 and $2^k - 1$, where k would

represent the number of bits to store any pixel value. Therefore, $f'(x_d, y_d, g_d)^2$ will be bounded by 2^{2k}. Now if u_3 is presented by m bits then the variation of d^2 due to the perturbation is bounded above by: $2^{-2(m+1)} * N * 2^{2(k-m-1)}$. Where N is the number of pixels inside the support of the range block. In applications we usually choose $m = k = 8$ which gives an upper bound on the square distortion d^2 of $\frac{N}{4}$. In a typical implementation an error less than 100 is acceptable. Therefore, the construction is numerically stable.

The total amount of bits needed to store the total image would be: $N_T = N * (\log_2 6M + m) + (N + M + 4) * K$ where N is the number of total range blocks, M is The total number of domain blocks, m is the number of bits used to quantize u_3 and K is the number of bits needed to store each pixel value.

For the major Algorithm, a choice of D_j, along with a corresponding S_j and O_j, determines a map $W_{j,i}$. Once we have the collection of all maps, 1024 for a 256×256 image we can decode the image by estimating x_w. The 1024 maps required 4096 bytes compared with the 65,536 bytes needed to store the original image. We get a compression ratio of 16:1.

0.10 Note about Complexity

The worst complexity that we can get in the major algorithm is of order $O(\frac{M^3N}{3})$ to encode the total image. Many techniques of classification can be used to reduce substantially this complexity. One classical technique is to think of blocks as distributions and classify them according to their variance.

0.11 Conclusion

Although fractal image coding is a relatively new technique, it has acquired a performance comparable with other methods such as JPEG or vector quantization. Furthermore, the field of research is far from being exhausted since there are many directions that have not yet been fully investigated (e.g., the use of non-affine transformations, the combination of fractal coding with other techniques and extensions to volume data and video frames). The main advantages of the fractal compression scheme are its ability to provide high compression ratios for a large class of images, the speed of its decoding process and its muli-resolution properties. But the encoding algorithm still suffers from long computation times. One approach is to think of using parallel implementation since range blocks could be encoded independently.

Bibliography

[B] M. F. Barnsley, Fractals Everywhere, Academic Press, New York (1988).

[BD] M. F. Barnsley and S. Demko, Iterated function systems and the global construction of fractals, Proc. Roy. Soc. London **A 399**, 243-275 (1985).

[BDS] M. J. Box, D. DAVIES, W. H. SWANN, Non-linear Optimization Techniques. (1971)

[BH] M. F. Barnsley and L. P. Hurd, Fractal Image Compression, A. K. Peters, Wellesley, Mass. (1993).

[F] Yuval Fisher, Fractal Image Compression, Theory and Application. Springer Verlag, New York, 1995

[F94] Y. Fisher, A discussion of fractal image compression, in Chaos and Fractals, New Frontiers of Science, H.-O. Peitgen, H. Jurgens and D. Saupe, Springer-Verlag (1994).

[H] J. Hutchinson, Fractals and self-similarity, Indiana Univ. J. Math. 30, 713-747 (1981).

[J] A. Jacquin, Image coding based on a fractal theory of iterated contractive image transformations, IEEE Trans. Image Proc. 1 18-30 (1992).

[J89] A. Jacquin. A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding. PhD thesis, Georgia Institute of Technology, August 1989.

[JF] E. W. Jacobs, Y. Fisher, and R. D. Boss. Image compression: A study of the iterated transform method. Signal Processing, 29:251-263, 1992.

[MW82] A. Mirante and N. Weingarten. The radial sweep algorithm for constructing triangulated irregular networks. IEEE Computer Graphics and Applications, 2(3): 11-21, 1982.

[MN] Mirolav Novak. Attractor Coding of Images. Linkoping Studies in Science and Technology. Thesis No. 382, May 1993.

[O] G.E.Oien. L_2-Optimal Attractor Image Coding with Fast Decoder Convergence. Ph.D. thesis, Norwegian Institute of Technology, Trondheim, Norway, June 1993.

[PK90] G. Petrie and T. J. M. Kennie. Terrain Modelling in Surveying and Civil Engineering. Whittles Publishing in association with Thomas Telford Ltd, 1990.