Appendix E The Use of Fractal Theory in a Video Compression System

The paper in this appendix was presented at the IEEE/NASA Data Compression Conference (DCC ‘92), Snowbird, Utah, March 24-27, 1992, its full reference is given below:

Ali, M., Papadopoulos, C., and Clarkson, T. G.,“The Use of Fractal Theory in a Video Compression System”, *Proc. IEEE Data Compression Conference (DCC’92),* 24-27 March, 1992, Snowbird, Utah, eds. James A. Storer and Martin A. Cohn, IEEE Comp. Soc. Press, 1992, p. 259-268.

IEEE Computer Society Press Order Number 2717

Library of Congress Number 92-70381

IEEE Catalog Number 92-TH0436-6

ISBN 0-8186-2717-4

The copy of the Proceedings may be obtained from: IEEE Computer Society Press
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264
U. S. A.

THE USE OF FRACTAL THEORY IN A VIDEO COMPRESSION SYSTEM

Maaruf Ali, Costas Papadopoulos, Trevor Clarkson

Communications Research Group

Department of Electronic & Electrical Engineering

King's College London, Strand, London WC2R 2LS, U.K.

Phone: +44-(0)171-836-5454 Ext.. 3583

Fax: +44-(0)171-836-4781

Email: m.ali@kcl.ac.uk

mali@orion.eee.kcl.ac.uk [Internet]

Abstract

This paper describes how Fractal Coding Theory may be applied to compress video images using an image resampling sequencer (IRS) in a video compression system on a modular image processing system. The first part of the paper describes the background theory of image coding using a form of fractal equation known as Iterated Function System (IFS) codes. The second part deals with the modular image processing system on which to implement these operations. The third part briefly covers how IFS codes may be calculated. Finally, how the IRS and 2nd order geometric transformations may be used to describe inter-frame changes to compress motion video.

Introduction

IFS encoding offers a very high compression ratio (CR) but is computationally intensive and thus requires specialised hardware or a lot of time (in the order of a second) for real-time image processing to be realised. To use IFS encoding's ability of high compressibility of images for video compression, one must either use specialised hardware to speed up the IFS encoding stage or adopt a new coding strategy where less time is spent on the IFS encoding of individual frames. The former method can be expensive for the transfer of technology to everyday applications and thus the second method is pursued. The technique described uses motion estimation to geometrically transform the IFS decoded image of the previous frame to generate the new frame. If no change occurs for a succession of frames then no new data need be transmitted since the image will be held in the framestore and thus theoretically an infinite video CR will be attained momentarily. This technique makes it possible to use IFS encoding in real-time video compression since not all the frames require to be IFS encoded. The two major benefits to be obtained by the use of Fractal Coding are:

i) the attainment of a very high CR - most necessary in today's crowded electromagnetic spectrum,

ii) the resolution independence property of the decoding of an IFS encoded image.

This means that the image may be generated and displayed to any resolution and thus solves the compatibility problems of displaying images for HDTV on conventional TV.

The difficult inverse problem of finding a suitable IFS code whose fractal image is to represent the real image and hence achieve compression is investigated through the use of:

i) a library of IFS codes and **complex moments**,

ii) the method of **simulated annealing**, for solving many parameter non-linear equations.

I) Image Coding Using Fractals

Coastlines, mountains and clouds are not easily described by traditional Euclidean geometry. Objects in nature may be described and mathematically modelled by Mandelbrot's fractal geometry. The two properties of fractals are that:

i) the fractal dimension need not be an integer, it may have fractional values, unlike the Euclidean dimension, from which the word fractal is derived,

ii) the property of self-similarity or scaling - central concept of fractal geometry.

The self-similarity of an objects is the property whereby magnified subsets appear similar or identical to the whole and to each other. It is a characteristic of fractals and sets them apart from Euclidean shapes which generally become smoother. Thus fractal shapes are self-similar and independent of scale or scaling and possess no characteristic size. Euclidean shapes may be described by a simple Algebraic formula whereas fractals are generally constructed using a recursive algorithm suited to computers. [1]

IFS and their use in Image Compression

Using fractals to simulate natural effects is not new. The innovation is to start with an actual image and find the fractals that will imitate it to the required degree of accuracy. Since these fractals are represented in a compact way, the whole image will be represented by a highly compressed data set, thus data compression is achieved.

IFS codes are used to represent the fractal transforms. An IFS is a collection of contractive affine transformations which express relations between parts of an image. The relations are able to define and convey intricate details of a picture. In general, an IFS consists of *m* affine transformations, W_1, W_2, ... W_m, with an associated probability each. The probabilities affect the rate of filling-in of the various regions and attributes of the image. Fractal compression is a lossy compression technique. The high CR may be increased further by applying the best lossless compression algorithm currently available to the IFS codes itself,

Affine Transformations are combinations of rotations, scalings and translations of the co-ordinate axes in *n*-dimensional space. **Figure 1** shows an example of an affine transformation, *W*, operated on a smiling face, *F*, lying on the *xy* plane, moving it to a new face, *W(F)*. *W* always moves points closer together - *W* has to be contractive to satisfy fractal theory.

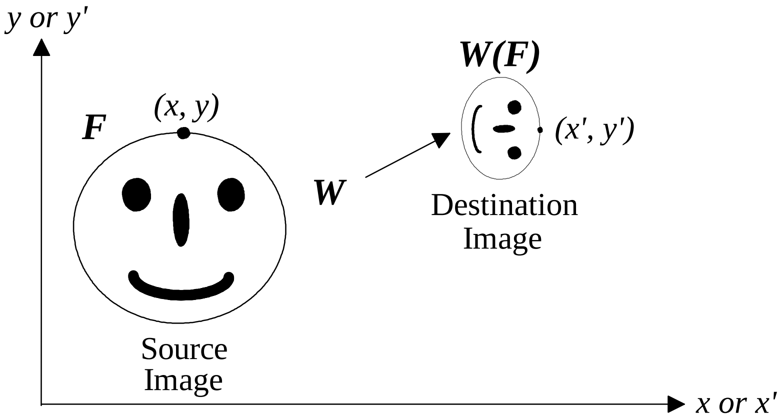


Figure 1. An affine transformation, *W*, operated on a smiling face, *F*. [1]

The general form for an affine transformation is:

$$W\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} ax+by+e \\ cx+dy+f \end{pmatrix}$$

If the translations, rotations, and scalings that make up W are known in advance, then the coefficients may be calculated by:

$$a = r\cos\theta, \quad b = -s\sin\phi, \quad c = r\sin\theta, \quad d = s\cos\phi,$$

where: r = scaling factor on x, s = scaling factor on y

θ = angle of rotation on x, ϕ = angle of rotation on y

e = translation on x, f = translation on y

The affine transformations can be implemented by the hardware that we have developed using the IRS(TMC2301). This chip performs image transformations according to the equations, [2]:

The Geometric Transformation Equations:

$$x(u,v) = \mathrm{A}u^2 + \mathrm{B}u + \mathrm{C}uv + \mathrm{D}v^2 + \mathrm{E}v + \mathrm{F}$$

$$y(u,v) = \mathrm{G}u^2 + \mathrm{H}u + \mathrm{K}uv + \mathrm{L}v^2 + \mathrm{M}v + \mathrm{N}$$

where: the coefficients A to N define the transformation,

x,y = source image co-ordinates, u,v = destination image coordinates.

The 2nd order terms are ignored since affine transformations are only first order transformations, thus the geometric equations simplify to:

$$x = a(x') + b(y') + e, \qquad y = c(x') + d(y') + f$$

where: B = a, E = b, F = e, H = c, M = d, N = f.

Calculation of Transformation Coefficients

There are six unknown coefficients to be solved for the Geometric Transformation equations. Therefore, by selecting six points in the destination image (x', y'), the 'Control Points', and finding their corresponding coordinates in the source image (x, y), two systems of linear simultaneous equations can be set up with six equations in each system and six unknowns. These two systems may be solved by Gaussian Elimination to obtain the required coefficients values. The selection of the six control points for calculating the coefficients is critical because, they must be:

uniquely identifiable - they must be traceable in the source image,

evenly spread in the cell - they must reflect changes all over the cell's area.

If the points are chosen to be on a cell's boundaries, then the advantages are:

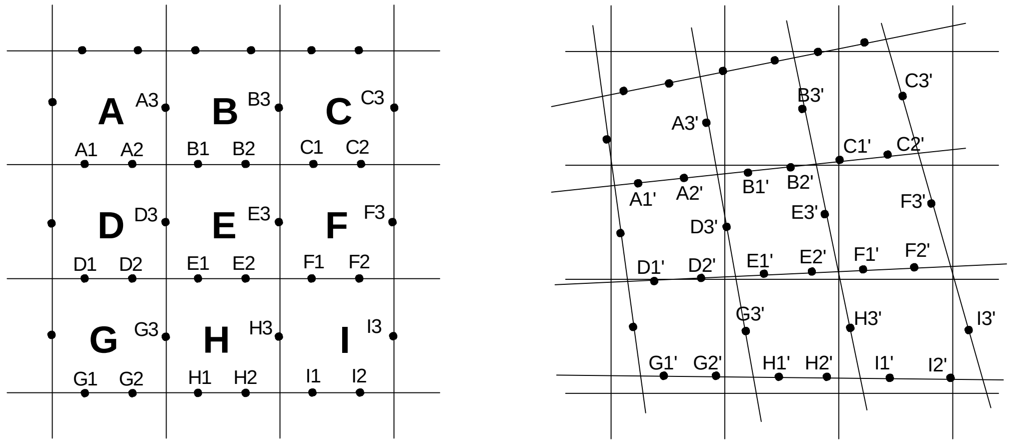
each point can be used for calculating the coefficients of two adjacent cells,

reduction by half the number of points traceable from the destination to source image,

smooth transition transformations from one cell to the next may be obtained.

Figure 2a shows an example of nine adjacent cells in the new (Destination) frame with the six control points for each cell, **Figure 2b** shows how these points may map into the Previous (Source) frame. For each cell three Points are selected, one on the right hand border and two on the bottom border. In order to calculate the coefficients of cell "E", for example, control points "E1, E2, E3, B1, B2, D3" would be utilised.

To enable the edges of objects in the image whose points are most likely to be uniquely identifiable to be found - edge detection operation is performed using convolution with Sobel operators, in order to select the control points. Using this technique, the intercept points in each cell where the edges in the image intercept the right hand border which is nearest to the border's centre, and the two points nearest to the border's "one third intervals" for the bottom border of the cell, are found and selected.



a) SELECTED CONTROL POINTS IN DESTINATION IMAGE

b) MAPPED CONTROL POINTS IN SOURCE IMAGE

Figure 2. Control Points

A motion detection algorithm is then applied to establish where those points were in the source frame, by Search Block Matching using a Mean Absolute Error algorithm.

II) HARDWARE IMPLEMENTATION

To perform the image processing operations a Modular Image Processing System [3] is being developed, whose architecture will offer both high speed processing and hardware flexibility. The system consists of a central processing module responsible for the overall system management, and a number of parallel dedicated hardware modules for performing specific image processing operations. **Figure 3** shows a block diagram of the overall architecture of the system. One of the parallel modules is the IRS used to implement the affine transformations.

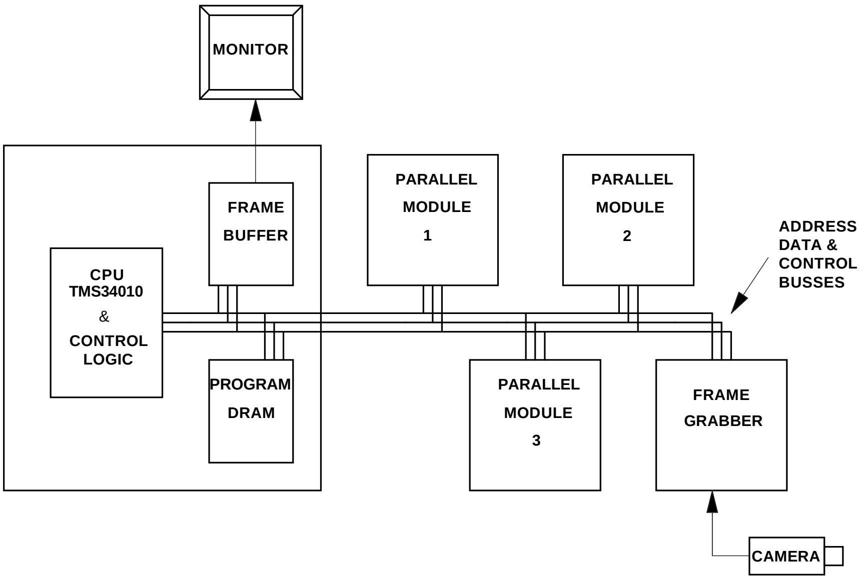


Figure 3. System Outline of a Modular Parallel Digital Image Processor

The system central controller is a TMS34010 Graphics System Processor (GSP) [4] - a graphics orientated microprocessor that offers a rich general purpose and graphics instruction set. The GSP's Address, Data and Control busses expand out of the central processing module to provide an interface for the parallel modules. The Frame Buffer is capable of storing and displaying up to 512x512 pixel images with 8 bits per pixel depth (256 grey levels).

IMAGE Resampling Module performs 1st and 2nd order geometric transformations, bilinear interpolation or convolution filtering using two TMC2301 IRSs as local processors. **Convolver Module** performs convolution in real-time with kernels of up to 8×8 size using the PDSP16488 convolver (Plessey) as the local processor. **IMAGE Compression Module** - The Fractal Compression Module to perform encoding is under development. A Discrete Cosine Transform (DCT) module to compress image data: can perform both Forward and Inverse DCT and also low pass filtering has already been built. It uses the IMS A121 DCT chip as the local processor. A number of **Motion Estimation Modules** performs search block matching using mean absolute error. These modules use STI3220 Motion Estimation Processors as the local processors: used for finding in the source image, the coordinates of the control points, which were selected in the destination image, **IMAGE Difference Module -** calculates mean absolute error between corresponding cells, in the new and previous frames, and also in the new and reconstructed frames. At the transmitter the data compression system will require all the above modules in order to perform all the operations necessary for encoding the sequence. At the receiver, however, much less processing is required for decoding. The receiver will only need to have the Image Resampling Module for performing the geometric transformations according to the parameters that it receives, decoding of IFS code can be achieved in software alone at the receiver, this can be stored as ROM.

III) How to find and decode IFS codes

The complex representation of an IFS code is required to calculate its moment and match it with the moment of the image segment it is to represent. The points (x, y) in the real 2D space may be thought of as points z in the complex plane **C**, then the affine transformation w_i can be expressed in the complex form $w_i(z)$ where:

$$z = x + \mathbf{i}y,$$

$$w_i(z)= c_i\, z + (d_i\, z)^* + b_i,\ i = 1,..., N. \text{ where: } z^* = x - \mathbf{i}y.$$

Comparing this with the polar form of the affine transformation (the probabilities stay the same in either case) the complex variables of the affine transform are found to be:

$c_i^r = ½(r \cos \theta + s \cos \phi)$ $\qquad$ $r^2 = (c_i^r + d_i^r)^2 + (c_i^c - d_i^c)^2$

$c_i^c = ½(r \sin \theta + s \sin \phi)$ $\qquad$ $\tan \theta = (c_i^c - d_i^c)/ (c_i^r + d_i^r)$

$d_i^r = ½(r \cos\theta - s \cos\phi)$ $s^2 = (c_i^r - d_i^r)^2 + (c_i^c + d_i^c)^2$ $b_i^r = e$

$d_i^c = ½(-r \sin\theta + s \sin\phi)$ $\tan\phi = (c_i^c + d_i^c)/(c_i^r - d_i^r)$ $b_i^c = f$

Relationship between Complex Moment and Complex form of IFS

The moment of an IFS is defined[5] by:

$$M_n = \int_k z^n d\mu(z) : n = 0, 1, 2, \ldots$$

where z = points generated by affine transformation w_i.

Consider our fractal image to be made up of m points z_k, then the moment M_n is:

$$M_n = \sum_{k=1}^{m} (z_k)^{\frac{n}{m}}$$

For an IFS with $r = s$ and $\theta = \phi$ (e.g. of the form $w_i(z) = a_i z + b_i$, $i = 1, \ldots N$) then:

$$M_n = \sum_{i=1}^{N} p_i \int_k (a_i z + b_i)^n d\mu(z)$$

M_n can be simplified to:

$$M_n = \left(1 - \sum_{i=1}^{N} p_i a_i^n\right)^{-1} \sum_{i=1}^{N} p_i \sum_{j=0}^{n-1} ({}^nC_j) a_i^j b_i^{n-j} M_j$$

The equation for M_n is obtained given in terms of the previous moments M_j, j = 0, ..., n-1 and the form of the affine transformation a_i, b_i, and p_i, i = 1, ..., N. Since M_0 = 1.0, the rest of the moments may be calculated, without the need for points z_k generation. Thus saving valuable computation time where 10,000 or more points may need to be generated to obtain accurate moments. Thus an IFS that describes an image may be found by attempting to make the moments of the IFS as close to the moments of the image. This only holds for the case $r = s$ and $\theta = \phi$. For the more general case, the general moment definition has to be used:

$$M_{jk} = \int_k z^j z^{*k} d\mu(z) : \quad j, k = 0, 1, 2, \ldots$$

In this case a matrix equation for the moments M_{jk} with $j+k = n$ has to be solved [6], this is in the form of: $- [C] = ([A] - [\boldsymbol{I}]) [M_{array}]$

where: $[M_{array}]$ = vector $(M_{0n}, M_{1(n-1)}, M_{2(n-2)}, ..., M_{n0})^T$,

$[A]$ = matrix, size = $(n+l) \times (n+l)$, elements are IFS parameter dependant,

$[\boldsymbol{I}]$ = Identity matrix, $\boldsymbol{I}_{ii} = 1$, $\boldsymbol{I}_{ij} = 0$ if $i \neq j$,

$[C]$ = IFS parameters and moments M_{jk} (where $j+k < n$) dependant vector.

Thus using $M_{00} = 1$ and the IFS code, the moment M_{jk} can be solved.

1) The moment library search method

The moments have to be normalised so that fractal images that are the same except for a global scaling may be compared. By having a large database of IFS codes and its associated normalised moments, this library may be used to search for an IFS code whose moment is closest to the normalised moment of an image to be encoded. This IFS code may then be retained and passed as the fractal transform of that image segment to use directly to compress that segment or instead the IFS code obtained may be used as a starting point to some non-linear solution method to find a closer IFS code.

2) Newton's Method to find an IFS code close to an image

Newton's method can be used to solve an equation of the form: $f\left(\vec{x}\right) = 0$. The problem is essentially: $f\left(\vec{x}\right) = f_1\left(\vec{x}\right) - f_1\left(\vec{x}_{image\ segment}\right)$

where: $f_1\left(\vec{x}\right)$ = normalised moments function of an IFS code

$f_1\left(\vec{x}_{image\ segment}\right)$ = normalised moments of an image calculated explicitly from the points of the image.

When $f\left(\vec{x}\right) = 0$, then the vector form of the IFS code, $\vec{x}$, has been found.

3) Simulated Annealing method to find an IFS code close to an image [6] [7] [8]

This is a better method of minimising functions of many variables as it will not go immediately to the local minima of a function - a problem inherent with Newton's Method. The problem concerns the thermodynamics of metal cooling and annealing given by the equation:

$$\text{Prob}(E) \approx \exp\left(\frac{-E}{kT}\right)$$

where: E = energy of system, T = temperature (Kelvin), B = Boltzmann's constant.

The method requires parameters that are analogues of T whose value is lowered as the method gets closer to the minima, and energy E, where energy is the value of the system to be minimised. At initial higher T's, changes to higher energy states are much more likely to be accepted and it is this feature of the method that allows the algorithm to find the global minima of a function rather than one of many local minima. The method may be used to find an IFS whose moments are close to a given set of moments. The following are required:

1) Description of system - use vector $\vec{x}$, size N_{ofx} = Number of points in image segment:

$$\vec{x} = \left(c_1^r, c_1^c, d_1^r, d_1^c, b_1^r, b_{1,}^c \ldots, d_{N_{Affines}}^r, d_{N_{Affines}}^c, b_{N_{Affines}}^r, b_{N_{Affines}}^c\right)^T$$

2) A random system change generator - accomplished by random vector $\vec{dx}$ variable of length randomly chosen between 0 and given length δ_l. δ_l and T were lowered simultaneously creating a new vector $\vec{x}_{new}$:

$$\vec{x}_{new} = \vec{x}_{old} + \vec{dx}$$

Check vector for valid IFS code production, if invalid then a new $\vec{dx}$ was generated.

3) The energy of the system, *E*, whose minimisation was required, was substituted by:

$$E = \left| f\left(\vec{x}\right) \right|^2 = \sqrt{\sum_{i=1}^{N_{ofx}} f_i\left(\vec{x}\right)^2}$$

where $f\left(\vec{x}\right) = f_1\left(\vec{x}\right) - f_1\left(\overrightarrow{image}\right)$. Use unnormalised moments for $f_1\left(\vec{x}\right)$ and $f_1\left(\overrightarrow{image}\right)$.

4) The parameter *T* and a method of lowering *T*. *T* governs the changes in the function *E*. Consider *E* value carefully as it affects the energy change, ΔE, that can be acceptable.

How to decode an IFS code - the Random Iteration Algorithm -a summary [6] [7]

1) Initialisation: $x = 0$, $y = 0$.

2) For $n = 1$ to Number_of_points_in_image, do steps (3) and (4).

3) Choose *k* to be one of the numbers 1,2, ..., *m*, with probability P_k.

4) Apply the transformation W_k to the point (x, y) to obtain (x', y').

5) Set (x, y) equal to the new point: $x = x'$, $y = y'$.

6) If $n >$ number_points_required_to_obtain_attractor, then plot (x', y').

7) Loop.

More points are added to an image by increasing the variable Number_of_points_in_image. This may be required to obtain an image with a greater resolution, e.g. for HDTV. Zooming may also be achieved by using an increased scale factor. The variables of (6) is around 100, but may be minimised by empirical methods. The ability of the random iteration algorithm to produce the same image time and time again independent of the random sequence of events chosen has been proven:

1) by carrying out computer-graphical math experiments;

2) by rigorous theoretical foundation of the mathematician John Elton of Georgia Institute of Technology, Georgia, USA.

IV) Motion Video Compression Using Geometric Transformations

The hardware may also be used to compress motion video. The initial frame of the sequence will be sent in its IFS encoded form. The preceding frames may be constructed using geometric transformations to describe inter-frame image changes, whilst simultaneously IFS encoding of a new frame sequence may be occurring. If a frame cannot be sent using geometric transformations then it will be sent as IFS codes. Loss of channel link will not mean loss of image as the image will be stored in a framestore. Infact the framestore may temporarily freeze the last frame whilst the new frame is being encoded as IFS codes.

i) The 1^{st} frame of the sequence is sent in its IFS encoded form, which is regenerated using the Random Iteration Algorithm at the decoder, this can be performed in real-time and entirely in software if necessary.

ii) The new frame to be sent is split-up into rectangular domain blocks, as shown in **Figure 4**. Comparison between the domain block of the new frame and the previous frame is made to determine which cells have altered using an Image Difference operation on the cells and forming the Mean Absolute Error. If the error is below a pre-set threshold then assume no change in the corresponding cell, otherwise, assume that it has changed.

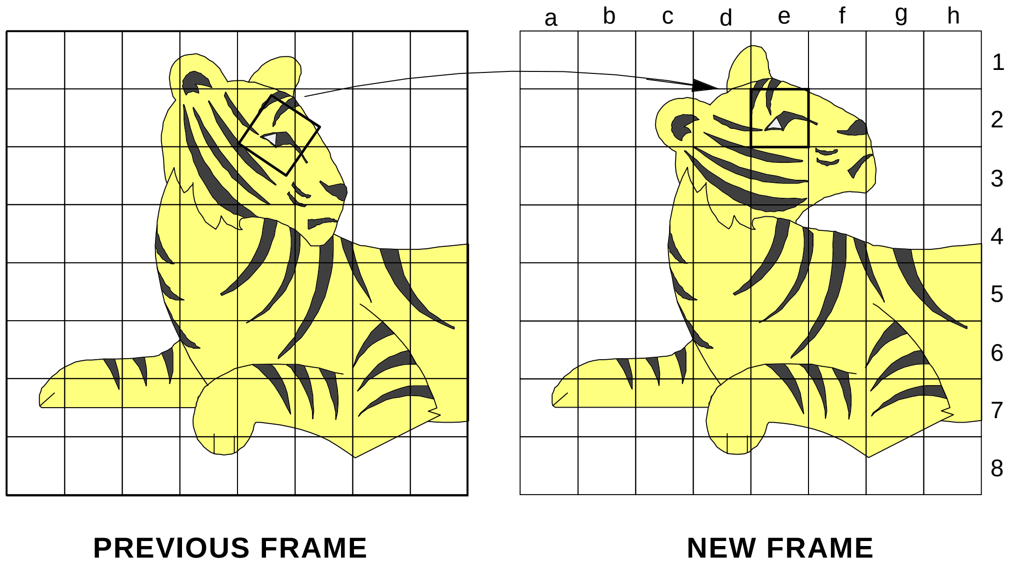


Figure 4 Example of Consecutive Frames in a Sequence

iii) No new information is transmitted for unaltered cells Altered cell are constructed by geometrically transforming parts of the previous frame. For example, cell “e2” in the new frame of Figure 4 can be reconstructed by rotating and translating the highlighted part in the previous frame.

iv) After the reconstruction of each cell, it is compared with the actual cell in the new frame to determine whether the reconstruction was successful or not. The comparison is again performed as in (ii) with an Image Difference operation on the cells, finding Mean Absolute Error. If this reconstruction is successful, then only the transformation coefficients need to be transmitted,

v) otherwise, the cell is sent in its IFS encoded form.

Reasons for Cell Reconstruction Failure

Completely new information may have come into the image such as:

- hidden views of objects which are revealed by 3-dimensional rotations,

- movement of another object which was obscuring the camera’s field of view,

- higher than 2nd order terms being necessary to accurately describe intra-cell changes.

V) COMPRESSION RATIOS

The overall CR will depend on the nature of the frame sequence. In general there will be three categories of cells, the average CR will depend on the percentage of cells that fall into these, cells that have:

i) remained unaltered since the last frame,

ii) altered and can be described by 2nd order geometric transformations,

iii) altered and cannot be described by 2nd order geometric transformations.

Assuming an image of size of 512×512 pixels with 8 bits per pixel depth, and being divided into 64 domain blocks, then the bitmap size of each cell will be 4096 bytes. If this cell falls into the 2nd category above, then it can be described with a set of 16 programmable parameters, which can be used by a pair of IRS devices (TMC2301) to reconstruct the cell. These parameters require 42 bytes of data giving a CR of 97.5:1. The programmable parameters which need to be stored or transmitted are directly related to the Geometric Transformation Equation coefficients. From the Table below, the worst case IFS CR is ~76: 1, which is 3.8 times higher than that obtained by using conventional DCT. The Fractal CR is variable and depends on the image and image quality required. Large compression factors of 10,000: 1 may be obtained for some images.

Cell Category	**Compression Ratios**
Unchanged Cells	Infinite
Cells That Can Be Geometrically Reconstructed	97:1
Cells That Cannot Be Reconstructed - Compressed Using IFS codes	~76: 1 to 10,000: 1
Cells That Cannot Be Reconstructed - Compressed Using DCT	20:1

VI) CONCLUSIONS

An Image Data Compression technique for frame sequences of digital images has been described, which promises CRs well above those offered by conventional Data Compression Methods.

The major advantage of using fractal techniques is that it offers very large image CRs from 76:1 to 10,000:1. The work being conducted using just the geometric transformation module has given CRs of 97.5:1. Using IFS codes - a type of fractal equation - to compress image segments has been explained. The difficult inverse problem of finding a suitable IFS code whose fractal image is to represent the real image and hence achieve compression is being investigated through the use of: a library of IFS codes and complex moment and the method of simulated annealing, for solving non-linear equations of many parameters. The implementation of the fractal encoder is still under development. The application of simulated annealing is being investigated, IFS codes are robust - thus they are ideal for transmission through noisy distortion inducing communication channels - since small deviation of the IFS codes will still produce recognisable images with minimal distortion. The resolution independence of the decoded image from the IFS code makes fractal coding the compression technique for implementing HDTV and ensuring compatibility with non-HDTV equipment.

A Modular Image Processing System has also been presented which can be used to perform the operations required to implement the coding and decoding for this technique. The hardware complexity of the system is concentrated in the coding end of the system which means that it is well suited for applications where a single encoder is used to provide coded information to many decoders, i.e. in broadcasting.

VII) REFERENCES

1. H.-O. Peitgen, D. Saupe, editors, *The Science of Fractal Images*. Springer-Verlag, New York, 1988. Adapted from Fig. 1.4, p.29.

2. J. Eldon R. Wegner, “Using the TMC2301 Image Resampling Sequencer”, *TRW Application Note TP-37*.

3. C. Papadopoulos, T. Clarkson, “Parallel Processing of Digital Images Using a Modular Architecture”. *IEE Proc. 6th Int., Conf. on DSP in Communications*, Sept., 1991.

4. “TMS34010 User’s Guide”, Texas Instruments (1988).

5. M. F. Barnsley, V. Ervin, D. Hardin, J. Lancaster, “Solution of an inverse problem for fractals and other sets”, *Proc. Natl. Acad. Sci*., USA, Vo1. 83, pp. 1975-77, April 1986.

6. D. Wilson, “Fractal Image Compression”, computer science MSc project report, Sept., 1988, Imperial College of Science, Technology & Medicine, University of London.

7. M. F. Barnsley, A. D. Sloan, “A better way to compress images”, *BYTE*, Jan 1988, pp. 215-223.

8. S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, “Optimisation by simulated annealing”, *Science*, Vol. 220, No. 4598, pp. 671-679, 13 May, 1983.