# Iterated Function Systems and Two-Dimensional Shape Representation

**P.A.Giles, A.Purvis**
(Electrical and Electronic Group)
**D. Waugh, R. Garigliano**
(Computer Science Group)

School of Engineering and Applied Science, Durham University,
Science Laboratories, South Road,
Durham, DH1 3LE.

*An application of Iterated Function Systems to the problem of automatic generation of two-dimensional shape representation is presented. A definition of an Iterated Function System is given, and the properties that make it suitable for shape representation are described. Details of programs implementing shape coding and rendering are provided together with an assessment of their performances.*

A fundamental requirement of any vision system designed to perform recognition tasks is a library of representations of objects it is likely to encounter. The classic approach to constructing such representations is to decompose the shape into simple primitive elements. However, for complex shapes it is usually necessary to use a large number of such primitives to achieve an accurate decomposition. The alternative is to define more complex, context-specific primitives, which would be of practical use for only a small set of shapes. The advantage of using an Iterated Function System (IFS) instead is that it enables us to construct a recursive definition of shape. This is achieved by using contractive affine transformations of the original shape as the primitives in its decomposition, and thus removing the need for shape primitives to be defined prior to encoding.

As a first approximation an IFS coding can be thought of as simply a compact list of numbers corresponding to the transform coefficients that describe how a shape maps into itself - in effect how the shape can be covered by a collage of smaller versions of itself. These numbers can then be used by a very simple rendering algorithm to produce a reconstruction of the whole of the original shape. This reconstruction is known as the 'attractor' of the IFS. The rendering process is rapid, stable, and does not depend on any information external to the IFS to enable accurate reconstruction. The nature of the IFS code enables the rendering of a shape with the same speed and accuracy independant of the size and orientation required.

Previous applications of IFS theory to shape encoding [1,2,3], have relied heavily on the interaction of the user with computer programs in order to make the necessary shape collages, and in the case of [3], coding was restricted to shapes that were self-similar and that lend themselves easily to the IFS coding technique. The methods outlined in our paper constitute an entirely automatic way of producing IFS codings.

Section 1 of this paper gives the definition of an IFS and shows how in theory it is possible for one to describe any shape. Section 2 details the programs used in the implementation of IFS theory for both coding and rendering, and section 3 gives preliminary results from their use. Section 4 contains a discussion of the progress made so far and the directions future research needs to take.

## 1. ITERATED FUNCTION SYSTEMS

This section defines what we mean by an Iterated Function System and demonstrates how it can be used to represent a shape. For brevity the following is not mathematically rigorous, but a full derivation starting from first principles can be found in [1].

Definition: A hyperbolic Iterated Function System consists of a complete metric space $(X, d)$, together with a finite set of contraction mappings, $w_n(X)$, with respective contractivity factors $s_n$, for n = 1, 2, ..., N. The notation for an IFS is:

$$\{X : w_n, n = 1, 2, \ldots, N\},$$

and its contractivity factor is:

$$s = max\ \{s_n : n = 1, 2, \ldots, N\}.$$

For applications in two-dimensions the space, $X$, is simply the Euclidean Plane and d, the standard Euclidean metric function. The $w_n$ are just transformations which map points closer together by a factor determined by their contractivities $s_n$, which take values in the range (0,1] .

The important property of an IFS is that if we define the transform $W(A)$ to be:

$$W(A) = w_0\ (A)\ U\ w_1(A)\ U\ w_2(A)\ U\ \ldots\ U\ w_N(A),$$

where $A$ is a bounded region in the plane, then the sequence of regions $A_0, A_1, A_2, \ldots, A_m$ given by iteratively applying W to A such that:

$$A_k = W_k(A) = W(W_{k-1}(A)),\ k = 0, 1, 2, \ldots, m,$$

where $W_0(A) = A$ has a limit, $L$, as m approaches infinity. That is:

$$W(L) = L.$$

This limit region is called the attractor of the IFS.

We say that an IFS represents a given shape if that shape is the attractor of the IFS. Physically then, the representation will consist of a list of transform coefficient values.



AVC 1989 doi:10.5244/C.3.9

## IFS Coding

The problem now is to be able to find the IFS of a given region in the plane. The solution is provided by the Collage Theorem [1,2,4,5], which states that for a given region, $A$:

$$h(L,A) < (1-s)^{-1} h(A,W(A)) \text{ for all } A,$$

where h is the Hausdorf metric function, $L$ the attractor, and s, the contractivity factor of the IFS.

Put simply, the theorem says that as the union of the mappings of the IFS becomes 'closer' to the original region $A$, so the attractor of the IFS becomes closer to $A$ also. Thus, all that we need to do is to make a 'collage' of the given region using contractive mappings of itself.

## Attractor Generation

From the derivation in the previous section it is apparent that given an IFS the corresponding attractor can be generated by iteratively applying the associated set of mappings to an arbitrary starting region. However, this will become an inefficient process when the number of transforms becomes very large or when the convergence to the attractor is slow. An alternative method, described by Barnsley [1,2], requires a modification of the basic IFS in that a probability, $p_n$, is associated with each of the mappings, $w_n$. The notation for the modified IFS is:

$$\{X : (w_n, p_n), n = 1, 2, \ldots, N\}.$$

The attractor of this IFS can now be obtained using the following Random Iteration Algorithm:

1. Take an initial point in the plane, $x_0$. Although the choice of this point can be made arbitarily it is best if it lies somewhere on the attractor.

2. One of the transformations from the IFS is chosen 'at random' with the probability of choosing $w_n$ being $p_n$.

3. The selected transformation is applied to the point $x_0$ to produce the point $x_1$. Another transform is chosen randomly and applied to $x_1$ to produce $x_2$, and so on.

4. The process is repeated for a large number of iterations and the set of points $\{x_0, x_1, \ldots, x_n\}$ that is produced will constitute an approximation of the attractor of the IFS. (It is only an approximation since it contains only a finite number of points).

The values of the probabilities of the transforms can be chosen arbitarily but to ensure an even distribution of points over the attractor the they should be in proportion to the area of the coded region that each covers.

The implementation of the Random Iteration Algorithm is discussed in the next section together with a program that calculates the IFS of a given shape.

## 2. IMPLEMENTATION

The implementation of IFS coding and rendering of the attractor has been attempted using ITEX 150 image processing hardware with a Sun 3/75 workstation running C software under the UNIX operating system. Initial attempts at producing codings have been conducted upon simple shapes in binary images generated by the systems graphics capabilities.

## Automatic Coding

The approach taken has been to concentrate on getting the best fit collage to the boundary of a given region to maximise the accuracy of the shape information. The operation of the coding program can be broken down into the following stages:

1. The boundary points of the given shape are detected using a simple edge following algorithm. An arc-length value is calculated for each point and the centroid of the shape is determined.

2. At each boundary point the value of dr/ds is calculated, where r is the radius value of the point measured from the centroid, and s is the arclength value along the boundary. The values obtained are smoothed using a Gaussian with a sigma value of approximately 1.7.

3. The boundary is segmented into a number of arcs, the endpoints of which correspond to zero crossings of the smoothed dr/ds values.

4. Any arcs of less than five pixels in length are merged with surrounding arcs.

5. The parameterised equations of the arcs are then approximated by fitting to functions quadratic in s using a least squares method. Each arc is then classified as either linear, concave, or convex by inspection of the equation coefficients. At this stage any adjacent straight line segments with similar gradient values are merged. This is necessary to counteract the tendancy of the segmentation process, based on dr/ds values, to break long straight segments at a point perpendicular to the centroid.

6. Using a least squares method contractive transformations which match arcs of the same type onto each other are calculated. Only transformations with contractivity factors less than 0.8 are considered to avoid the value of the $(1 - s)^{-1}$ factor in the Collage Theorem becoming too large. An error function is calculated for each transform generated to test the quality of the match. The error function is:

$$E = E_a [( S_x^2 + S_y^2 )/ 2]^{1/2}$$

where $S_x$ and $S_y$ are the scale factors of the transform in the x and y directions respectively, and $E_a$ is a measure of the overlap between the transform of the shape and the original. All values are normalized to restrict E to the range [0,100].

7. If the best error value obtained for a match to a given arc is less than the set threshold then the transform is accepted as part of the IFS and written to a file. The matched arc is then removed from further consideration. If no suitable match is found then the unmatched arc is halved and the two new segments so produced are placed at the back of the segment queue.

8. The search continues considering each segment in turn until the queue is empty.

## Attractor Rendering

The only difficulty with the implementation of the Random Iteration Algorithm is the choice of probability values. This is best overcome as described in [2], if the transforms are restricted to being affine transformations for which the determinant of the transform matrix is equal to the ratio of the area or the transformed region to the area of the original. Thus, choosing the probabilities for each transform to be in proportion to its determinant will result in an even distribution of points over the attractor. In practice this can give very small values of probability for some transforms so a minimum threshold is set below which probability values are not allowed to fall. A separate program has been developed that reads a file containing an IFS, calculates suitable probabilities for each transform, and produces its attractor at the position, scale, and orientation specified by the user.

By choosing the centroid of a shape as the origin of our co-ordinate system during coding we can guarantee that the point (0,0) lies on the attractor. Taking this as our initial point for the Random Iteration Algorithm means that all subsequent points can be plotted without having to wait for them to converge onto the attractor.

## 3. PROGRAM PERFORMANCE

The IFS coding of simple shapes contained in binary images has been achieved using the programs described in the previous section. Figure 1 depicts the segmentation of the boundary of a simple heart shape based upon the information contained in the dr/ds plot. Attempts have been made throughout the program development to make it as insensitive to noise as possible, especially in this initial segmentation stage. The success of this is shown in figure 2 which depicts the segmentation of the same heart shape after a rotation through 45 degrees resulting in the addition of significant amounts of noise along the boundary. By comparision with the previous figure it can be seen that the two segmentations are almost identical, having the same number and type of boundary segments, but with a slight

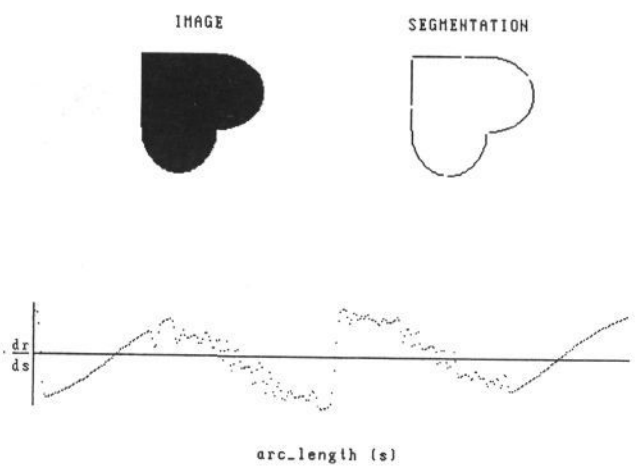



*Figure 1. The segmentation of a heart shape in the absence of noise.*

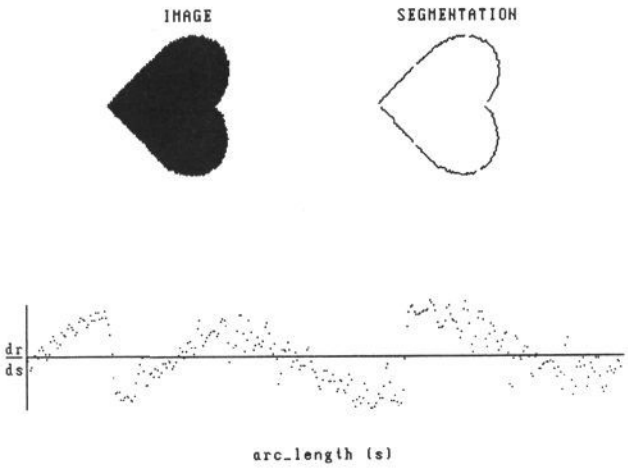



*Figure 2. The segmentation of a heart shape in the presence of noise.*

shift in the positions of the segment endpoints. This is despite the fact that the graph in figure 2 has many more zero crossings than that in figure 1.

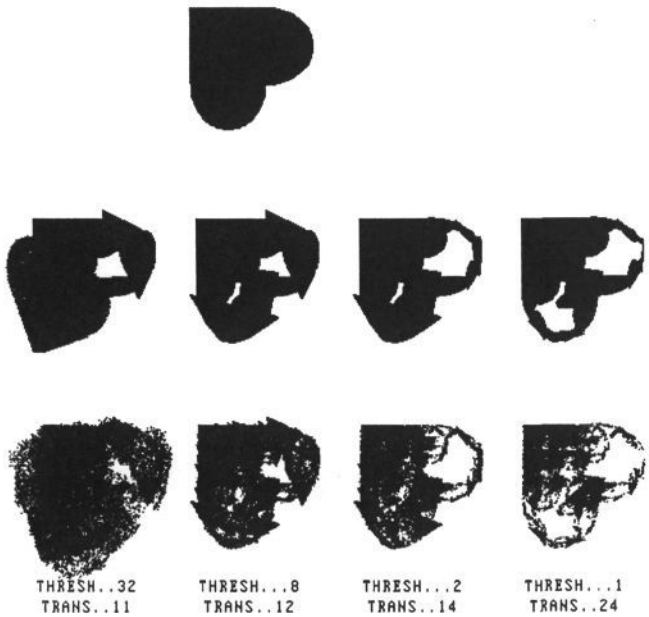
The results of the collage construction are shown in figure 3. The sequences of collages corresponding to decreasing error thresholds show a clear convergence to the original shapes. An effect that becomes apparent at low thresholds is that the transforms appear to migrate towards the boundary leaving large gaps in the interior of the collage. This is due to the transforms becoming very small in order to minimise matching error and as a result they do not extend much into the body of the shape. In themselves the gaps in the collage are not a problem since it is only the shape boundary information we are really interested in.

Figure 3 also shows the output from the attractor rendering program for each of the collages. Again the determinstic relationship between the error threshold and the accuracy of the representation is well demonstrated. However, the effect of leaving gaps in the collage becomes more noticable because of the recursive structure of the attractor which means that gaps get mapped along with each transform resulting in a more diffuse coverage of the attractor than could be expected from examination of the original collage. This too should be acceptable because the collages have no gaps on the boundary and so none should appear in the rendering of the attractor. It can be seen in figure 3c that this is not in fact the case since large sections of the boundary that are clearly visible in the collage have failed to be produced during the rendering process. This is because the missing regions of the boundary have been matched to themselves during collage construction. The practical result of this is that a narrow section of the collage is being constructed from an even narrower section, (a contraction mapping of itself). Upon rendering the attractor becomes vanishingly thin at these points.

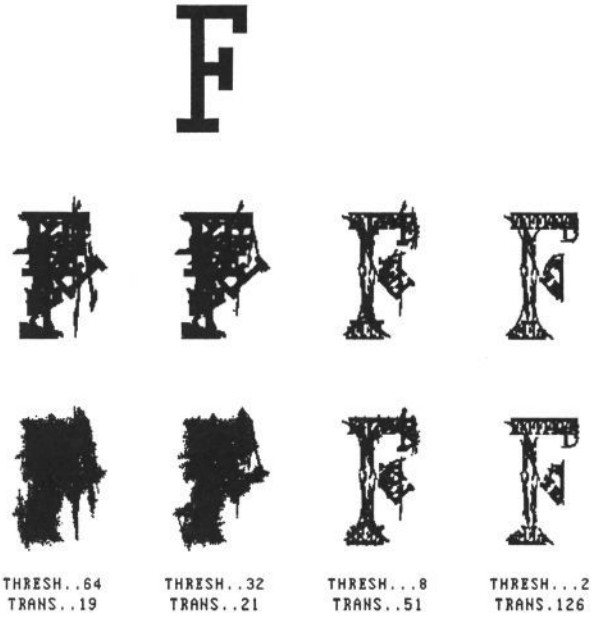
Figure 4 shows the results of collage generation for the heart shape at varying thresholds and in the presence of noise. As demonstrated the segmentation is quite robust so

the differences between these collages and those in figure 3a are due almost entirely to the effect of noise on the calculation of the error measure for each transform. As would be expected more transforms are required to produce a collage of the noisy image. However, the limiting effect of the resolution of the original image is demonstrated by the fact that both the noisy and clean images require a similar number of transforms when the error threshold becomes too small.

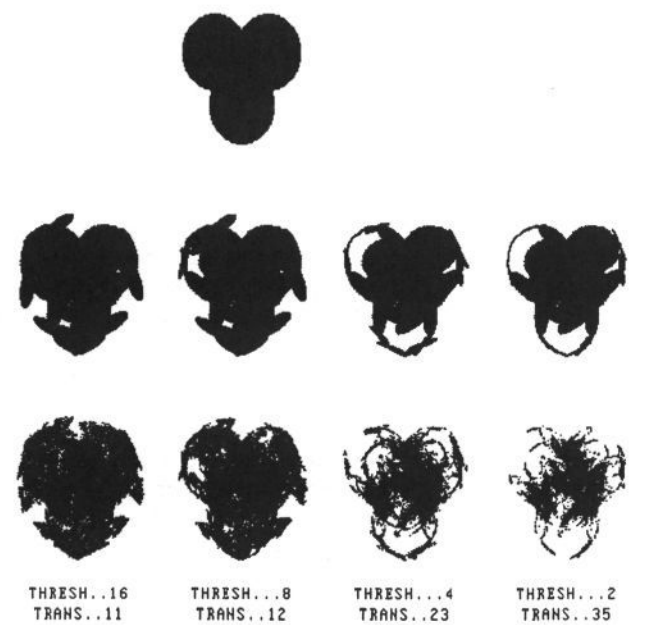
Finally, the time taken for the IFS codes to be produced ranged from 60 to 300 seconds depending on the number of transforms used and the complexity of the shape. No attempts have been made to optimise the speed of the coding program but it is certain that an improvement of an order of magnitude can be achived with the hardware currently being used. Rendering of the attractor was achieved at a rate of 1500 pixels per second and the pictures in the figures contain 50000 pixels each. The speed of the rendering process is only limited by the speed at which floating point calculations can be done.



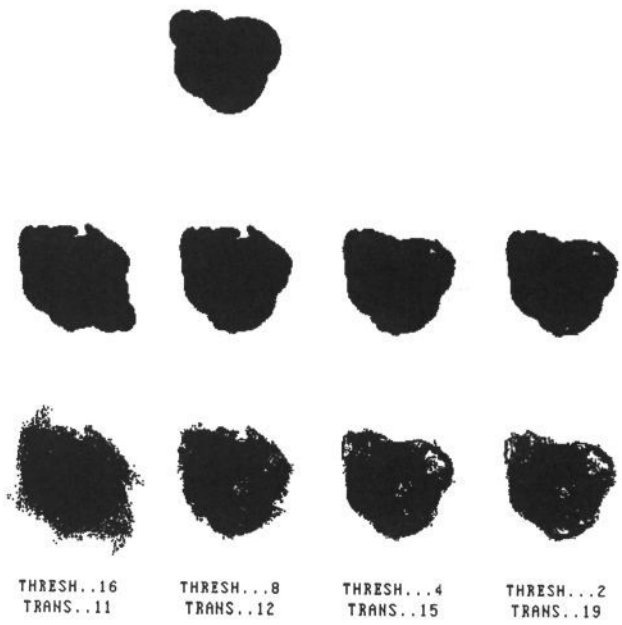

(a)




(b)




(c)




(d)

*Figure 3. Collages and their attractors for the coding of various shapes. In each of the figures a-d the picture on the top row is the original shape. The next row is a sequence of collages and beneath them is a rendering of the corresponding attractor. The error threshold and the number of transforms used in the construction of the collages is given beneath each collage-attractor pair.*

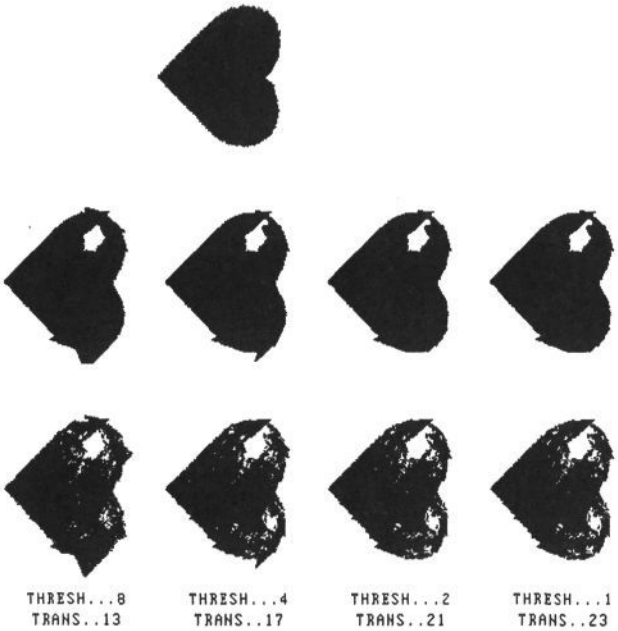



*Figure 4. Collages and attractors for codings of a noisy image*

## 4. CONCLUSION

This paper demonstrates that it is practical to automatically generate IFS representations of two-dimensional shapes to a level of accuracy limited only by the resolution of the original image. The method has been assessed in terms of the speed of code generation, accuracy of shape reconstruction and the ability to cope wih noisy images. The results show that viable applications to machine vision could be possible provided attention is paid to the following areas:

i. To avoid the appearance of gaps in the shape boundary upon the generation of the attractor it is necessary to find collages that completely cover the original shape. There are two approaches that can be made towards the solution of this problem. Firstly, we can simply expand the current technique and look for transformations that cover the interior as well as the boundary. Alternatively, we can follow the example of [3], and add a 'fixed set map' to the IFS - in effect describing a shape as the union of the attractor of the IFS and a fixed region. If the fixed map can be expressed as the combination of ellipses or a similarly simply defined shape then it can be incorporated into the IFS in such a way that the Random Iteration Algorithm can still be used. This second method does however detract from the simplicity of the idea of a completely recursive definition of shape.

ii. As it stands the methods described in this paper are capable of producing reasonably accurate reconstructions of a coded shape with possible applications to object recognition through template matching using the rendering of the attractor. A more powerful way of using the IFS codes would be to be able to match shapes in the 'code space' - by the direct comparison of code coefficients. In order to attempt this the values of the transform coefficients in the IFS must be made invariant with respect to the scale and orientation of the coded shape.

iii. The present search strategy for collage construction is relatively crude and rather slow. Often the calculated IFS contains one or more 'redundant' transforms in that they have been included to match a section of the boundary that has already been adequately covered by other transforms matching different sections. Thus, a more sophisticated search strategy could produce more compact codes and more quickly.

iv. The figures of the previous section show the importance of setting the correct error threshold. If the threshold is too high poor representations are obtained, whilst a too low setting results in extended execution times for relatively little improvement in the representation. Methods for determining the optimum threshold for a given shape are required, or else the development of a completely threshold-independant coding scheme.

Research is currently in progress in all of the above areas.

Acknowledgements
The authors would like to thank M.Brown, J. Anderson, and G.Gapper of the BAe Sowerby Research Centre for useful discussions. P.Giles and D.Waugh acknowledge SERC studentship support, and the former is grateful for BAe CASE sponsorship. This work has been made possible by the generous support of the Durham University Special Equipment Fund.

## 5. REFERENCES

1. **Barnsley, M.** *Fractals Everywhere.* Academic Press (1988).

2. **Barnsley, M. Sloan, A.D.** "A Better Way to Compress Images" *Byte* Jan. (1988 ) pp 215-223.

3. **Libeskind-Hadas, R. Maragos, P.** "Application of Iterated Function Systems and Skeletonization to Synthesis of Fractal Images" *SPIE Vol. 845 Visual Communications and Image Processing II* (1987) pp 276-284.

4. **Barnsley, M. Ervin, E. Hardin, D. Lancaster, J.** "Solution of an Inverse Problem for Fractals and Other Sets" *Proceedings of the National Academy of Science USA* vol.83 Apr. (1985 ) pp 1975-1977.

5. **Barnsley, M. Demko, S.** "Iterated Function Systems and the Global Reconstruction of Fractals" *Proceedings of the Royal Society of London* A399 (1985) pp 243-275.