Fractal Representation of Planar Shapes

Władysław Skarbek*
Frontier Research Program RIKEN, ABS Laboratory,
Hirosawa 2-1, Wakoshi, Saitama 351-01, Japan, email: ska@tora.riken.go.jp

Krystian Ignasiak**
Electronics and Information Technology Department,
Warsaw University of Technology, Poland

Miloud Ghuwar
Institute of Computer Science,
Polish Academy of Sciences, Warsaw, Poland

Abstract. For a collection of planar shapes, (e.g. characters in the given font) a fractal operator is defined which when applied iteratively to a standard set of initial contours, it reconstructs original objects with a high precision. This construction enables fast scalings and rotations of planar individual objects by simple algebraic operations on the fractal representation. Significant compression of data volume was obtained for shape representation.

Key words: Fractals, Image Processing, Shape Modelling.

1 Introduction

Several attempts were made to apply fractal approach to shape modelling. For instance Barnsley and Jacquin [2] defined RIFS (recurrent iterative function system) in which a collection of contours can be reproduced. However, the design process exhibits very high algorithmic complexity.

Ip, Wong, and Mong [5] followed the idea of PIFS (Partitioned Iterative Function System) of Jacquin [6] in order to encode contours of Chinese character.

Their approach, though algorithmically less complex than the RIFS method, has several deficiencies: only scalings by positive integer power of two are admissible, rotations are not possible, and cycles in decoding process can happen.

In this paper we offer the fractal encoding scheme which has none of the above deficiencies: enlargements of any integer scale affine mapping can be used, size reducings are possible for a broad range of scales, rotation by any angle are admissible, and the decoding process takes at most five global iterations to converge to a stable high quality approximation of original shapes. Though

* On leave from Institute of Computer Science, Polish Academy of Sciences.

** The work was supported by the grant no. 8T11C01808 of Polish State Scientific Committee (KBN).

contrary to Ip's, Wong, and Mong method segment fitting is not perfect, possible gaps in decoded contours are enough small to be filled by the linear interpolation without observed deformations.

2 Mathematical foundations

Any set of planar objects drawn from a binary image, can be considered as a collection of their contours (external and internal ones): $\mathcal{C}_1$,...,$\mathcal{C}_k$. Each contour $\mathcal{C}_i$ can be identified with a function

$$\mathcal{C}_i : [0, |\mathcal{C}_i| - 1] \to R^2 \tag{1}$$

defined on a discrete interval of length $|\mathcal{C}_i|$ equal to the number of pixels lying on this contour. By appropriate shifting the function domain, we can represent the whole collection of contours by the single function $X : [0, L-1] \to R^2$, where $L = \sum_{i=1}^{k} |\mathcal{C}_i|$.

We model the codomain of the function X by R^2 instead of Z^2 in order to put our objects into a complete metric space.

However, the points from the original contour must obey *continuity* conditions which impose further restrictions on X:

(1) discrete continuity of contour: in the city norm $\|X(i) - X(i+1)\| \leq 2$ if $i, i+1$ belong to the same domain of a contour.
(2) contour closing, i.e. local periodicity of X: $\|X(f) - X(l)\| \leq 2$ if f, l belong to the same domain of the contour and they are minimum and maximum in this domain, respectively.

Let us denote by $\mathcal{X}'$, the space of all objects X satisfying conditions (1) and (2). For practical reasons instead of the city norm we take the Euclidean norm l_2 to measure the distance between points in the plane. This norm is easily extended to the space $\mathcal{X}'$. In order to provide the convergency for our procedure, we have to close topologically $\mathcal{X}'$ to the space $\mathcal{X}$ in l_2 norm.

In fractal approach for the given contour collection X_0 we are looking for a contractive operator $F : \mathcal{X} \to \mathcal{X}$ which is nearly invariant at X_0. These conditions guarantee that the iterative sequence $Y, F(Y), F^{\circ 2}(Y), \ldots$ converges to a fixed point which is near to X_0. It follows from Banach's fixed point theorem [1] and the collage theorem of Barnsley [3].

In order to build such an operator we try to follow the idea of Jacquin[6] used by him for image compression. However here, we deal with 1D discrete signals which are continuous (in a discrete sense) and locally periodic.

The fractal operator is build from local affine operators, which are based on two interval subdivision schemes:

1. The interval $[0, L-1]$ is covered by target domains (t-Domains) $T \in \Pi$ which are consecutive discrete intervals of length k;

2. The interval $[0, L-1]$ is covered by source domains (s-Domains) $S \in \gamma$ of the length $2k$ which are located within the domain of the given contour with the displacement Δl.

A contour part which is defined on t-Domain (s-Domain) is called *target segment* (*source segment*). In the construction of local operators on $\mathcal{X}$, the important process is matching of target domains to source domains, since each best fitting pair of domains uniquely determines local action of the fractal operator.

Let $X_T = (P_0, \ldots, P_{k-1})$, $T \in \Pi$, $X_S = (Q_0, R_0, \ldots, Q_{k-1}, R_{k-1})$, $S \in \gamma$ be target and source segments to be matched. Firstly, we perform a preprocessing step by the "prime" operation:

$$P'_i \doteq (P_i - P_0)/2 \text{ and } Q'_i = (Q_i - Q_0)/2 \, .$$

Hence instead of X_T and X_S we further analyze segments:

$$X'_T = (P'_0, \ldots, P'_{k-1}) \text{ and } X'_S = (Q'_0, \ldots, Q'_{k-1}) \, .$$

By the *matching error* $\epsilon(T, S)$ we mean the minimum total square error over all linear transformations M which approximate X'_T by $M(X'_S)$ in this way that $M(Q'_{k-1}) = P'_{k-1}$ and the least square error among other corresponding points is minimum.

Note, that the prime operation effectively normalizes two domains in order to have their first points in the beginning of the coordinate system. This removes two free parameters from affine transformations M, transferring them into pure linear local mappings defined by only four parameters.

The formula for $\epsilon(T, S)$ is as follows:

$$\epsilon(T, S) = \min_M \sum_{i=1}^{k-2} \| P'_i - M(Q'_i) \|^2 \tag{2}$$

By $\mu(T) \in \gamma$, we denote the s-Domain which gives the best fit for $T \in \Pi$ i.e.

$$\epsilon(T, \mu(T)) = \min_{S \in \gamma} \epsilon(T, S) \quad \text{and} \quad \mu(T) = \arg\min_{S \in \gamma} \epsilon(T, S).$$

By M_T, we denote the best affine mapping of $X_{\mu(T)}$ onto X_T.

Having $\mu(T)$ we can define the fractal operator by the following formula:

$$F(Y) = \sum_{T \in \Pi} M_T(Y'_{\mu(T)}) \text{ for } Y \in \mathcal{X} \tag{3}$$

Algorithmically, F acts on any $Y \in \mathcal{X}$ taking consecutively t-Domains $T \in \Pi$ and mapping the current X bounded to s-Domain $\mu(T)$ by evaluating space points $M_T(Q'_i), i = 0, \ldots, k-1$. At this process we do not need to know the displacement vector between initial points of mapped segments. The knowledge of domain identifying codes is enough to localize both segments. In practice any sequential numbering of domains is proper.

3 Searching for the best fit affine mapping

Let $P'_i = [p^i_x, p^i_y]^T$, $Q'_i = [q^i_x, q^i_y]^T$ be normalized points. We search for the local mapping M in the form:

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{4}$$

Assuming that $q_y^{(k-1)} \neq 0$, from the constrain $M(Q'_{k-1}) = P'_{k-1}$ we get conditions on elements b, d of the matrix M:

$$b = \alpha_b + \beta_b a, \; d = \alpha_d + \beta_d c, \; \beta_d = \beta_b \tag{5}$$

where:

$$\alpha_b = \frac{p_x^{(k-1)}}{q_y^{(k-1)}}, \; \beta_b = -\frac{q_x^{(k-1)}}{q_y^{(k-1)}}, \; \alpha_d = \frac{p_y^{(k-1)}}{q_y^{(k-1)}} \tag{6}$$

Then the least square method leads to the following solution for nonlinear segments defined on t-Domains:

$$a = \frac{\sum_{i=1}^{k-2} \left(q_x^{(i)} + \beta_b q_y^{(i)}\right) \left(p_x^{(i)} - \alpha_b q_y^{(i)}\right)}{\sum_{i=1}^{k-2} \left(q_x^{(i)} + \beta_b q_y^{(i)}\right)^2} \tag{7}$$

$$c = \frac{\sum_{i=1}^{k-2} \left(q_x^{(i)} + \beta_d q_y^{(i)}\right) \left(p_y^{(i)} - \alpha_d q_y^{(i)}\right)}{\sum_{i=1}^{k-2} \left(q_x^{(i)} + \beta_d q_y^{(i)}\right)^2} \tag{8}$$

Note that for nonlinear t-Domains, i.e. such that its points are not colinear, we have the following nonlinearity condition:

$$\sum_{i=1}^{k-2} \left(q_x^{(i)} + \beta_d q_y^{(i)}\right)^2 \neq 0 \tag{9}$$

This condition ensures the correctness of the above formulas (7,8).

When $q_x^{(k-1)} \neq 0$ we derive the symmetrical formulas for b, d coefficients ($\beta_c = \beta_a$):

$$\alpha_a = \frac{p_x^{(k-1)}}{q_x^{(k-1)}}, \; \beta_a = -\frac{q_y^{(k-1)}}{q_x^{(k-1)}}, \; \alpha_c = \frac{p_y^{(k-1)}}{q_x^{(k-1)}} \tag{10}$$

$$b = \frac{\sum_{i=1}^{k-2} \left(q_y^{(i)} + \beta_a q_x^{(i)}\right) \left(p_x^{(i)} - \alpha_a q_x^{(i)}\right)}{\sum_{i=1}^{k-2} \left(q_y^{(i)} + \beta_a q_x^{(i)}\right)^2} \tag{11}$$

$$d = \frac{\sum_{i=1}^{k-2} \left(q_y^{(i)} + \beta_c q_x^{(i)}\right) \left(p_y^{(i)} - \alpha_c q_x^{(i)}\right)}{\sum_{i=1}^{k-2} \left(q_y^{(i)} + \beta_a q_x^{(i)}\right)^2} \tag{12}$$

Using the above formulas for each s-Domain S which is outside the contour including t-Domain T we find the best s-Domain $\mu(T)$ minimizing the error $\epsilon(T, S)$.

Note that in case when both $q_x^{(k-1)} \neq 0$ and $q_y^{(k-1)} \neq 0$ then the nonlinearity condition (9) is equivalent to its dual counterpart:

$$\sum_{i=1}^{k-2} \left(q_y^{(i)} + \beta_a q_x^{(i)} \right)^2 \neq 0 \tag{13}$$

4 Algorithms

In this section, we include the pseudocode for encoding and decoding algorithms and discuss changes of shape parameters at its rotation, scaling, and translation.

1. **Encoding algorithm**: The encoding process begins from partitioning all the contours into target segments of length k and covering the contours by source segments of length $2k$. As a matter of fact, only pointers to segment beginnings in contour representation, are created.
   The last target segment of each contour requires a special service:
   (a) if it has the length $k_l < k/2$ it is joined to the previous segment so that the last segment is of length $k_l := k + k_l$;
   (b) if it has the length $k > k_l \geq k/2$ it is left without any change.
   Having the last segment of length k_l different than k we have to provide source segments of length $2k_l$ to manage that.

```
for all target segments P_i
   if target segment is linear
      store its end points
   else
      for all non-linear source segments Q_j
         compute affine mapping,
            i.e. a,b,c,d (formulas (7,8),(11,12))
         compute error of matching, formula (2)
         if the error is smaller than the previous one
            store current error and segment parameters
      endfor
      store number of source segment and affine mapping
endfor
```

2. **Decoding algorithm**: One iteration of decoding process has the pseudocode:

```
for all contours
   make starting base in the contour (square)
   for all target segments in the contour
```

```
        apply affine mapping to the proper source segment
        store result as a new target segment for the next iteration
    endfor
endfor
```

3. **Translation of shape**: In our method translations are performed on pixel representation of shapes, i.e. after the reconstruction of rotated and scaled shapes. Incorporating them into parameters of local operators would result in increase of the reconstruction time.
4. **Rotation of shape**: To rotate a shape by the angle α we have to modify the following parameters of local operator:
   - the starting point (x, y) of each segment and the ending point of each linear segment:
   $$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} ;$$
   - the matrix M_T of the local mapping:
   $$M_T' \doteq \begin{bmatrix} a_r & b_r \\ c_r & d_r \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} .$$
5. **Scaling of shape**: To scale a shape by the factor s the folowing parameters are multiplied by s:
   - the total length of each contour;
   - the starting point of each segment and ending points for linear segments;
   - the length k of t-Domains;
   - the displacement Δl between s-Domains.

   The discrete nature of t-Domains is the reason of some restrictions on reducing factors. After the scaling each t-Domain should have integer number of points, so any scaling factor s for which $s \cdot k$ is integer is proper. For instance reducing of the shape size at $k = 16$, can be obtained at 15 possible scales from the set $S_r(16)$ where $S_r(k)$ is defined as follows:
   $$S_r(k) \doteq \left\{ s = \frac{i}{k} \mid i = 1, 2, \ldots, k-1 \right\} \tag{14}$$
   The possible enlargements are obtained for scalings from the set $S_e(k)$ which can be defined as the set algebraic sum of positive integers N^+ with the set $S_r(k)$: $S_e(k) \doteq N^+ + S_r(k)$.

5 Experimental results

For most experiments we have used shapes from the training set shown in Figure 1(a). For experiments requiring big training set the shapes from the Arabic alphabet (cf. [4, 8]) were chosen. The Arabic alphabet is composed of about 60 different shapes. Many of them have secondary components. Generally, a shape complexity is higher for the Arabic characters than for the Latin ones.

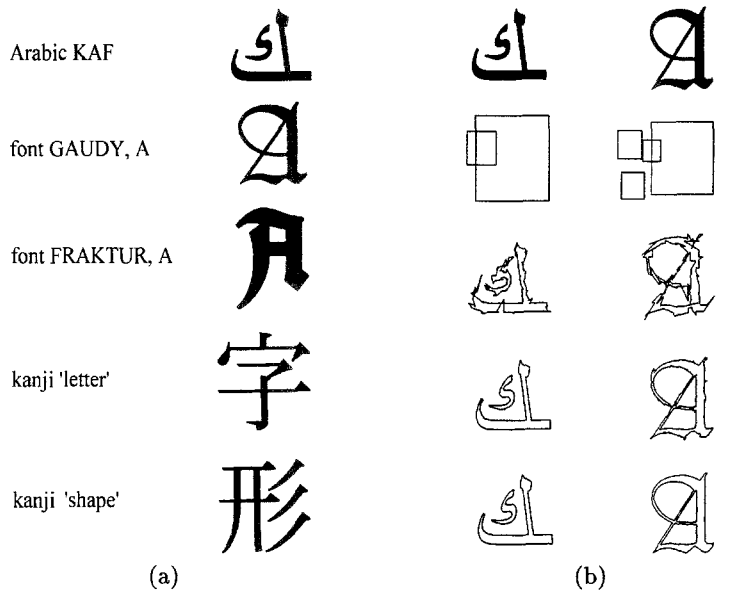


Fig.1. (a) A particular training set of symbols. (b) Original symbol, starting base, the first, the second and the fifth iteration of the fractal operator for Arabic KAF and Gaudy A.

The symbols were digitized manually using a handy scanner at the resolution 400dpi. The typical size of obtained images, was 200 by 200 binary pixels.

In the preprocessing step the binary images of characters, i.e. the bitmaps were automatically converted into a collection of contours.

In the encoding stage the fractal operator is found for many settings of the following parameters: k – t-Domain length, Δl – displacement between s-Domains, t – linearity threshold.

If the given segment has the nonlinearity measure (formula (9)) below the threshold t, it is considered as the linear segment and it is represented by its endpoints.

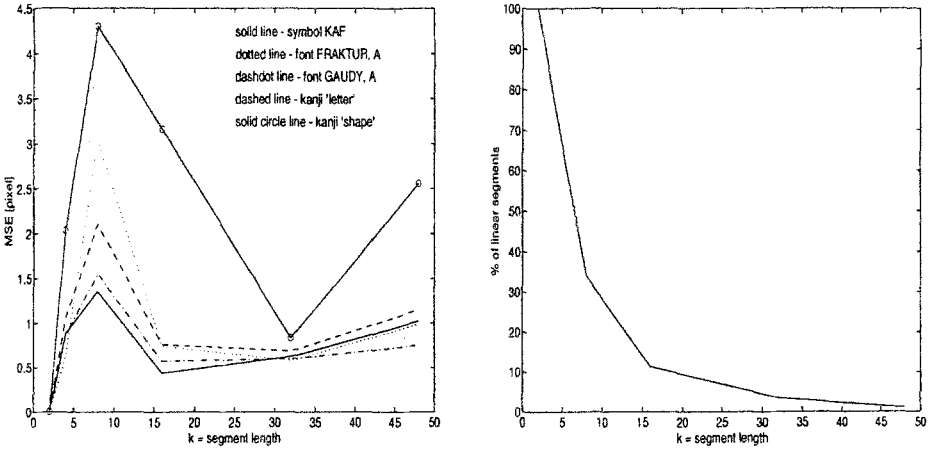
The decoding stage is illustrated in Figure 1(b). The main parameter of this step is n – the number of iterations for the fractal operator. We see that five iterations are enough to reconstruct the original shape.

The initial collection of contours is always chosen as a set of square contours with the number of pixels equal to the number of pixels in original contours. The position of each square contour is aligned to the top leftmost point of the original contour. Though theoretically we could choose any closed contours with the given number of points, the most convenient appeared contours of squared shapes.

In the implementation all the contours are kept in separate data arrays. In such a representation the local mapping to the given t-Domain T was made by choosing the every second position in the index range of the corresponding s-

Domain $\mu(T)$, shifting the coordinates to the initial point of T, dividing them by two and finally mapping them linearly using the matrix M_T.

As it was expected the Euclidean norm cannot ensure the discrete continuity of the fractal attractor we have built for shapes. It appears sometimes that gaps of one or two pixels width arise. In practice, filling them with a discrete straight line does not introduce any visible distortions.



(a) (b)

Fig.2. (a) MSE as a function of segment's length k. (b) Percentage of linear segments in all shapes as a function of segment's length k ($\Delta l = 32,\ t = 0.5$).

The quality of the given contour reconstruction can be measured by a mean squared error defined by differences (δ_x, δ_y) between the original points and the reconstructed points:

$$MSE(\mathcal{C}, \mathcal{C}') \doteq \frac{1}{|\mathcal{C}|} \sum_{i=0}^{|\mathcal{C}|-1} (\delta_x^2 + \delta_y^2) \tag{15}$$

The Table 1 illustrates the dependence of the reconstruction error on the number of iterations. The unit of measurement for MSE is the pixel. We see that the reconstruction error is already stabilized at three or four fractal operator iterations. To ensure full stability we take five iterations as the standard for chosen shapes.

The Figure 2(a) shows an interesting fact: the error of approximation is not a monotonical function of segment's length. This strange bahaviour can be explained by the following analysis based on Figure 2(b):

1. for $k = 2$, all the segments are linear and they are very short, so the error of approximation is near zero;

n	1	2	3	4	5
Arabic KAF	24.37	2.12	0.103	0.072	0.072
font GAUDY, A	13.84	0.82	0.102	0.094	0.094
font FRAKTUR, A	6.52	0.54	0.095	0.095	0.095
kanji 'letter'	10.80	0.67	0.074	0.074	0.074
kanji 'shape'	6.94	0.50	0.111	0.105	0.105

Table 1. Dependence of MSE (in pixels) on the number of iterations n for chosen characters ($k = 32$, $\Delta l = 16$, $t = 0.5$).

2. for $k = 4$ most segments are linear, but they are longer, so they introduce a higher error;
3. for $k = 8$ most segments are treated as nonlinear, but longer now linear segments introduce big error;
4. for $k = 16$, and $k = 32$ most segments are nonlinear and they approximate well, so the error is small;
5. for $k > 32$ nonlinear segments become longer and they start to introduce bigger error.

Such behavior of the error is a tradeoff between approximation features of linear and nonlinear segments.

The linear segments fit for very short k, the nonlinear ones fit for longer segments, when they can demonstrate their power.

Obviously, if we take the bigger t-Domains, we get the higher compression but the reconstruction error is higher too. The Table 2 illustrates this dependence. The training set was consisting of 20 randomly chosen Arabic characters. We see that up to the length $k = 32$ the level of the error is acceptable.

k	4	8	16	32	48
SIN	0.13	0.15	0.12	0.39	1.26
SHIN	0.13	0.15	0.18	0.46	1.69
AYN	0.07	0.08	0.13	0.38	0.64
FA	0.15	0.15	0.33	0.44	0.61
KAF	0.06	0.06	0.11	0.21	0.61
HA	0.06	0.05	0.10	0.23	0.28

Table 2. The error MSE (in pixels) as the function of t-Domain length k for several Arabic characters ($n = 5$, $\Delta l = 8$, $t = 0.5$).

The essential parameter for the quality of obtained characters is the nonlinearity threshold t. How it affects the MSE quality measure, we can conclude

from the right part of the Table 3. We see that up to the level $t = 1.0$, the error is still acceptable.

t	0.1	0.5	1.0	5.0	10.0	50
SIN	0.13	0.12	0.12	0.51	1.00	4.41
SHIN	0.18	0.18	0.17	0.21	0.23	0.30
AYN	0.13	0.18	0.17	0.21	0.23	0.30
FA	0.33	0.33	0.33	0.22	2.50	9.62
KAF	0.12	0.11	0.10	0.15	0.16	2.40
HA	0.10	0.10	0.10	0.68	1.80	3.70

Table 3. The error MSE (in pixels) as a function of nonlinearity threshold t ($k = 16$, $\Delta l = 8$, $n = 5$).

Fig.3. Examples of reducing and magnifying operations ($k = 32$, $\Delta l = 16$, scale $0.25 \div 1.5$ step 0.25).

Some examples of reducing and magnifying operations are given in Figure 3 while rotations by arbitrary angles are shown in Figure 4.

We have measured the compression ratio r of our representation by the formula:

$$r \doteq \frac{\text{size of entropy coded contour files}}{\text{size of entropy coded output file}} \tag{16}$$

The Table 4 shows the dependence of the compression ratio on the t-Domain length. 40 randomly chosen Arabic characters were taken to build the training set.

The encoding process is rather time consuming. Its time dependence on number of shapes in the training set is given in Figure 5(a). On the other hand, decoding process if very fast (cf. Figure 5(b)). In this experiments contours from Arabic characters were taken as the training set.

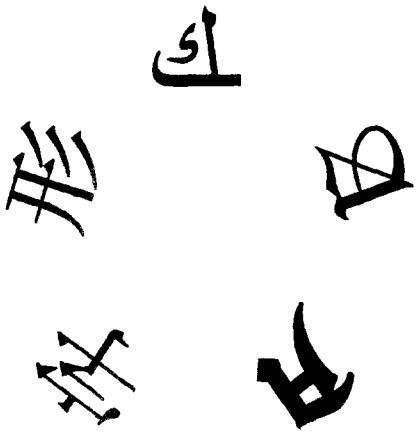
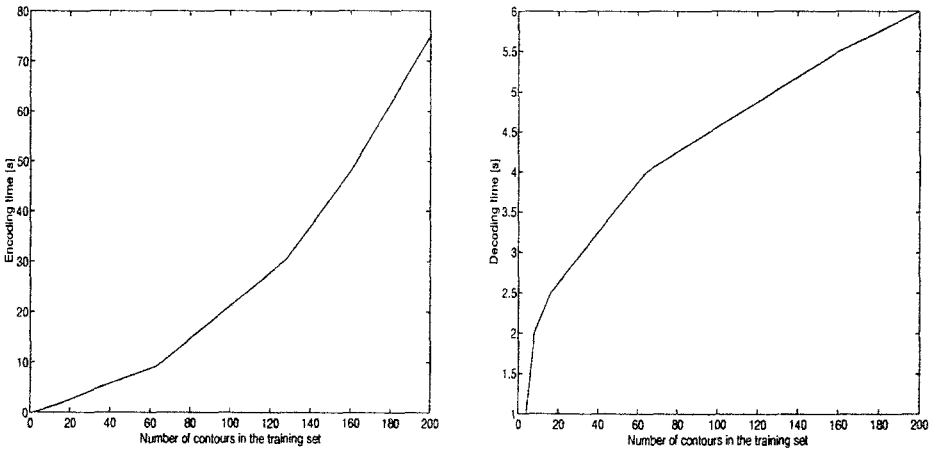


Fig.4. Rotations of shapes ($k = 16$, $\Delta l = 8$).

k	4	8	16	32	48
r	8.68	9.37	14.19	22.08	28.46

Table 4. Compression ratio as the function of t-Domain length ($t = 0.5$, $\Delta l = 8$).



(a) (b)

Fig.5. The encoding (a) and decoding (b) times as functions of number of contours in the training set ($k = 32$, $\Delta l = 8$, decoding in 5 iterations).

6 Conclusions

Fractal operator acting on collections of contours reconstructs the original shapes with a high precision after only few iterations. It is a unique technique which combines simultaneously the following features:

- shapes can be individually
  - reduced,
  - magnified,
  - rotated,

  by immediate modification of fractal operator parameters;
- compression ratio is comparable with the *chain code* method [7];
- shape representation is produced completely automatically.

References

1. Banach S. (1978): *Oeuvres*, vol. I and vol. II, Polish Scientific Publishers, Warszawa, 1978.
2. Barnsley M. and Jacquin A. (1988): Application of Recurrent Iterated Function Systems to Images, *SPIE Vol. 1001 Visual Communication and Image Processing'88*, 122-131.
3. Barnsley M. (1988): *Fractals everywhere*, Academic Press.
4. Ghuwar M. and Skarbek W, (1994): Recognition of Arabic Characters – a Survey, *ICS PAS Report* no. 740.
5. Ip H., Wong H., and Mong F. (1994): Fractal Coding of Chinese Scalable Calligraphic Fonts, *Computers & Graphics*, vol. 18, no. 3, 343-351.
6. Jacquin A. (1992): Image coding based on a fractal theory of iterated contractive image transformations, *IEEE Transactions on Image Processing,* vol. 1, no. 1, 18-30.
7. Jain A. K. (1989): *Fundamentals of digital image processing*, Prentice Hall International.
8. Mahmoud S. A. (1994): Arabic Character Recognition using Fourier Descriptors and Character Contour Encoding, *Pattern Recognition*, vol. 27, no. 6, 815-824.