Image Contour Detection
Using Neural Network-based Fractal Coding

CHEN Xin    ZHANG Li-Ming
(Electronic Engineering, Fudan University, Shanghai, 200433, China)

Abstract
In this paper, we propose to fix the selection of domain blocks when implementing fractal image coding with parallel neural network technologies. The obtained fractal code can be used to get the fine image contour after one iteration from any even gray-level images (except all black), as well as to reconstruct the original image after many times iteration from any images. At the same time, this method still keeps the features of simple structure and low computation complexity. The simulation results show that this method is effective.

1 Introduction

Barnsley was the first to propose the notion of Fractal Image coding by iterated function system (IFS), using the self-transformability of images [1]. Jacquin and Fisher developed a partitioned iterated function system (PIFS) to improve the IFS so that the fractal code is determined automatically [2][3]. But the PIFS is still complex and time-consuming.

The neural network is characterized with the ability of parallel processing, nonlinear mapping and learning, and it has been successfully used in many scopes. When neural network is used in fractal image compression [4], the computation is speeded and promise good compression results. On this basis, we propose to fix the selection of domain blocks during fractal image encoding, so the fractal code we acquired can be used to obtain the fine image contour after one iteration from any even gray-level images (except all black), as well as to reconstruct the original image after many times iteration from any images. A method to obtain the image contour by common fractal code is given in [5], it requires an accurate determination of contour blocks in the fractal encoder. Compared with that algorithm, the computation complexity of ours is little, and the contour is more accurate. If we use the algorithm of neural network-based fractal coding with fixed domain block selection, it is possible to do the image processing based on the compressed code.

2 Fractal Image Coding and Neural Network

The affine mapping $\tau$ from domain blocks to range blocks in fractal image coding at every pixel is defined as

$$
\begin{bmatrix}
  x' \\
  y' \\
  z'
\end{bmatrix} = \tau
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix} =
\begin{bmatrix}
  0.5 & 0 & 0 \\
  0 & 0.5 & 0 \\
  0 & 0 & s_i
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix} +
\begin{bmatrix}
  e_i \\
  f_i \\
  b_i
\end{bmatrix}
$$

Here, the $(x,y)$ and $z$ are the coordinates and intensity of a pixel in domain block respectively, $(x',y')$ and $z'$ are those of a pixel in range blocks, the $s_i$ is the scale of contrast and the $b_i$ is the brightness, $(e_i,f_i)$ are the location shift. The $s_i, b_i, e_i, f_i$ should be recorded at the fractal encoder and will be used in iteration at the decoder, the collage theorem ensures that the iteration is convergent. When compressing the image, the best matching domain block for every range block is
searched, if the image size is \( C \times C \), the range block size is \( K \times K \) and the domain block size is \( 2K \times 2K \), the whole image can produces \((C-2K+1)^2\) domain blocks. The calculated amount to compare the similarity of one range block and one domain block is proportional to \( K^2 \), so to the whole image, the computation complexity of fractal encoding is proportional to \((C-2K+1)^2 \cdot K^2 / K^2 = (C-2K+1)^2 \cdot C^2 \). Since \( K \) is a constant, the computation complexity of fractal encoding can be simplified to \( O(C^4) \).

The neural network can be used to define the mapping. The general model of a neural network is shown in figure 1. The inputs \( X_0, X_1, \ldots, X_{N-1} \) are connected to the neuron \( v \) through the weights \( W_{00}, W_{11}, \ldots, W_{N-1} \) respectively. \( z \), the output of \( v \), is determined by the function \( o : \quad z = o\left(\sum_{i=0}^{N} w_i x - \theta \right) \), where \( \theta \) is the threshold.

！[Figure 1: the basic model of neuron]

The computations of neural networks can execute in parallel, and this is the major reason why we can speed up the compression process (search of fractal codes).

3 Learning of the Neural Network

As shown in figure 2, the neural network is used to implement fractal image coding. Each pixel of the image is considered as a range, and is corresponding to a neuron \( j \) whose output is the gray level of this pixel. For each neuron, there are four inputs, they are the gray level of pixels \( i, i+1, i+2 \) and \( i+3 \), which forms the domain.

Therefore, The output value \( z_{j}^{'} \) of neuron \( j \) is determined by the values \( z_i, z_{i+1}, z_{i+2}, z_{i+3} \), the corresponding weights \( w_{ji}, w_{ji+1}, w_{ji+2}, w_{ji+3} \), the threshold \( \theta_j \), and the generation function \( o(x) \). It is defined as:

\[
\begin{align*}
   z_{j}^{'} = & \left\{ \begin{array}{ll}
   \sum_{k=i}^{i+3} w_{jk} z_k - \theta_j & \text{if } x = 255 \\
   \sum_{k=i}^{i+3} w_{jk} z_k - \theta_j & \text{if } 0 < x < 255 \\
   0 & \text{if } x \leq 0
   \end{array} \right.
\end{align*}
\]

Where \( o(x) = \begin{cases} 255, & x \geq 255 \\ x, & 0 < x < 255 \\ 0, & x \leq 0 \end{cases} \)

！[Figure 2: the architecture of the neural network-based fractal coding]

The corresponding weights and the threshold represent the mapping from the domain to the range. The proper weights and thresholds can be obtained in compression (training) procedure, and the original image can be reconstructed in decompression (retrieving) procedure. Then, the neural network approach can implement the computation of PIFS.

According to the topology of the proposed neural network, the steps of learning of neurons are as follows:

1) Initialize \( w_{ji}, w_{ji+1}, w_{ji+2}, w_{ji+3} \) and \( \theta_j \) as small random numbers;

2) Compute \( d_j = z_j^{true} - z_j^{'} \), where \( z_j^{'} \) is
the output value of neuron and $z_j^{true}$ is the true gray level of pixel $j$:

3) Adjust the weights and threshold as: $\Delta w_{jk} = \eta \times \frac{d_j}{z_k} \quad \Delta \theta_j = - \eta \times d_j$.

Where $k = i, i+1, i+2, i+3$ and $\eta$ is the learning rate parameter;

4) Compute the output of neuron with the adjusted weights and threshold, and repeat the learning steps iteratively until it is acceptable.

At the decoder, the image is reconstructed from any initial image by iteration. At the time $t$, the output of neuron $j$ is defined as equation (1).

$$z_j^{(t)} = \sigma\left(\sum_{k=0}^{k^3} w_{jk} z_k^{(t)} - \theta_j\right)$$

At the time $t+1$, the inputs of neuron $j$ are changed:

$$z_j^{(t+1)} = z_j^{(t)}$$

Then the outputs of neurons are changed iteratively until the system reaches a stable state.

When the image is processed in parallel, it can be expressed in matrix. We define the operation between two matrices equals to the operation between corresponding elements of these matrices respectively. For example, $A, B, C, D \in \mathbb{R}^{m \times n}$, $k = 1, 2, 3, 4$, $A_{ij}, B_{ij}, C_{ij}, D_{ij}$ are the elements of these matrices respectively. $A = B / C$ is equals to $A_{ij} = B_{ij} / C_{ij}$, $A = B \times C$ is equals to $A_{ij} = B_{ij} \times C_{ij}$, $A = \sum_{k=1}^{4} D_{ij}$ is equals to $A_{ij} = \sum_{k=1}^{4} D_{ij}$.

If the image size is $m \times n$, $Z^{true}, Z \in \mathbb{R}^{m \times n}$ are the original image and output image respectively, $Z^1, Z^2, Z^3, Z^4 \in \mathbb{R}^{m \times n}$ are constructed so that $Z^k_{i,j}, k = 1, 2, 3, 4$ are the four inputs of neuron at $(i, j)$, and they are transformed from $Z^{true}$: $Z^k = P^k(Z'^{true})$. The weights and thresholds of all the neurons form the matrix: $W^k, k = 1, 2, 3, 4$, $\theta \in \mathbb{R}^{m \times n}$, where $W_{i,j}$ is the corresponding weight between $Z^k_{i,j}$ and $Z'_{i,j}$, $\theta_{i,j}$ is the threshold of $Z'_{i,j}$. The generating function of all the neurons is:

$$Z' = \sigma\left(\sum_{k=0}^{k^3} W^k \times Z^k - \theta\right).$$

The target of the study of neurons is: $\min_{W^k} (|Z'^{true} - Z_{i,j}'|^2)$.

Then the weights matrixes and threshold matrix are modified:

$$\Delta W^k = \eta \times d^\top / Z^k, \Delta \theta = - \eta \times d$$

(2)

where $\mathbf{d} = Z^{true} - Z'$, $k = 1, 2, 3, 4$ and $\eta$ is the learning rate parameter. In order to guarantee that the mappings from domains to ranges are convergent, the weights are all bounded between 0 and 1. In order to increase the learning speed of neurons, the range of threshold is set within $-10$ and $0$.

Then the $W^k, \theta$ are post-processed, the compression ratio is about 10.

The neuron network can get good results after being trained for 8 times. During the learning of the neural network, every adjustment of the weights and threshold of one neuron needs only 23 mathematical operations (including addition, subtraction, multiplication and division). It means for an image whose size is $C \times C$, the computation complexity is $8 \times 23 \times C \times C$, and can be simplified as
Compared it with \( O(C^2) \), the computation complexity of common PIFS, the neural network-based fractal image coding is timesaving.

At the decoder, the output of time \( t \):

\[
Z_{ij}^{(t)} = o \left( \sum_{k=1}^{4} W_{ik}^k \times Z_{ik}^{(t)} - \theta_{i,j} \right)
\]  

(3)

is used as the inputs of time \( t+1 \):

\[
Z_{ik}^{(t+1)} = P_k(Z_{ik}^{(t)}), k=1,2,3,4
\]  

(4)

4 Image Contour Extraction Using Fractal Code

Because of the spatial continuity in gray level of the image, the adjacent pixels of range pixel \( j \) are close to it in gray level, so these adjacent pixels can be used as the domain pixels. Figure 3 shows the position relationship of range and domain pixels.

The range neuron is at the position: \((i,j)\), and the domain inputs are:

\[
Z_{i,j}^1 = Z_{i,j-1}^{true}, Z_{i,j}^2 = Z_{i-1,j}^{true},
\]

\[
Z_{i,j}^3 = Z_{i+1,j}^{true} \triangleq Z_{i,j+1}^{true}
\]

The reason for this is that the weights \( W_{ik}^k \) will have some distinct characteristics, so the image contour can be extracted from the fractal code.

![Figure 3](image)

Figure 3 the position relationship of range and domain pixels

In the smooth area of image, the difference of gray levels between range pixel and domain pixels is small, \( Z_{ij}^{true} \approx Z_{i,j-1}^{true} \approx Z_{i-1,j}^{true} \approx Z_{i+1,j}^{true} \), according to equation (2), every modification:

\[
\Delta W_{ij}^1 \approx \Delta W_{ij}^2 \approx \Delta W_{ij}^3 \approx \Delta W_{ij}^4,
\]  

finally \( W_{ij}^1 \approx W_{ij}^2 \approx W_{ij}^3 \approx W_{ij}^4 \). And if the initialization values of weights are very small, we have

\[
\theta_{i,j} \ll \sum_{k=1}^{4} W_{ik}^k \times Z_{ik}^k,
\]  

(5)

Then

\[
Z_{ij}^{true} = o \left( \sum_{k=1}^{4} W_{ik}^k \times Z_{ik}^k \right),
\]

So

\[
W_{ij}^1 = W_{ij}^2 = W_{ij}^3 = W_{ij}^4 \approx 0.25\]  

(6-a)

\[
\sum_{k=1}^{4} W_{ik}^k \approx 1\]  

(6-b)

In the edge of image, the gray level changes quickly. Take figure 4 for example, the range pixel \( j \) is an contour point, which is on an edge where the gray level changes vertically, and \( i,j \) belongs to the lower gray level area, it is surrounded by one or several pixels with higher gray level. If the higher gray level is \( q \) times of the lower one, then for the domain pixels,

\[
Z_{i,j}^{true} \approx Z_{i,j-1}^{true} \approx Z_{i-1,j}^{true} \approx Z_{i+1,j}^{true},
\]

\[
Z_{i+1,j}^{true} = q \cdot Z_{i,j}^{true}, q > 1.
\]

According to equation (2), every modification

\[
\Delta W_{ij}^1 \approx \Delta W_{ij}^2 \approx \Delta W_{ij}^3 \approx \Delta W_{ij}^4 \approx \frac{1}{q} \cdot \Delta W_{i,j}^1,
\]

finally

\[
W_{ij}^1 \approx W_{ij}^2 \approx W_{ij}^3 \approx W_{ij}^4 \approx \frac{1}{q} \cdot W_{ij}^1
\]

\[
Z_{ij}^{true} \approx o \left( \sum_{k=1}^{4} W_{ik}^k \times Z_{ik}^k \right) \approx 4 \times W_{ij}^1 \times Z_{i,j}^{true},
\]

then

\[
W_{ij}^1 = W_{ij}^2 = W_{ij}^3 = 0.25, W_{ij}^4 = \frac{1}{q} \times 0.25
\]  

(7-a)
\begin{equation}
\sum_{k=1}^{4} W_{i,j}^k < 1 \quad (7-b)
\end{equation}

Figure 4: the contour point of image

For the contour point of \( i,j \), it is surrounded by one or several pixels with lower gray level. With the similar analysis but \( q < 1 \), we cannot get equation (7-b). But the contour is reflected by point \( i,j \), the contour is extracted. And the threshold \( \Theta \) can be set to a lower limit, the equation (5) is true for most points and won’t cause many mistakes.

When extracting the image contour, from any even gray level images (suppose the gray level \( Z_{i,j}^{(0)} = l, l \neq 0 \), at time \( t=1 \), from

\[ Z_{i,j}^{(1)} = P_k(Z_{i,j}^{(0)}) \]

we can get \( Z_{i,j}^{(1)} = l \) for the smooth area of image substitute equation (6-b) into equation (3) we get \( Z_{i,j}^{(1)} - l \approx 0 \) but for the contour points, substitute equation (7-b) into equation (3) we get \( Z_{i,j}^{(1)} < l \) and when \( q \) is large \( \sum_{k=1}^{4} W_{i,j}^k \) becomes smaller \( |Z_{i,j}^{(1)} - l| \) is a large value. With a set threshold \( th \) if \( |Z_{i,j}^{(1)} - l| > th \), then point \( i,j \) is a contour point, otherwise point \( i,j \) belongs to a smooth area. It means that after one iteration, the contour is extracted. If \( th \) is too large, some contour points will be lost, if \( th \) is too small, a small change in gray level is detected. For different images, it can be adjusted easily to get better results. Normally \( th \) is set between 20 to 40.

**Simulation Results**

Figure 5 is the simulation results of this algorithm. Figure 5-b and 5-g are the image contours detected by our algorithm, figure 5-c is the transitional results of reconstructing image of lenna, which is after 30 times iteration from an initial image that is all white. Figure 5-d is the final reconstructed image of lenna. Figure 5-e is the contour detected from normal fractal coding [5], compared figure 5-b with figure 5-e, it is clear that in the former one, the eyes and ornaments of the cap are all detected, because the range’s size is one pixel, which is fine enough to acquire the characteristic of small areas. Also there are no Sierpinski triangles in figure 5-b, which appear in figure 5-e as the error in the determination of the contour, and it is due to the inaccuracy in the determination of contour blocks during fractal encoding. The result in figure 5-b is better.

In this paper, we propose to fix the selection of domain blocks when implementing fractal image coding with parallel neural network technologies. Without the reconstruction of images, the image contour is extracted directly from the fractal code. If images are compressed with this algorithm to form a fractal code database, then the quick browse and search of images can be implemented, or it can be used as the basis of other image processing technologies such as image editing and recognition [4][6].

**References**


Figure 5  (a) original image of lenna, (b) the detected contour of lenna after one iteration, (c) transitional result of reconstructing after 30 times iteration, (d) reconstructed image of lenna, (d) the detected contour of lenna given in [5], (e) original image of peppers, (f) the detected contour of peppers after one iteration