Self-Affine Mapping System and Its Application to Object Contour Extraction

Takashi Ida, *Member, IEEE*, and Yoko Sambonsugi

***Abstract*—A self-affine mapping system which has conventionally been used to produce fractal images is used to fit rough lines to contours. The self-affine map's parameters are detected by analyzing the blockwise self-similarity of a grayscale image using a simplified algorithm in fractal encoding. The phenomenon that edges attract mapping points in self-affine mapping is utilized in the proposed method. The boundary of the foreground region of an alpha mask is fitted by mapping iterations of the region. It is shown that the proposed method accurately produces not only smooth curves but also sharp corners, and has the ability to extract both distinct edges and blurred edges using the same parameter. It is also shown that even large gaps between the hand-drawn line and the contour can be fitted well by the recursive procedure of the proposed algorithm, in which the block size is progressively decreased. These features reduce the time required for drawing contours by hand.**

***Index Terms*—Boundary extraction, chaos, fractals, segmentation.**

I. Introduction

THE EXTRACTION of the contours of objects in an image has been investigated actively as it is a significant technology in image editing, image searching, image recognition, and other image processing procedures. Our proposal is aimed at applications which demand highly accurate contours, such as object-based composition, mixing, and editing. Once an object contour is extracted, the background image outside the object can be replaced by another image or a composite image can be made with other objects. On the other hand, MPEG-4, the international standard of moving picture coding, includes the arbitrary shape coding [1] which can be utilized for object-based composition. Object extraction needs are becoming more exacting in image communication and storage [2].

It takes much time and effort to draw an accurate contour by hand. One solution often used to overcome this problem is that the line is first drawn roughly near the contour and then it is fitted automatically to the contour. An active contour model called the snake model has been investigated widely as a contour fitting method [3]–[5]. In the snake method, an energy function is defined, according to continuity and smoothness of contour line and image features. The object contour is extracted by minimizing the energy function. Even though the roughly drawn line is fairly distant from the object contour, the line is able to be fitted to the contour in an ideal case: the object contour is formed by distinct edges and the background is almost flat. However, blurred edges of the object contour or the texture in the background introduce misextraction. It is also difficult to extract sharp corners accurately because of the smoothness term in the energy function.

A new fitting method which provides highly accurate contours is proposed in this paper. The concept of our method is quite different from that of the snake method. A self-affine mapping system instead of the energy minimization procedure is used to approach and fit the roughly drawn line to the object contour. The contractive self-affine mapping system has been used to produce fractal figures [6], [7]. We use it for object contour extraction in this paper. The self-affine map's parameters are detected by analyzing the blockwise self-similarity of a grayscale image using a simplified algorithm in fractal encoding [7], [8]. In an earlier paper [9], we showed that edges attract mapping points during iteration of the map when the mapping points are initially set near the edges. This attraction phenomenon is also utilized in the method proposed here, and the contour is obtained as an attractor of the mapping system [10]. The object contour is extracted as a self-similar curve instead of a smooth curve. The proposed method can extract sharp corners since they have the self-similarity.

It is shown that the mapping system extracted both distinct and blurred contours of objects, both sharp corners and smooth curves. As a result, highly accurate contours were extracted. It is also shown that even large gaps between hand-drawn lines and contours can be fitted well by the recursive procedure of the proposed algorithm, in which the block size is progressively decreased [11].

In Section II, the self-affine map is defined and the typical behavior of the mapping system and its conventional applications are described briefly. The proposed method and the experimental results are shown in Sections III and IV, respectively. The proposed and the conventional snake method are compared in Section V and the conclusions are outlined in Section VI.

II. Self-Affine Mapping System

A. Mapping System Based on Image Self-Similarity

An image having a support $G \subset \boldsymbol{R}^n$ with intensity $g(\boldsymbol{x})$ for all $\boldsymbol{x} \in G$ is considered. In the case that affine maps α_i: $A_i \to \boldsymbol{R}^n$ and β_i: $\boldsymbol{R}^1 \to \boldsymbol{R}^1$ exist such that

$$\forall \boldsymbol{x} \in A_i,\ g(\boldsymbol{x}) = \beta_i\left(g\left(\alpha_i(\boldsymbol{x})\right)\right),\ (i = 1, 2, \cdots, I) \qquad (1)$$

for some regions $A_i \subset G$, we say that the texture in A_i is similar to the texture in $\alpha_i(A_i)$, and the image is blockwise self-similar

Manuscript received April 23, 1999; revised April 6, 2000. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Robert J. Schalkoff.

The authors are with Multimedia Laboratory, Corporate Research & Development Center, Toshiba Corporation, Kawasaki 212-8582, Japan (e-mail: takashi.ida@toshiba.co.jp; youko@eel.rdc.toshiba.co.jp).

Publisher Item Identifier S 1057-7149(00)07589-8.

Fig. 1. Texture in each smaller block is almost similar to the texture in the larger block including the smaller block.

on A_i and $\alpha_i(A_i)$. Where I is the number of region A_i. The set $\{A_i, \alpha_i, \beta_i | i = 1, 2, \cdots, I\}$ is called the self-affine model of the image.

An example of the self-affine model with $I = 5$ is shown in Fig. 1. Where the smaller blocks are set as $A_i (i = 1, 2, \cdots, 5)$. Each texture in A_i is similar to the texture in the larger block $\alpha_i(A_i)$ including A_i. Here, α_i and β_i were set as expanding maps and identity maps, respectively. Small difference between the left-hand term and the right-hand term in (1) was allowed. It can be seen that the texture pattern in the smaller block is almost the same as that in the larger block. This self-affine model is regarded as an extension of the compressed data of fractal image coding [7], [8]. In fractal coding, A_i are set to cover the whole image and do not overlap each other, α_i are restricted to expanding maps, and β_i are restricted to contracting maps.

If (1) is assumed, the following equation is also obtained:

$$\forall \boldsymbol{x} \in \alpha_i(A_i),\ g(\boldsymbol{x}) = \beta_i^{-1}\left(g\left(\alpha_i^{-1}(\boldsymbol{x})\right)\right),\ (i = 1, 2, \cdots, I). \tag{2}$$

Therefore, another self-affine model $\{\alpha_i(A_i), \alpha_i^{-1}, \beta_i^{-1}\}$ is available directly from a self-affine model $\{A_i, \alpha_i, \beta_i\}$.

Let Ω denote the set of all subsets of G. A self-affine map S: $\Omega \rightarrow \Omega$ is defined by

$$S(\boldsymbol{X}) = \left\{\bigcup_{i=1}^{I} \alpha_i\left(A_i \cap \boldsymbol{X}\right)\right\} \cup C. \tag{3}$$

Here, C is a constant set referred to as the condensation. When a region $\boldsymbol{X}$ is provided, the intersection of A_i and $\boldsymbol{X}$ is mapped by α_i (see Fig. 2). The union of those mapped regions and a constant condensation set are the results of the map.

When all α_i are contractive, S is equivalent to the well-known transformation referred to as LIFS (local iterated function systems) [7] and produces fractal figures by iterations of S with an initial set $\boldsymbol{X}_0$ [6]. Fig. 3 shows process whereby the Sierpinski triangle is produced. A square region was set as $\boldsymbol{X}_0$. Here, $I = 3$ and $A_i (i = 1, 2, 3)$ were set as the same region, the whole

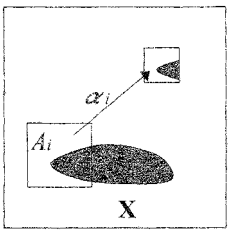

Fig. 2. Self-affine mapping. The intersection of A_i and $\boldsymbol{X}$ is mapped by α_i.

image plane G. α_i were set to map G into top right part, bottom left part and bottom right part, respectively.

B. Extraction of the Self-Affine Model

Let $n = 2$ for map w_i with square domain $W_i \subset G$, which is expressed by

$$w_i(\boldsymbol{x}) = r_i\left(\boldsymbol{x} - \tilde{\boldsymbol{x}}_i\right) + \left(\boldsymbol{\tau}_i + \tilde{\boldsymbol{x}}_i\right), \tag{4}$$

$$r_i > 1 \tag{5}$$

where $\tilde{\boldsymbol{x}}_i$ is the center point of W_i. An example of W_i and $M_i = w_i(W_i)$ is shown in Fig. 4. W_i is translated by vector $\boldsymbol{\tau}_i$ and expanded by r_i.

Extracting a self-affine model using w_i as α_i and v_i as β_i in (1) is considered, where

$$v_i(z) = p_i z + q_i, \tag{6}$$

$$0 \le p_i \le 1. \tag{7}$$

First, some W_i are allocated. The allocation method depends on the application of the self-affine model. Second, a search is performed to identify the parameters of maps r_i, s_i, t_i, p_i, and q_i which provide the minimum difference between the left-hand and right-hand sides of (1) for every W_i. Here, s_i and t_i are components of the vector $\boldsymbol{\tau}_i$. The sets $\{W_i, w_i, v_i\}$ and $\{M_i, m_i, u_i\}$, where $M_i = w_i(W_i)$, $m_i = w_i^{-1}$, and $u_i = v_i^{-1}$, are the extracted self-affine models.

Note that a block matching algorithm is usually used for the search [7], [8].

Block Matching Algorithm

Step 1: Predetermined initial values are set to $r, s,$ and t. A sufficiently large value is set to E, which keeps the minimum difference.

Step 2: The sampled intensity values $g(w(\boldsymbol{x}))$ for every $\boldsymbol{x} \in W_i$ are calculated. Usually, a bilinear filter or any other interpolation filter is used to obtain the values because sampling points may be placed between the pixel points of images.

Step 3: Predetermined initial values are set to p and q.

Step 4: $v(g(w(\boldsymbol{x})))$ is calculated for every $\boldsymbol{x} \in W_i$.

Step 5: The differences between $\{g(\boldsymbol{x}) | \boldsymbol{x} \in W_i\}$ and $\{v(g(w(\boldsymbol{x}))) | \boldsymbol{x} \in W_i\}$ are calculated. The mean squared error or the mean absolute distance is used as a measure

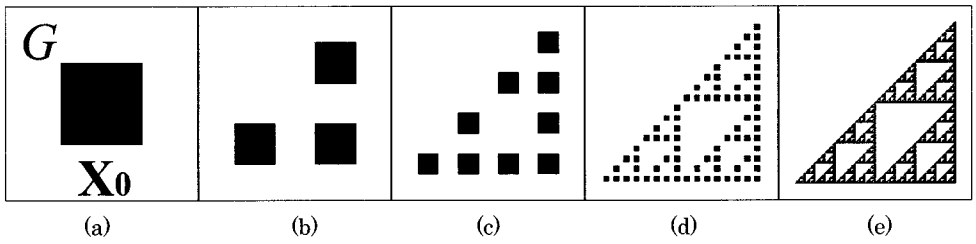

Fig. 3. Sierpinski triangle is produced by contractive self-affine mapping. (a) Square region was set as the initial set, (b) region was mapped into three parts: top right, bottom left, and bottom right, (c) two iterations of mapping, (d) four iterations, and (e) eight iterations.

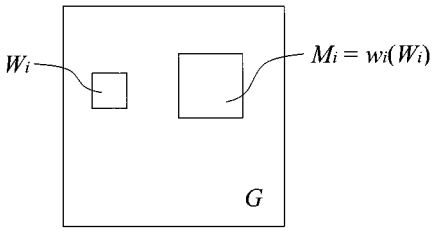

Fig. 4. Example of W_i and $M_i = w_i(W_i)$.

of the difference. If the difference is smaller than E, E is replaced by the difference and the current $r, s, t, p,$ and q are kept as $r_i, s_i, t_i, p_i,$ and q_i.

Step 6: In the case that all prepared p and q are checked, the procedure goes to step 7. In other cases, p and q are changed in the predetermined order and the procedure returns to step 4.

Step 7: In the case that all prepared $r, s,$ and t are checked, the procedure is complete. In other cases, $r, s,$ and t are changed in the predetermined order and the procedure returns to step 2.

C. Behavior of Self-Affine Mapping System

We will show two typical behaviors of the self-affine mapping system in this section. All α_i are expanding for the first case and contracting for the second case.

Figs. 5 and 6 show the results of self-affine mapping using the self-affine model of the test image Lenna (512×512, 8 bits/pixel). G was partitioned into blocks of 16×16 pixels, and they were set as $W_1, W_2, \cdots, W_{1024}$ in order of blocks from left to right with block lines from top to bottom. The block size of M was set to double that of W ($r = 2$), and v was simplified as an identity map ($p = 1, q = 0$). The block matching algorithm was used for extraction of the self-affine model in which s and t were moved pixel by pixel in $-8 < s, t < 8$, and the mean absolute distance was used as a measure of the difference. The initial set $\boldsymbol{X}_0$ consisted of 16 384 points placed uniformly in G at intervals of four pixels.

Let S_w be a self-affine map with expanding maps such that

$$\alpha_i = w_i \tag{8}$$
$$A_i = W_i \tag{9}$$
$$C = \phi \tag{10}$$

in (3). The results of two iterations, six iterations, and 20 iterations of S_w are shown in Fig. 5(a)–(c), respectively. Here, all points in $\boldsymbol{X}$ are shown as white points on the original images. The coordinates of every point of $\boldsymbol{X}$ were stored in the memory as floating-point variables that were overwritten for each map. The mapped points were repelled from the edges and formed groups. The points moved within each group but did not jump to other groups as a result of the mapping. This is because the points near an edge leave the edge by mapping w_i in the ratio of r_i, and some trapping regions form in flat regions surrounded by edges [9]. The number of points in $\boldsymbol{X}$ does not change because the union of W_i covers the whole of G and W_i do not overlap each other.

The results for the self-affine map S_m with contracting maps such that

$$\alpha_i = m_i \tag{11}$$
$$A_i = M_i \tag{12}$$
$$C = \phi \tag{13}$$

are shown in Fig. 6. In this experiment, many points belong to multiple M_i; therefore, the number of points increases rapidly with each mapping. In the above case, the initial 16 384 points became 65 536 points after the first mapping. We traced only the point by m_k where

$$k = \max\{i | M_i \text{ includes the point}\} \tag{14}$$

when the point was in multiple M_i due to limitations in the computer memory size. Because points which do not belong to any M_i disappear in the next S_m, the number of points decreased to 12 554 after 20 iterations of S_m. Many mapped points were attracted to the nearby edges. This is because the distances between an edge and the points within the same M_i are reduced by mapping m_i in the ratio of $1/r_i$. It seems that there are very few points in Fig. 6(c) because the distances between many of the mapped points were less than the pixel interval.

Thus, it was found that mapped points are repelled from (or attracted to) edges by the self-affine mapping S when α_i in S are expanding (or contracting) maps.

D. Image Processing Using Self-Affine Maps

Some image representation methods using the self-affine map parametrized by the self-affine model have been proposed as applications. These include fractal segmentation [9], [12], edge detection [9], and compression with pixel chaining decoding

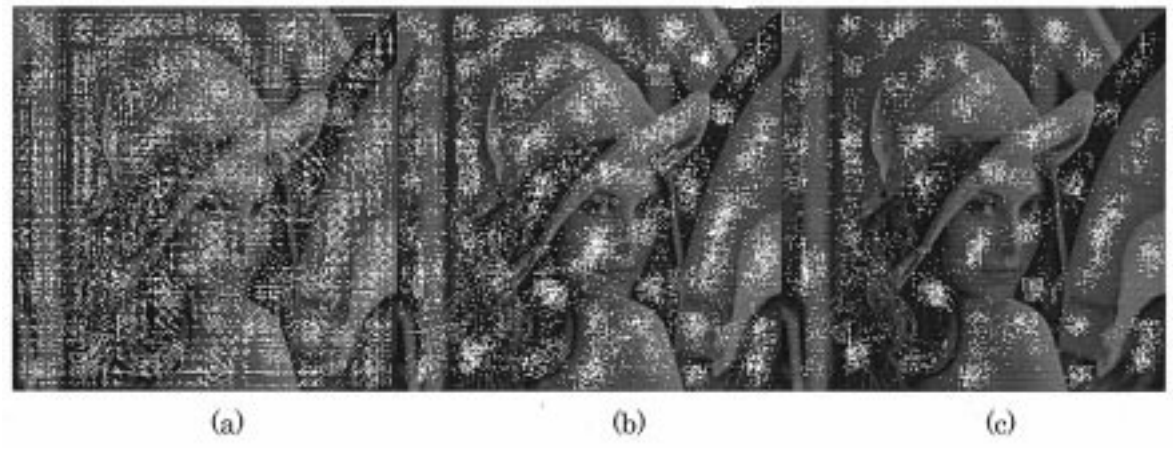

Fig. 5. Results of expanding S_w. (a) Two iterations, (b) six iterations, and (c) 20 iterations. The mapped points (white) were repelled from the edges.

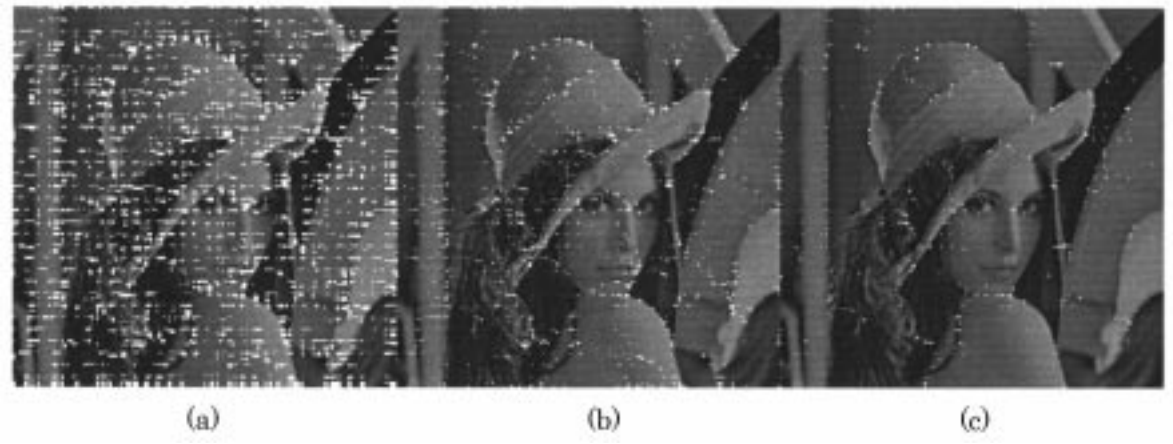

Fig. 6. Results of contracting S_m. (a) Two iterations, (b) six iterations, and (c) 20 iterations. Many mapped points were attracted to nearby edges.

TABLE I
APPLICATIONS OF SELF-AFFINE MAPPING

Applications	Self-affine modeling		Self-affine map			
	W allocation	Range of p	α	A	C	$\mathbf{X}_0$
Segmentation	Method A	$p = 1$ (preferred)	w	W	ϕ	Pixel points
Edge detection	Method B	$p = 1$ (preferred)	m	M	ϕ	G
Compression	Method A	$\|p\| < 1$ (mandatory)	w	W	ϕ	Pixel points
Boundary fitting	Method C	$p = 1$ (preferred)	m	M	$\mathbf{X}_0 \cap \bigcap_{i=1}^{I} \overline{W_i}$	Given foreground

[12], [13]. In all of these applications, first a self-affine model is extracted and then iteration of the self-affine map is performed.

The differences in the methods are listed in Table I. Here, the W allocation methods are as follows.

W allocation method A:

G is partitioned into W_i such that W_i do not overlap each other and every point in G is covered by one W_i.

W allocation method B:

G is partitioned into blocks as in method A. The blocks in which the variance of pixel values is greater than a predetermined value are set as W_i.

In segmentation using S_w, the points initially at the pixel positions are mapped, and mapped points such as those in Fig. 5(c) are classified into clusters. The segmented image is obtained by putting a color on the initial position of each mapped point depending on the cluster to which the mapped point belongs.

The edge image is directly obtained as the limit set of the S_m iterations. Here, $\boldsymbol{X}$ is an area instead of a set of points, and each pixel $\boldsymbol{x} \in W_i$ is replaced by the corresponding sampling value at $w_i(\boldsymbol{x})$ as S_m in the actual procedure. $\boldsymbol{X}$ forms bands near edges as the union of W_i by applying S_m once to G, the bandwidth is reduced by repeating S_m, and the edges are detected.

$p = 1$ is preferred for segmentation and edge detection because the self-affine model, which is related to the self-similarity, is extracted with this condition [9].

Fractal image coding was originally proposed by Barnsley [14], [15] as a compression method for binary images and was applied to grayscale images by Barnsley and Jacquin [7], [8]. The image is extracted as a self-affine model, using the block matching algorithm with the condition $|p| < 1$, which is needed to decode the image. Usually, a contractive transformation of images called the fractal transformation is used for decoding. The image is obtained as an attractor with the transformation. In another decoding method using self-affine mapping, a map of points in $\boldsymbol{R}^3$ combined with S_w and u_i is defined. The decoded image is obtained as the set of intensity values which do not diverge as a result of iteration of the map [13]. Some variations of this decoding method are shown in references [12] and [13].

The boundary fitting referred to in Table I is shown in the next section.

III. Proposed Contour Extraction

We propose a new method for boundary fitting of alpha masks to object contours in which the contour is obtained by calculating the attractor of the self-affine mapping system. The phenomenon that edges attract mapping points in contractive self-affine mapping is utilized in the proposed method.

Fig. 12(a) is an example of a hand-drawn line for extracting the contour of the woman in the grayscale image. A binary image called an alpha mask is made by setting 1 for the pixels inside the line to define them as the foreground and 0 for the other pixels to define them as the background. Our purpose is to fit the boundary $\boldsymbol{b}$ between the foreground and background in the alpha mask to the object contour $\boldsymbol{c}$ in the grayscale image.

A. Proposed Method

The proposed algorithm using w and v is as follows.

Contour Extraction Algorithm

Step 1: The side length e of W_i is set to $e_{\max}$.

Step 2: $W_i (i = 1, 2, \cdots, I)$ are placed using the W allocation method C as follows.

W allocation method C:

The variable k is set to 1. The alpha mask is scanned pixel by pixel from left to right and line by line from top to bottom. When a pixel has a different value from the upper pixel or the left-hand pixel and is not included in a $W_i (i < k)$ which has been placed already, W_k is added centered on the pixel and k is incremented to $k+1$.

Step 3: The self-affine model is extracted.

Step 4: The map S is iterated η times to the foreground region of the alpha mask. In (3), let

$$\alpha_i = m_i \left(= w_i^{-1}\right) \tag{15}$$

$$A_i = M_i \left(= w_i (W_i)\right). \tag{16}$$

C is fixed to

$$C = \boldsymbol{X}_0 \cap \bigcap_{i=1}^{I} \overline{W}_i \tag{17}$$

to keep the regions where W_i is not placed as they are. Here, $\boldsymbol{X}_0$ is the foreground region of the initial alpha mask. The procedure of each map S is such that the area

$$\bigcup_{i=1}^{I} W_i \tag{18}$$

of the alpha mask is overwritten by pixel value 0 first. Pixel value 1 is set at each pixel $\boldsymbol{x} \in W_i$ such that the corresponding sampling value at $w_i(\boldsymbol{x})$ is equal to 1.

Step 5: Let $e = e/2$. In the case that e is smaller than $e_{\min} (\leq e_{\max})$, the procedure is complete. In other cases, the procedure returns to step 2.

B. Desirable Conditions for the Proposed Method

For $\boldsymbol{b}$ to exactly match $\boldsymbol{c}$, the following conditions must be satisfied.

Condition 1) $\boldsymbol{c}$ is equal to the invariant set of S.

Condition 2) All m_i are contractive.

Condition 3) $\mathbf{b}$ is sufficiently close to $\boldsymbol{c}$.

For condition 1, if $\boldsymbol{c} \cap W_i$ is similar to $\boldsymbol{c} \cap M_i$ for every i, $\boldsymbol{c}$ is equal to the invariant set of S. Mismatches in the similarity result in boundary fitting errors. Small e values increase the probability of this condition being satisfied.

For condition 2 to be satisfied, it is sufficient if (5) is satisfied.

An ideal condition for condition 3 is that both $\boldsymbol{b}$ and $\boldsymbol{c}$ are covered by W_i

$$\boldsymbol{b} \subset \bigcup_{i=1}^{I} W_i \tag{19}$$

$$\boldsymbol{c} \subset \bigcup_{i=1}^{I} W_i \tag{20}$$

and parts of $\boldsymbol{b}$ and $\boldsymbol{c}$ are within every W_i

$$\forall i, \boldsymbol{b} \cap W_i \neq \phi \quad \text{and} \tag{21}$$

$$\forall i, \boldsymbol{c} \cap W_i \neq \phi. \tag{22}$$

Equations (19) and (21) are satisfied when W allocation method C is used. Larger e values increase the probability of (20) and (22) being satisfied. Since W_i is set centered on a pixel in $\boldsymbol{b}$, the side length of W_i should be more than double the length of the gap between $\boldsymbol{b}$ and $\boldsymbol{c}$ for condition 3 to be satisfied.

C. Parameter Setting

The proposed method works well when each W_i includes a part of $\boldsymbol{c}$ as mentioned above. In such cases, a similar block can often be found in candidate blocks including W_i, and it is estimated that M_i includes an edge other than $\boldsymbol{c}$ when an M_i is selected from a place that is far from W_i. Therefore, the search

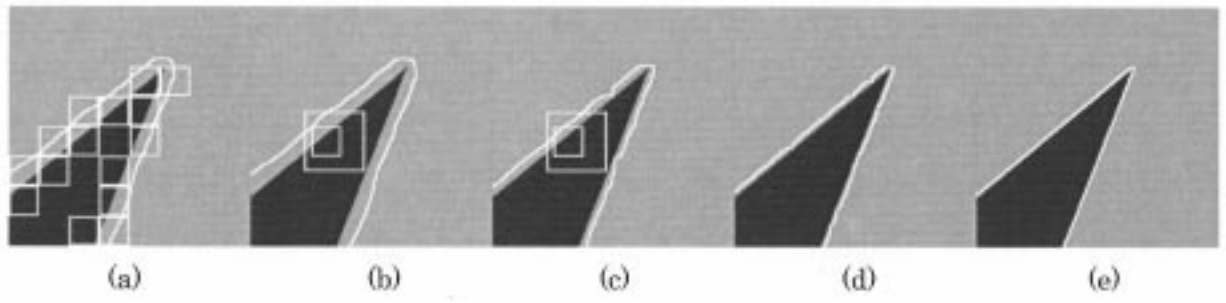

Fig. 7. Fitting to a sharp corner from outside. (a) W allocation. (b) Initial state. (c) One iteration. (d) Two iterations. (e) Five iterations. The alpha mask boundary $\boldsymbol{b}$ and blocks W and M are shown with white lines. The gap between $\boldsymbol{b}$ and the object contour $\boldsymbol{c}$ was reduced with each successive mapping iteration, and $\boldsymbol{b}$ matched $\boldsymbol{c}$ in (e).

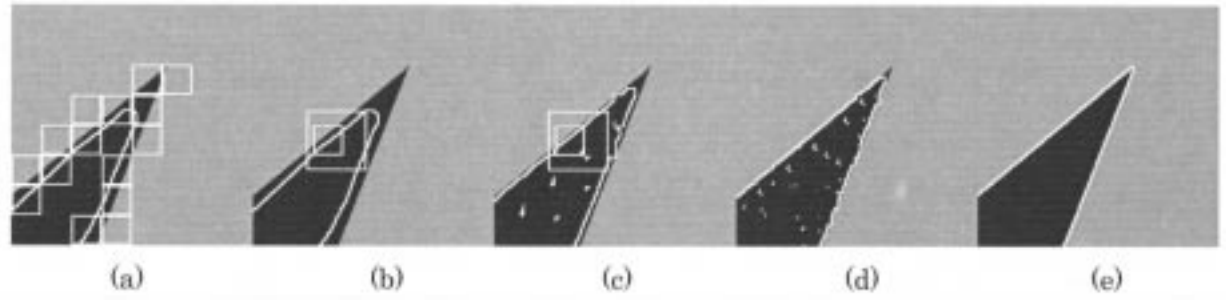

Fig. 8. Fitting to a sharp corner from inside. (a) W allocation. (b) Initial state. (c) One iteration. (d) Two iterations. (e) Five iterations. The small background regions which appeared in (c) disappeared eventually, and the result shown in (e) was almost the same as that in 7.

range of the translation parameters in block matching should be such that M_i includes W_i and

$$-\frac{r-1}{2}e \le s, t \le \frac{r-1}{2}e \tag{23}$$

for images which include edges other than $\boldsymbol{c}$. Satisfaction of (23), however, does not always provide the best result. In experiments using simple shapes which have no edge other than $\boldsymbol{c}$, we used larger ranges than those specified by inequality (23) because there was no fear of detecting another edge. This approach provided better results.

In the case of the map v, extraction of the self-affine model is better when v is the identity mapping, i.e. $p = 1$ and $q = 0$ [9].

Because 1) the maximum gap length between $\boldsymbol{b}$ and $\boldsymbol{c}$ is about $e/2$, 2) the gap is reduced in the ratio of $1/r$ with each mapping, and 3) $\boldsymbol{X}$ does not change further when the gap becomes smaller than the pixel interval

$$\frac{e}{2}\left(\frac{1}{r}\right)^{\eta} < 1 \tag{24}$$

and η, which presents a sufficient number of iterations of S, is given by

$$\eta > \frac{\log\frac{e}{2}}{\log r}. \tag{25}$$

IV. Experiments

In the experiments, the block size of M_i was set to double that of W_i ($r = 2$), and v was set as the identity map ($p = 1,\ q = 0$).

The block matching algorithm with full search was used in step 3, the search range of s and t was set as $-R \le s,\ t \le R$, and the mean absolute distance was used as the measure of the difference. Here, the sampling value $g(w(\boldsymbol{x}))$ was calculated by averaging the four pixels around the sampling point $w(\boldsymbol{x})$.

In step 4, the sampling value of the alpha mask at $w_i(\boldsymbol{x})$ was detected as the majority value, 0 or 1, of the four pixels around the sampling point. If the numbers of 0s and 1s were equal to each other, the sampling value was set to 0.

A. Simple Shapes

Figs. 7 and 8 show the results for a simple test image including a sharp corner. The initial boundary $\boldsymbol{b}$ outside the object and W allocation are shown with white lines in Fig. 7(a). Fig. 8(a) shows a case in which $\boldsymbol{b}$ was set inside the object. The W allocation method B was used for these experiments. Therefore, S is the same in Figs. 7 and 8. The parameters for the experiments are shown in Table II.

The parts (b)–(e) in each of Figs. 7 and 8 show $\boldsymbol{b}$ for the initial state, one iteration, two iterations, and five iterations, respectively. W and the corresponding M are also shown in (b) and (c).

It can be seen in Fig. 7 that the texture of W (smaller block) in (c) is the same as that of M (larger block) in (b), and that the gap between $\boldsymbol{b}$ and $\boldsymbol{c}$ in (c) is half of that in (b). The gap was reduced with each successive mapping, and $\boldsymbol{b}$ finally matched $\boldsymbol{c}$. $\boldsymbol{b}$ did not change further after five iterations.

In Fig. 8, small background regions appeared in (c) because the right bottom corner of M in (b) was outside the foreground region of the alpha mask. However, the holes became smaller with each successive mapping iteration and eventually disappeared.

As shown in Fig. 7(e) and Fig. 8(e), $\boldsymbol{b}$ converges to a unique contour if $\boldsymbol{b}$ is set near $\boldsymbol{c}$ initially. When we set $\boldsymbol{b}$ deeply inside

TABLE II
PARAMETERS FOR EXPERIMENTS

	Figs. 7, 8, and 9	Fig. 10	Fig. 11	Fig. 12	Figs. 13 and 14
Image size	256 × 256	256 × 256	256 × 128	512 × 512	320 × 240
e_{max}	32	32	32	32	32
e_{min}	32	4	32	4	4
W allocation	Method B	Method B	Method C	Method C	Method C
R	64	32	16	16,8,4,2	16,8,4,2
η	5	5,4,3,2	5	5,4,3,2	5,4,3,2

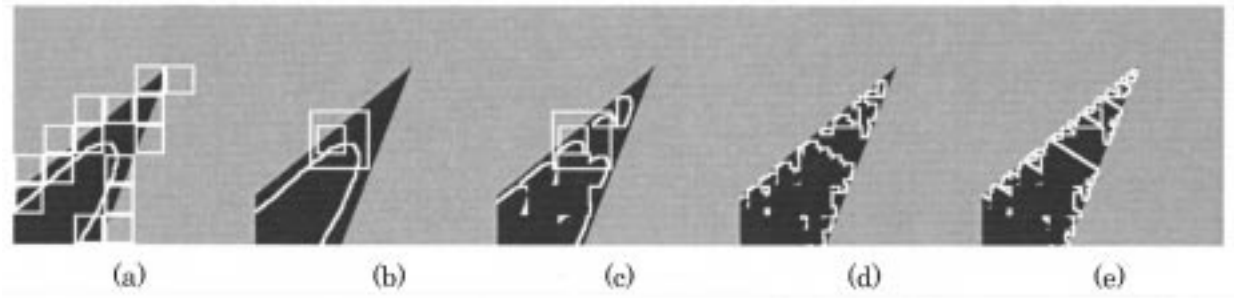

Fig. 9. Example of a failure by the proposed method. (a) W allocation. (b) Initial state. (c) One iteration. (d) Two iterations. (e) Five iterations. When $\boldsymbol{b}$ was set far from $\boldsymbol{c}$, $\boldsymbol{X}$ converged to a complicated fractal shape.

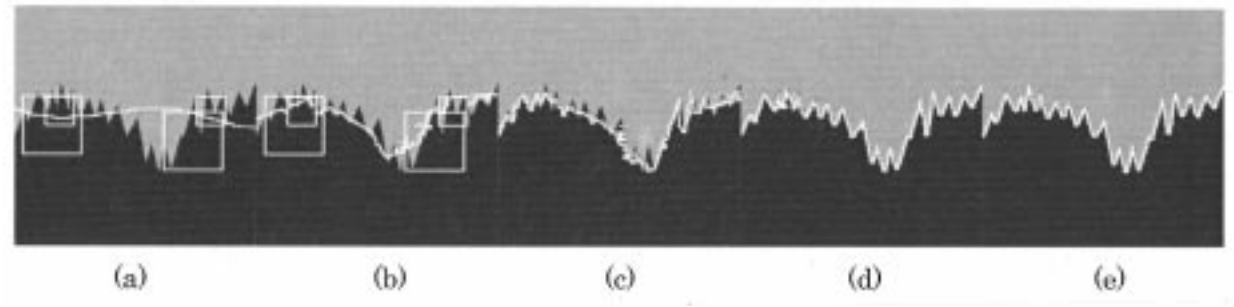

Fig. 10. Fitting to a zigzag shape. (a) Example of W and M. (b) $e = 32$. (c) $e = 32$ and 16. (d) $e = 32$, 16, and 8. (e) $e = 32$, 16, 8, and 4. Even large gaps between $\boldsymbol{b}$ and $\boldsymbol{c}$ could be fitted well, and the contour was extracted in detail by the recursive procedure in which the block size was progressively decreased.

the object, the lack of the foreground region became large and $\boldsymbol{X}$ converged to a complicated fractal shape as shown in Fig. 9. The proposed algorithm does not assure conservation of topology of the contour line. The authors cannot explain these phenomena analytically at this stage, and cannot show the rigorous boundary for failures. When the condition 3 shown in Section III-B is satisfied at least, the failure such as Fig. 9 can be avoided.

Fig. 10 shows the result for a zigzag image. e was changed from 32 to 4 and η was given by (25) for each e. Similar M were not found for large W. The case of $e = 32$ is shown in (a). $\boldsymbol{b}$ became like an envelope of $\boldsymbol{c}$ as shown in (b). The corners were produced in the process using smaller e. The result shows that a large W is useful for closing the gap between $\boldsymbol{b}$ and $\boldsymbol{c}$ when the gap is large, and a smaller W is useful for producing details of the contours.

The proposed method does not work well near parallel edges as shown in Fig. 11. Because the distance between edges in the original image was smaller than e of W, the block which provided similarity for one of the edges (the higher contrast edge) was detected as M. As a result, $\boldsymbol{b}$ was fitted to the left edge. There was no problem when the higher contrast edge was a part of the object contour. However, when there is a higher contrast edge parallel to the object contour and the gap between them is small, $\boldsymbol{b}$ will be fitted to the higher contrast edge instead of the object contour.

B. Real Images

The contour of the woman in the test image Lenna (512 × 512 pixels, monochrome, 8 bits/pixel) was extracted using the proposed method. The white line in Fig. 12(a) is roughly drawn by hand as an initial $\boldsymbol{b}$. The $e_{\max}$ and $e_{\min}$ were set to 32 and 4, respectively. The search ranges of s and t were provided by (23) for each e as shown in Table II.

The result of W allocation with $e = 32$ is shown in Fig. 12(b) and (c) shows the intermediate stages during the process and

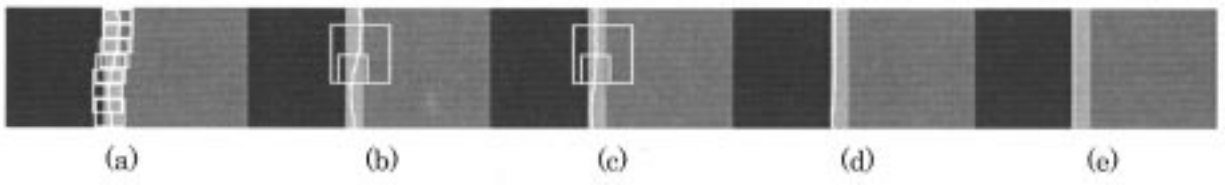

Fig. 11. Fitting to parallel edges. (a) W allocation. (b) Initial state. (c) One iteration. (d) Two iterations. (e) Five iterations. $\boldsymbol{b}$ was fitted to the higher contrast edge.

Fig. 12. Results using the test image Lenna. (a) Contour as drawn by hand. (b) Allocation of W. (c) and (d) Results of contour extraction. (c) $e = 32$. (d) $e = 32, 16, 8$, and 4.

corresponds to the completion of $e = 32$. The final result is shown in Fig. 12(d).

The corners between the brim and the body of the hat were not included in any W_i with $e = 32$; therefore the corners were not extracted as yet in Fig. 12(c). The corners and details of the contours were produced by smaller block sizes and almost all of $\boldsymbol{b}$ fitted with $\boldsymbol{c}$ finally. At the hair contour on the right side, an edge inside the hair was extracted wrongly for the reason explained in relation to Fig. 11.

The n-step search [16] was performed in addition to the full search for faster searching. The processing time using an MPU with a 350 MHz clock speed is shown in Table III. The n-step search was more than three times as fast as the full search. The extracted contour by the n-step search, whose figure is not shown, was almost the same as that by the full search [Fig. 12(d)].

We also examined a constant block size with $e = 4$. No fitting was achieved with only this block size.

Fig. 13. Fitting to sharp corners with distinct edges using a color image. (a) Initial state. (b) n-step search. (c) Full search.

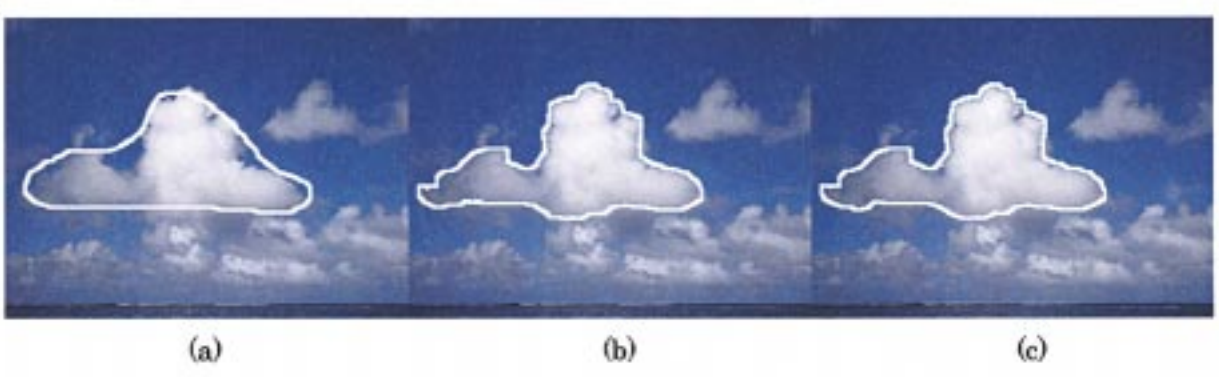

Fig. 14. Fitting to a blurred contour. (a) Initial state. (b) n-step search. (c) Full search. Both distinct and blurred contours were extracted well with the same parameters.

TABLE III
PROCESSING TIME

Image	Proposed method (msec)		Snake method
	n-step search	full search	(msec)
Lenna	800	2570	850
Tower	370	1500	390
Sky	440	2160	320

Figs. 13 and 14 show the results for color images we call Tower and Sky, respectively (320×240 pixels, RGB color, 24 bits/pixel). In the block matching step, the sum of the mean absolute distances of the three color components was used as the measure. The other procedures and the parameter values were the same as in the Lenna experiments.

As shown in Fig. 13, sharp corners were extracted accurately in the full search. The n-step search included some errors such as a missing rooftop at the center and extraction of a part of the background tower in the left part.

The cloud in Fig. 14 has a blurred contour on the left side. The contour was, however, extracted well using the same parameters as in the above experiments in both the n-step search and full search. The n-step search was about five times faster than the full search.

V. COMPARISON OF THE SELF-AFFINE MODEL AND SMOOTH CURVE MODEL

We proposed a self-affine model as an object contour model in this paper. A contour model is usually required for the extraction of edges as object contours. There are usually many edges of texture and noise other than $\boldsymbol{c}$ in both the foreground and the background except in ideal texture-free and noise-free images such as those used in Section IV-A. However, misextraction of the texture or noise as a contour is avoidable if a proper model is used as the constraint of the contour line.

Continuity and smoothness are used as energy function constraints in the snake method [3], [4]. This model works well for smooth curved contours. However, sharp corners of contours are difficult to extract without making the weight coefficient for smoothness small. Neglecting the smoothness may introduce misextraction of texture or noise.

The experimental results of contour extraction by the snake method are shown in Fig. 15. We implemented the time-delayed discrete dynamic programming algorithm [4], and parameters including the weight coefficients for continuity, smoothness and edge energy in the energy function were optimized for each image. Images on the left in Fig. 15 are the initial states, i.e., a rough shape was drawn by hand. The extracted contour by the snake method is shown on the right. The processing time is shown in Table III; there is little difference between the snake method and the proposed method with the n-step search.

Fig. 15. Contour extraction by the snake method. The images on the left are the initial contours drawn by hand, and the images on the right are the contour extracted images. The snake method failed to extract sharp corners as shown in (a), (b), and (d). The edges in background of (c) were misextracted. The blurred edges of the cloud in (e) could not be extracted.

Block matching required much time in the case of the proposed method, and the snake method needed much preprocessing, including smoothing of the original image and the edge energy image, before the minimization process.

As shown in Fig. 15(a), the top of the sharp corner was missed by the snake method, and only envelope is detected for the zigzag contour in (b) and (d), because of the constraint of smoothness. On the other hand, as shown in Figs. 7, 10, and 13, the self-affine model based on blockwise self-similarity can match the contour corners as long as each block contains only a single corner, because a single corner shape has self-similarity. The edges of the background on the right-hand side of Fig. 15(c) were misextracted. Those background edges which vertically cross the object contour did not disturb the extraction by the proposed method as can be seen in Fig. 12. The blurred contour of the cloud was extracted well by the proposed method as shown in Fig. 14. The snake method, however, could not extract the contour since the absolute value of the edge energy was so small that the line in Fig. 15(e) passed through the cloud contour in the process of minimizing the energy function.

A smooth curved line does not have exact self-similarity in ordinary cases, and the extracted contour by the proposed method is not perfectly smooth (the fractal dimension is greater than one) even though $\boldsymbol{c}$ is smooth. However, jagged edges did not

appear in the experiments because the amplitude of the roughness was smaller than the pixel interval.

The weakness of our method compared with the snake method is that the applicable contour gap is bounded with the block size as mentioned in Section III-B. In the case that the gap is too large, a combination of the two methods, in which the snake method is applied first and then the proposed method is applied, would provide a good result.

VI. Conclusions

The self-affine map was defined as an extension of LIFS, with the applicability of the mapping system extending to image processing including image segmentation, edge detection, and compression.

A new method for fitting of the alpha mask boundary to an object contour was proposed as a new application of the self-affine mapping system. It was shown that the proposed method produces not only smooth curves but also sharp corners and highly accurate details, and it is capable of extracting both distinct and blurred edges without requiring parameters to be changed. It was shown that even large gaps between the hand-drawn line and the contour could be fitted well by a recursive procedure in which the block size was progressively decreased. These features reduce the time required for drawing contours by hand.

Since the behavior of the self-affine map shown in Section II-C is unique, we expect many more applications related to image edges, contours, and regions to be developed in the future. Those would include edge adaptive filtering, object based processing, recognition, searching and so on.

Acknowledgment

The authors would like to thank R. Tokunaga, Assistant Professor, University of Tsukuba, for his valuable suggestions. The authors are also grateful to the reviewers for their many useful comments.

References

[1] N. Brady, "MPEG-4 standardized methods for the compression of arbitrarily shaped video objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 1170–1189, Dec. 1999.

[2] "Special issue on segmentation, description, and retrieval of video contents," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, Sept. 1998.

[3] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.

[4] A. Amini, T. Weymouth, and R. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Trans. Pattern Anal. Machine Intel.*, vol. 12, pp. 855–867, Sept. 1990.

[5] L. Cohen and I. Cohen, "Finite-element methods for active contour models and balloons for 2-D and 3-D images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1131–1147, Nov. 1993.

[6] B. Mandelbrot, *The Fractal Geometry of Nature*: W. H. Freeman and Company, 1982.

[7] M. Barnsley and L. Hurd, *Fractal Image Compression*. Wellesley, MA: Peters, 1993.

[8] A. Jacquin, "A novel fractal block-coding technique for digital images," in *Proc. ICASSP-90*, 1990, pp. 2225–2228.

[9] T. Ida and Y. Sambonsugi, "Image segmentation and contour detection using fractal coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 968–975, Dec. 1998.

[10] T. Ida, Y. Sambonsugi, and T. Watanabe, "Boundary fitting of extracted objects using LIFS" (in Japanese), *Trans. Inst. Electron., Inform., Commun. Eng. D-II*, vol. J82-D-II, pp. 1282–1289, Aug. 1999.

[11] T. Ida and Y. Sambonsugi, "Self-affine mapping system for object contour extraction," in *Proc. Int. Conf. Image Processing '99*, 1999, pp. 250–254.

[12] N. Lu, *Fractal Imaging*. New York: Academic, 1997.

[13] T. Ida and T. Kagoshima, "Image decoding of fractal coding using escape-time algorithm" (in Japanese), *Trans. Inst. Electron., Inform., Commun, Eng. D-II*, vol. J79-D-II, pp. 812–818, May 1996.

[14] M. Barnsley, *Fractals Everywhere*. New York: Academic, 1988.

[15] M. Barnsley and A. Jacquin, "Application of recurrent iterated function systems to images," in *Proc. SPIE Visual Communications Image Processing*, vol. 1001, 1988, pp. 122–131.

[16] R. Plompen, "Displacement compensated prediction," in *Motion Video Coding for Visual Telephony*: PTT Research Neher Laboratories, 1989, pp. 325–332.

Takashi Ida (M'95) received the B.Eng. and M.Eng. degrees in electrical engineering from Waseda University, Tokyo, Japan, in 1987 and 1989, respectively.

Since 1989, he has been with the Research and Development Center of Toshiba Corporation, Kawasaki, Japan. His research interests include image compression and image processing based on fractal theories.

Mr. Ida received the 1994 Young Engineer Award from the Institute of Electronics, Information and Communication Engineers (IEICE), Japan. He also received awards for young engineers from the Institute of Electrical Engineers of Japan (IEEJ) in 1994, and from the Japan Society for Fuzzy Theory and Systems (SOFT) in 1996, respectively. He is a member of IEICE.

Yoko Sambonsugi received the B. Eng. and the M. Eng. degrees in electrical and computer engineering from Yokohama National University, Yokohama, Japan, in 1990 and 1992 respectively.

Since 1992, she has been with the Research and Development Center of Toshiba Corporation, Kawasaki, Japan. Her current research interests include still/motion image processing, coding and segmentation.

Ms. Sambonsugi is a member of IEICE.