Texture-Based Image Retrieval Using Fractal Codes

Aidong Zhang[†], Biao Cheng[†], Raj Acharya[‡]

[†]Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260
{azhang, bcheng}@cs.buffalo.edu

[‡]Department of Electrical and Computer Engineering
State University of New York at Buffalo
Buffalo, NY 14260
acharya@eng.buffalo.edu

Abstract

This paper presents an approach to texture-based image retrieval which determines image similarity on the basis of the matching of fractal codes. Image fractal codes are generated via a fractal image compression technique that has been recently proposed as an effective image compression method. Each image is represented by a set of self-transformations through which an approximation of the original image can be reconstructed. These self-transformations, which are unique to each image and are semantically rich, are termed *fractal codes*. An image data model is proposed which constructs each image as a hierarchical structure. Each image is decomposed into block-based segments which are then assembled by a hierarchy on the basis of inclusion relationships. Each segment is then fractally encoded. The fractal codes of an iconic image are used as texture key and are matched with the fractal codes of images in a database by applying searching and matching algorithms to the hierarchies of the database images to locate the segments which best match the fractal codes of the iconic image. Retrievals of both exact and inexact matching of images are supported.

1 Introduction

Content-based image retrieval has been proposed to allow retrievals to be performed on the basis of a variety of aspects of image content [ACF+93, CYDA88, TPF+91, RS91, GS, HLHC92, BPJ93, WN94]. For example, we may wish to locate a set of images related to or a single image which most closely resembles a given iconic image. Such retrievals must use embedded content features, such as the shape, color, texture, layout, and position of various objects in an image. Texture in images has been recognized as important aspect of human visual perception [TMY78, WN94]. As discussed by Tamura et al. [TMY78] and Niblack et al. [NBE+93], main textural features include:

- *Coarseness*: measures the scale of the texture. For example, when two patterns differ only in scale, the magnified one is coarser.
- *Contrast*: measures the vividness of the texture and is a function of the variance of the gray-level histogram.
- *Directionality*: measures the "peakedness" of the distribution of gradient directions in the image. For example, the image may have a favored direction or is isotropic.

These textural features may be used to identify contents of images, such as clouds, ocean, trees, human faces and etc. Once each image is visually segmented into homogeneous textural regions, images can then be classified based on their textural features.

Textural features of images contain main visual characteristics of images that can be utilized in content-based image retrieval. Chang [SC94] has demonstrated an effective query-by-texture approach using quad-tree segmentation. However, visual texture segmentation in general remains to be a challenging problem. We observe that recent developments on fractal coding for image compression [BS88, FL92] provide an interesting approach to describing image texture content. Fractal codes of images can be utilized to avoid visual texture segmentation.

In fractal coding, an image is partitioned into a collection of non-overlapping regions termed *ranges*. For each such image range, another region termed *domain* and a transformation are chosen so that the domain best approximates the range. The code for the range consists of a geometrical description of the domain and the transformation. While the pixel data contained in the range and in the domain are used to determine the code, they are not part of the code itself. Thus, a high compression ratio can be achieved. The main characteristics of this approach are that it assumes that image redundancy can be captured and exploited through piecewise self-transformability [Jac93]. The method approximates an original image from a finite number of iterations of the image transformations called *fractal codes*. The self-transformation property used in this method to compress images may also significantly facilitate image retrieval if utilized to specify images and automatically index images by content. While this potential has been recently observed by Sloan [Slo94], research in this direction has not yet been pursued in detail.

In this paper, we shall investigate a solution to texture-based image retrieval by directly matching the fractal codes of the images without requiring visual texture segmentation. The fractal codes of iconic images are used as texture keys to locate relevant images in the database. Our approach includes an image data model, and searching and matching algorithms. The data model uses block-based decomposition on images. Each image is represented as a hierarchical structure (termed nona-tree) in which, at each descending level, an image segment is divided into nine subsegments of increasingly smaller size. Each subsegment represents a possible characteristic of the segment. A searching and matching strategy is then provided to locate the segment which best approximates the given iconic image. Given the fractal codes of an iconic image, this texture key is compared with the segments in the hierarchical structure of each image that are larger than or equal to the iconic image. Relevant database images are selected based on the given matching rate. Different matching rates are determined for both exact and inexact matching of images. Initial experiments conducted on natural images demonstrate that this approach not only greatly reduces image storage but also is effective in retrieving images by texture. We thus believe that fractal coding provides a promising approach to texture-based image retrieval.

The remainder of this paper is organized as follows. Section 2 presents our approach to the performance of image matching by fractal codes. We will introduce the main characteristics of fractal coding technique for image compression and discuss the significance of fractal coding in image retrieval. In Section 3, we propose a data model to decompose image data. Section 4 presents a searching and matching strategy to texture-based image retrieval. In Section 5, we set forth our initial experimental results conducted on natural images. Concluding remarks and future directions are offered in Section 6.

2 A Texture-Based Retrieval Approach

This section will present an approach to texture-based image retrieval based on fractal image compression.

2.1 Fractal Image Coding

A variety of image compression methods intended to greatly reduce storage space and promote efficient transmission have been extensively studied. Among these, a novel and promising approach called *fractal image compression* has recently drawn much attention. This approach is based on the concepts and mathematical results of iterated function systems (IFS). Barnsley and his co-workers at the Georgia Institute of Technology first recognized the potential of IFS for computer graphics applications. Jacquin [Jac92, Jac93] proposed the first fully automated algorithm for fractal image compression. This algorithm was based on affine transformations acting locally rather than globally. An image to be encoded is partitioned into non-overlapping range blocks. The task of a fractal encoder is to find a larger block of the same image (a domain block D) for every range block such that a transformation $W(D)$ of this block is a good approximation of the range block.

The transformation W is a combination of a geometrical transformation and luminance transformation. In matrix form, W can be expressed as follows:

$$W\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & s \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e \\ f \\ o \end{bmatrix}$$

where z denotes the pixel intensity at position (x, y), s is the contrast ratio, and o is the luminance offset.

Only the transformation of each range block and the position of the domain need to be transmitted to the decoder. This transformation code, iteratively applied to any initial image, will generate a simulated version of the original image. To ensure convergence at the decoder, the union of all transformations W must be contractive. Thus, the fractal compression approach is lossy [MAH94]. Since Jacquin's initial work on fractal block image coding, many extensions have been proposed to enhance the initial concept. Worthy of mention is the introduction by Fisher, Jacob and Boss [FJB91] of adaptive methods in the encoding process. Other researchers have focused on reducing the complexity of encoding or increasing decoding speed [LQR94].

Fractal coding offers a highly promising approach to image compression. Within only a short developmental period fractal coding has yielded results which rival the best examples of data compression by other methods. While it is a lossy method and is usually fairly slow to encode, the decoding process is simple, fast, and is independent of resolution. With a set of contractive transformations generated through fractal coding, the process of mapping domain images into range images can be approached iteratively, eventually converging to an approximate reconstruction of the original image. Each image can then be represented by its fractal codes in the database. Thus, fractal codes are sufficient for image representation in databases. Images can always be encoded off-line and the fractal codes decoded in real-time.

We will now discuss the significance of fractal coding in image retrieval by texture. We will demonstrate that images can be differentiated based on their fractal codes (or transformations). Consequently, the characteristics of an image can be identified by its fractal codes.

Consider the following three transformations that are given by Barnsley and Sloan [BS88]:

$$W_1\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$W_2\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0.25 \\ 0.5 \\ 0 \end{bmatrix}$$

$$W_3\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix}$$

Let us choose an initial image with the square $\{(x, y) : 0 \leq x \leq 1, 0 \leq y \leq 1\}$. The images generated from the first three iterations of the

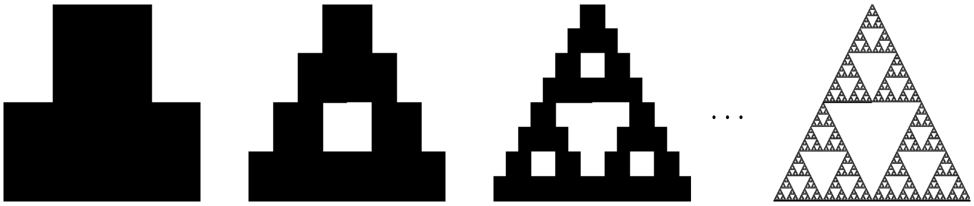

Figure 1: Image reconstruction.

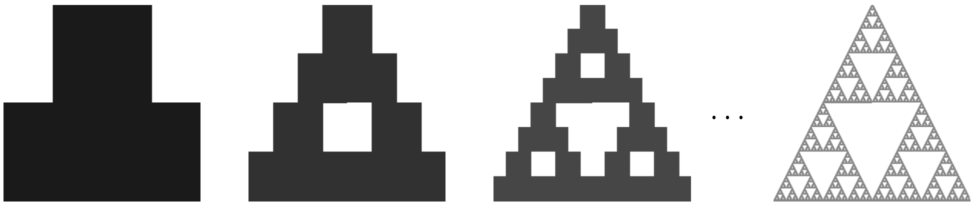

Figure 2: Image reconstruction.

above transformations and the converged image are shown in Figure 1. The variation of the contrast of this image can be shown by changing the value of s in the transformation. Figure 2 demonstrates an variant of Figure 1 when s is changed from 1 to 0.9.

We now slightly change the coefficients a and d in the above transformations to the following:

$$W_1 \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.51 & 0 & 0 \\ 0 & 0.51 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$W_2 \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.51 & 0 & 0 \\ 0 & 0.51 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0.25 \\ 0.5 \\ 0 \end{bmatrix}$$

$$W_3 \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.51 & 0 & 0 \\ 0 & 0.51 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix}$$

After repeating the same steps as those in Figure 1, we obtain a different image which is shown in Figure 3.

As a result, the fractal codes of images can be used to identify these images. More importantly, we observe that the fractal codes of images in some manner capture the textural features of the images. The transformations of the ranges in an image are generated by best mapping from domain blocks within the image to the ranges. Following this method, similar domains in similar images may be simultaneously chosen as the best approximation of the similar ranges in these images. Consequently, their fractal codes may be identical.

2.2 Fractal Codes Matching

Following the above discussion, we see that it is possible to uniquely determine the semantic contents of images using fractal transformations. We now discuss the application of this property to texture-based image retrieval. A common retrieval request to an image database system would involve finding all images in that database which contain a subimage that is similar to a given iconic image. The processing of such a retrieval request requires a matching strategy which can facilitate the identification of the iconic image as a constituent of (or closely similar to) images in the database.

We propose a fractal code matching strategy to determine the similarity of images. This matching approach does not require a procedure for carefully segmenting images based on textural contents. Thus, the challenging problems that are still faced in vision analysis can be avoided.

Fractal code matching involves comparing

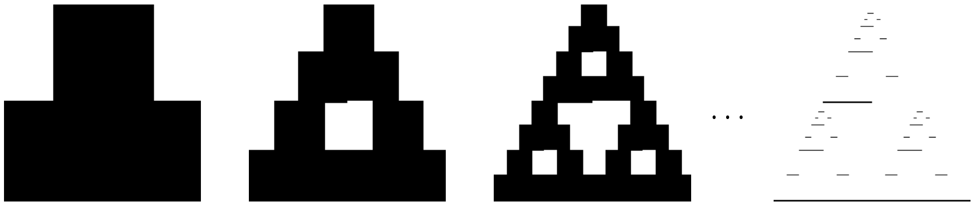

Figure 3: Image reconstruction.

range transformations of a pair of images. As a range in each image is best approximated by a domain that lies within that image, any relationships of overlap between the two images may play a significant role in matching. Such relationships include identity, major-part overlap, and minor-part overlap.

Let I_1 and I_2 be two identical images. Consider the situation in which images are partitioned into non-overlapping 4×4 pixel range blocks and these ranges are approximated by 8×8 pixel domain blocks. Following the definition of the fractal coding method, any two identical range blocks of I_1 and I_2 must have the identical domain blocks which provide the best self-transformation. Thus, their fractal codes must be identical, and, consequently, the fractal codes of both entire images are identical. As a result, a given iconic image can easily be matched with identical database images by matching their fractal codes.

Let I_1 and I_2 be two images and I_2 include I_1. The situation now becomes more complex if fractal code matching is used to find I_2 when given I_1. Let I_1' be the portion of I_2 that is identical with I_1. If fractal coding is performed on the images, a domain block that is not part of I_1' but is in I_2 may be chosen as the best self-transformation for a range block in I_1'. In such a situation, the fractal codes of I_1 may not be identical with the fractal codes of I_1'. Furthermore, let us consider a case in which I_1 and I_2 are two images which overlap but are not mutually inclusive. Let the overlaps be i_1 and i_2 in I_1 and I_2, respectively. We may still be interested in retrieving I_2 when given I_1. Similarly to the situation above, when fractal coding is performed on the entire images, a domain block that is not part of i_1 but is in I_1 may be chosen as the best self-transformation for a range block in i_1, and a domain block that is not part of i_2 but in I_2 may be chosen as the best self-transformation for the identical range block of i_2 in I_2. Thus, the two identical segments in I_1 and I_2 may have different best transformations, and, consequently, different fractal codes which render matching impossible.

Sloan [Slo94] has proposed the straightforward juxtaposition of an iconic image with each image in the image database. Synthetic images are then generated and encoded through fractal coding. The generated fractal codes are then analyzed. As the fractal encoder will approximate the ranges of the synthetic image by domains that lie within that synthetic image, an overlap between the iconic image and a portion of a particular database image will likely result in a range in the latter being best approximated by a domain in the former. The similarity of two images can thus be assessed by noting the frequent choice of a block in one image as the domain for another image. While this method is effective, the online encoding of iconic image with database images can be costly. For a 256×256 pixel image combined with a 64×64 pixel iconic image, the encoder must consider at least 64640[1] domains before it determines the best approximation for each range, and there are 4096 ranges. The cost thus renders this method impractical unless it is accompanied by specially-designed hardware such as a fractal compression accelerator board with a fractal compression chip.

We therefore wish to find an effective method of fractal code matching without resorting to the integrated encoding of database images and iconic

[1] 64640 is determined by $(256-8)^2 + (64-8)^2$.

images. As discussed above, the matching between one iconic image and a second which is embedded within a larger image may be unsatisfactory because the embedded image may potentially be represented by a larger selection of domain blocks. To solve this problem, we propose to decompose each image into various blocks of smaller size which will then be fractally encoded. Decomposition and the ensuing encoding will constrain the best approximation of each range to a more closely-related domain. Two ranges in identical segments are thus more likely to separately choose the identical best approximate domains and, as a result, to have identical transformations. Matching between the fractal codes of the iconic image and various segments in the database images is then performed. The detailed discussion of these topics is presented in the following two sections.

In the following discussions, we will denote the number of matched transformations between two images as the *matching rate* $\mathcal{R}$. The matching rate is used to measure the similarity of two images, with a higher matching rate indicating a greater degree of similarity.

3 Image Data Model

We shall now present a data model to be used for specifying image contents in the image database. Image contents include raster data (bitmaps) and fractal codes.

Image data must be organized in a fashion which supports the proposed matching-based retrieval of images by texture. We therefore propose to decompose each image into segments of various sizes before both encoding and matching are performed. As the identical segment to the iconic image may be located anywhere within the larger image, the decomposition must ensure that this segment image will not be completely lost after decomposition. Thus, the choice of an appropriate data model for decomposed images would be governed by the following factors:

(1) We should be able to efficiently access the fractal code of a small portion of the images;

(2) Images should be partitioned into segments of varying size so that a domain block can be chosen which is close to the corresponding range; and

(3) If a portion of a database image matches an iconic image, this portion should be captured as fully as possible in one of the decomposed blocks.

Following these considerations, we have developed a data model by which the image data can be accessed in a flexible manner. This data model organizes each image into a hierarchical structure which we term a *nona-tree*.

A nona-tree is a hierarchical data structure providing quick image data matching for data retrieval. A nona-tree is based on a slight modification of the principle of recursive decomposition of pictures that has been used in quadtrees [Cha89]. Each decomposition on an image segment produces nine subsegments of equal size. Figure 4 illustrates this decomposition strategy, which is described as follows:

- (segments 1,2,3,4 in Figure 4) four equal-sized quadrants;
- (segment 5 in Figure 4) the central area of the image block, of the same size as each quadrant; and
- (segments 6,7,8,9 in Figure 4) the central areas of the upper, bottom, left, and right halves of the image block, of the same size as each quadrant.

Further decompositions are carried out on only those segments which have non-single grey level pixels and for which the leaf nodes are restricted to certain size. The decomposition process can be described by a nona-tree with its root representing the entire image and its children representing the decomposed segments which, in turn, are roots for further decomposed segments. Each internal node has exactly nine children. Figure

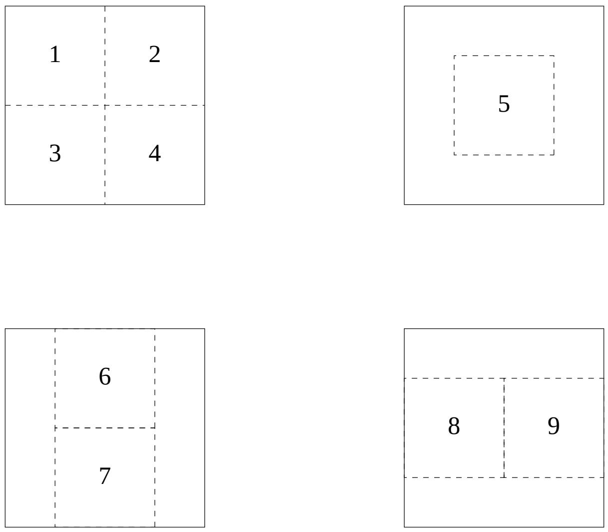

Figure 4: Block-based decomposition.

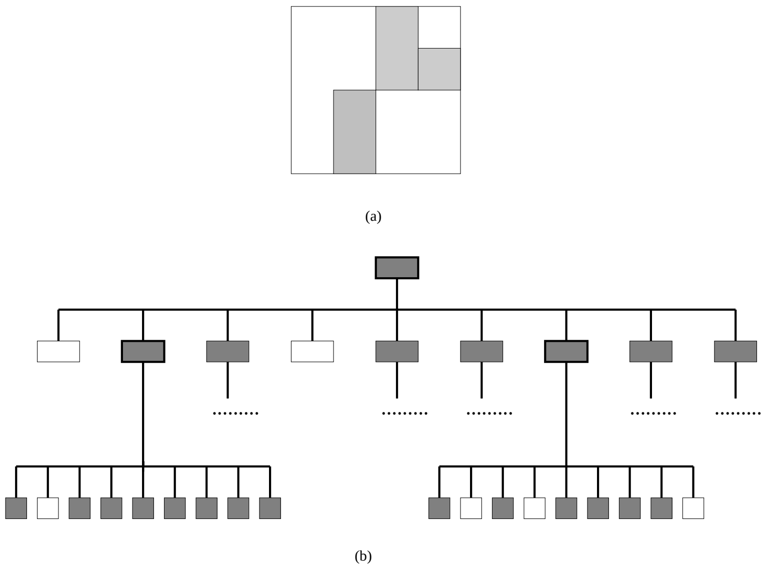

Figure 5: (a) Image; (b) Nona-tree.

5 illustrates the representation of an image by a nona-tree data structure.

The structure of a nona-tree, with nine subsegments in the arrangement described, is governed by factor (3) itemized above. Assume that the sizes of iconic images are always the same as that of segments in the nona-tree. If not, the iconic images can be tailored to meet the requirement. This will be discussed in more detail in the following section. The proposed method of decomposition ensures that, if a portion of a database image matches an iconic image, there then exists a segment which covers at least $\frac{9}{16}$ of the iconic image. Figure 6 illustrates the situation. In this example, we assume that the sizes of iconic images are compatible with that of the first level decomposition. In Figure 6 (a), the shaded area of the iconic image is $(\frac{3}{4}r)^2$, which is $\frac{9}{16}r^2$. This shaded area is covered by segment 1 defined in Figure 4 (also segments 5, 6 and 8). In general, as shown in Figure 6 (b), the area A of the iconic image that is covered by a segment defined in Figure 4 has the following cases:

- $r_1, r_2 \leq \frac{1}{4}r$: the area covered by segment 5 is:

$$A = (r - r_1) \times (r - r_2) \geq \frac{9}{16}r^2.$$

- $r_1 \leq \frac{1}{4}r$ and $r_2 \geq \frac{1}{4}r$: the area covered by segment 6 is:

$$A = (r - r_1) \times (\frac{1}{2}r + r_2) \geq \frac{9}{16}r^2.$$

- $r_1 \geq \frac{1}{4}r$ and $r_2 \leq \frac{1}{4}r$: the area covered by segment 8 is:

$$A = (r - r_2) \times (\frac{1}{2}r + r_1) \geq \frac{9}{16}r^2.$$

- $r_1, r_2 \geq \frac{1}{4}r$: the area covered by segment 1 is:

$$A = (\frac{1}{2}r + r_1) \times (\frac{1}{2}r + r_2) \geq \frac{9}{16}r^2.$$

This also holds when the iconic image is located in other places within the database image.

Fractal coding is applied to each node in the nona-tree. Thus, each node contains the fractal codes instead of the original raster data. These fractal codes are generated on the basis only of the raster data of the node rather than the entire image.

In practice, we limit the size of leaf blocks to a minimum of 64×64 pixels.

4 Searching and Matching Strategy

Let us now consider an image database in which all images are represented by their fractal codes in nona-trees. Let I be an iconic image consisting of an image portion which may represent an object, such as an anomaly in a medical image. Two typical retrieval requests would involve finding all images which contain or are similar to the iconic image. We now discuss procedures for matching the fractal codes of the iconic image with the relevant images in the database. A correspondence is first established between database image contents and fractal codes, and retrievals are then performed on the basis of the relationship between these fractal codes and iconic image I. We can then locate those images within a large image database which contain or are similar to the iconic image, as governed by set criteria.

4.1 Iconic Images

Let an image $\mathcal{M}$ contain iconic image I. There are two cases to be considered: (1) $\mathcal{M}$ is the same as I; or (2) I is a subimage of $\mathcal{M}$. In case (2), we assume that I can be anywhere in $\mathcal{M}$. Let both $\mathcal{M}$ and I be separately fractal-coded with 4×4 non-overlapping ranges and 8×8 domains. If the relationship between $\mathcal{M}$ and I follows case (1), then, following the principle of fractal coding, the transformations obtained in encoding $\mathcal{M}$ must be the same as those obtained in encoding I. Thus, we can determine that two images are identical on the basis of direct matching between the fractal codes of the two images. However, If

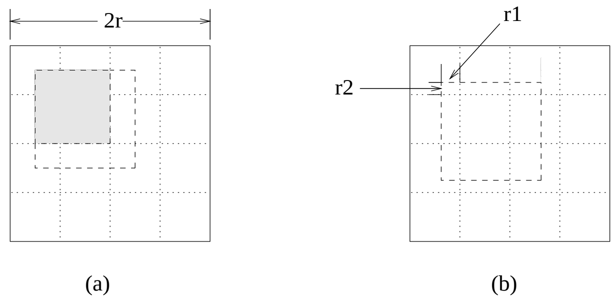

Figure 6: Covered area of iconic image by segments.

the relationship between $\mathcal{M}$ and I follows case (2), the situation will be far more complicated.

Consider that the pixel in the position of the first row and the first column in I should match the pixel in the position of the second row and the second column in $\mathcal{M}$. In this situation, each range in I overlaps with some ranges in $\mathcal{M}$ but may not match any range in $\mathcal{M}$. As a result, it is likely that the chosen domain which best approximates the range in $\mathcal{M}$ will be different from any of the domains chosen for the overlapping ranges in I. The transformations of the two images may thus have few matches. If the pixel in the position of the first row and the first column in I matches the pixel in the position of the fifth row and the fifth column in $\mathcal{M}$, then each range in I matches a range in $\mathcal{M}$, and both ranges may be best approximated by identical domains. Consequently, their transformations match.

To handle this situation, each iconic image is tailored to different cases before encoding. There are sixteen cases that must be considered:

I_{ij}: the first i pixel rows and the first j pixel columns are removed from I, where $0 \leq i, j \leq 3$.

I_{ij} ($0 \leq i, j \leq 3$) covers the possibility that, when i first rows and j first columns of pixels of I are removed, the ranges of I may be identical to some of the ranges in $\mathcal{M}$. We term these tailored cases of I *the variants of I.*

We assume that the size of an iconic image is always identical to the segments at certain level in the nona-tree. Iconic images may have to be tailored to meet this requirement.

4.2 Matching Rates

Let us consider the situation given in Figure 6 (b). If the iconic image is a subimage of the database image, there is then at least $\frac{9}{16}(\frac{r}{4})^2$ number of ranges which will definitely be covered by a segment in the nona-tree of the image. These covered ranges within both iconic image and database image may have identical fractal transformations. We define a *matching deviation* ε to be a positive integer. If there exists a segment within the image such that the matching rate $\mathcal{R}$ between the segment and the iconic image satisfies:

$$\frac{9}{16}(\frac{r}{4})^2 - \varepsilon \leq \mathcal{R} \leq (\frac{r}{4})^2,$$

we then consider the database image to be similar to the iconic image. The choice of matching deviation ε determines closeness of code matching between the iconic image and the retrieved database images. As ε becomes larger, the matching condition becomes weaker. As a result, more database images will be selected. However, some of them might be irrelevant to the retrieval request. Thus, ε should be decided based on the nature of applications and the expectation of users.

4.3 Algorithms

A relationship between a given iconic image I and any image $\mathcal{M}$ in the database is determined by comparing the fractal codes of the variants of I with different portions of $\mathcal{M}$. This can be done by traveling through the nona-tree of $\mathcal{M}$. The searching procedure designed below proceeds from the root to the bottom of the nona-tree as follows:

Algorithm 1 (Traversal searching) *Input:* $I_{ij}(0 \leq i, j \leq 3)$ *iconic images and one nona-tree* $\mathcal{T}$ *of image.*

1. *For each iconic image* $I \in \{I_{ij} : 0 \leq i, j \leq 3\}$*, I matches with the root of* $\mathcal{T}$*. If the matching rate is satisfactory, then report the result and stop.*
2. S = *the children of the root of* $\mathcal{T}$*.*
3. *For each iconic image* $I \in \{I_{ij} : 0 \leq i, j \leq 3\}$*, I matches with each element of S. If a satisfactory matching rate is obtained, then report the result and stop.*
4. *If any element of S has children and the size of the children is greater than or equal to the size of the iconic images, then S = the union of the children of all elements in S, go to step 3; otherwise, no matching is found, stop.*

The traversal searching algorithm may work well for some images. For example, if a segment containing the iconic image appears very different from other parts in the database image, most ranges in the segment would then have their best domain transformations within the segment. Consequently, the matching of the iconic image with the entire database image may still result in high matching rate. However, this algorithm may not be the most efficient one for other images. For example, if the ranges in the segment containing the iconic image have their best domain transformations outside of the segment, matching between the iconic image and its equal-sized segments in the database image will then generate the best matching rate. Thus, we also give another searching algorithm as follows:

Algorithm 2 (Direct searching) *Input:* $I_{ij}(0 \leq i, j \leq 3)$ *iconic images and one nona-tree* $\mathcal{T}$ *of image.*

1. S = *the children of* $\mathcal{T}$ *which have identical size with the iconic image.*
2. *For each iconic image* $I \in \{I_{ij} : 0 \leq i, j \leq 3\}$*, I matches with each element of S. If a satisfactory matching rate is obtained, then report the result and stop.*
3. *No matching is found, stop.*

We now discuss the process of matching images and the calculation of the matching rate. When an iconic image is compared with an image using traversal searching algorithm, it is possible that two completely different ranges with non-related domains may have identical transformations. Such matched cases should not be included in the calculation of the matching rate. Thus, the first step in this calculation is the elimination of these irrelevant matches, which can be accomplished by comparing range addresses among these matched ranges. A matching segment is first determined which includes the majority of matched ranges. A matched range which is remote from this major segment is considered to be an irrelevant match.

Algorithm 3 (Code matching) *Input: the fractal codes of two images.*

1. *Select those ranges in both images that have identical transformations.*
2. *Eliminate the irrelevant matches and calculate the matching rate.*

Note that item 2 in the code matching algorithm is not needed when the direct searching is applied.

We now compare the efficiency of this approach with Sloan's approach. Assume database images consist of $n \times n$ pixels and the depth of nona-trees be i. The maximum numbers of the segments that must be compared in Algorithms 1 and 2, respectively, are:

$N_1 = 1 + 9 + 9^2 + ... + 9^i$, and

$N_2 = 9^i$.

Each segment must be compared with sixteen variants of the given iconic image. The value of i depends on the difference between the size of the database image and that of the iconic image. In practice, i is usually between 1 and 4. For example, consider a database image of 256×256 pixels. If we limit the size of the leaf nodes to 64×64 pixels, i will then be 2. The maximum number of the segments in the nona-tree is $1 + 9 + 9^2$, which is 91. The total number of segment comparisons between the database image and the iconic image will be 16×91, which is 1456. As we have mentioned earlier, the total number of the segment comparisons that must be considered in Sloan's approach is:

$$((256-8)*(256-8)+(64-8)*(64-8))*(64*64) = 264,765,440.$$

This huge number of segment comparisons must be repeated for each database image to be compared. Therefore, our approach greatly reduces the number of segment comparisons that needs to be done.

5 Experiments

Experiments have been conducted on selected natural images in an image database. These experiments have been performed on SUN Sparc workstations using POSTGRES, an object-oriented database system developed at the University of California at Berkeley, and the HIPS Image Processing system developed at New York University.

Images are decomposed into segments of varying sizes and then converted to fractal codes in their nona-trees before addition to the database. These images are then classified into categories on the basis of their text descriptions in the database. A set of iconic images is chosen and converted to fractal codes. These iconic images are used as texture keys. Retrievals are processed for each iconic image to locate all images in the database of which it is a constituent.

Figure 7: Lena iconic image.

Figure 8: Lena image.

Let us consider an iconic image I_1 with 64×64 pixels taken from the famous Lena in Figure 7 and an image $\mathcal{M}_1$ of Lena in the database with 256×256 pixels in Figure 8. $\mathcal{M}_1$ contains I_1 as its subimage starting at the position of the 133th row and the 129th column. Matching I_1 with $\mathcal{M}_1$ results in only 10 relevant ranges with identical transformations. The best match between I_1 and the 128×128 pixel segments in the nona-tree of $\mathcal{M}_1$ is the subimage $\mathcal{M}_2$ starting at the position of the 129th row and the 129th column with the size 128×128 pixels. Matching I_1 with $\mathcal{M}_2$ results in 102 relevant ranges with identical transformations. The best match between I_1 and the 64×64 pixel segments in the nona-tree of $\mathcal{M}_1$ is the subimage $\mathcal{M}_3$ starting at the position of the 129th row and the 129th column with the size 64×64 pixels. Matching I_1 with $\mathcal{M}_3$ results in 199 relevant ranges with identical transformations. Note that this matching rate is much higher than $\frac{9}{16}(\frac{r}{4})^2$ which is 144.

We now consider another example. Let an iconic image with 64×64 pixels be in Figure 9 and the matching deviation be 40. The three closest matches shown in Figure 10 are returned to user

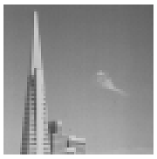

Figure 9: Iconic image.

using Algorithms 2 and 3. In fact, matching the iconic image with 256×256 pixels in Figure 10 (a), (b), and (c) results in only 11, 17, and 12 numbers of relevant ranges with identical transformations, respectively. The best matches between the iconic image with 128×128 pixel segments in Figure 10 (a), (b), and (c) are 36, 81, and 47 numbers of relevant ranges with identical transformations, respectively. The best matches between the iconic image with 64×64 pixel segments in Figure 10 (a), (b), and (c) are 121, 217, and 112 numbers of relevant ranges with identical transformations, respectively. The last three matches are satisfactory to the given matching deviation. All three images contain a segment which is similar to the iconic image but is not identical.

The experimental results are quite encouraging. Our experiments indicate that rates for both exact and inexact matches between an iconic image and an image in the database of which it is a constituent or closely similar are therefore feasible and reliable.

6 Conclusions and Future Research

In this paper, we have presented a novel approach to texture-based image retrieval. This approach provides an image data model and a matching strategy to compare images based on fractal codes generated using fractal image compression. As we have shown, the self-transformation property of fractal image compression offers the opportunity to distinguish one image from another based on their fractal codes. This process operates automatically, with no human assistance required to extract image content for retrieval.

Initial experimental results conducted on natural images have demonstrated this approach to be effective. However, many issues remain to be addressed, and we plan to continue this research in the following directions:

- Compare variants of the fractal image compression method and select an approach that offers the best similarity matching of identical images.
- Build a comprehensive indexing system for image databases based on the features offered in fractal codes.
- Build an interactive query language to support texture-based retrievals on the basis of these indexed image codes.

References

[ACF+93] Manish Arya, William Cody, Christos Faloutsos, Joel Richardson, and Arthur Toga. QBISM: A Prototype 3-D Medical Image Database System. *IEEE Data Engineering Bulletin*, 16(1):38–42, 1993.

[BPJ93] J.R. Bach, S. Paul, and R. Jain. A Visual Information Management System for the Interactive Retrieval of Faces. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):619–628, 1993.

[BS88] Michael F. Barnsley and Alan D. Sloan. A Better Way to Compress Images. *BYTE*, pages 215–223, 1988.

[Cha89] S.K. Chang. *Principles of Pictorial Information Systems Design*. Prentice Hall, 1989.

[CYDA88] S.K. Chang, C.W. Yan, Donald C. Dimitroff, and Timothy Arndt. An Intelligent Image Database System. *IEEE Transaction on Software Engineering*, 14(5):681–688, May 1988.

(a) (b) (c)

Figure 10: Closest matches comparing with Figure 9.

[FJB91] Y. Fisher, E.W. Jacobs, and R.D. Boss. Iterated Transformation image compression. Technical Report TR-1408, Naval Oceans Systems Center, San Diego, CA, 1991.

[FL92] Y. Fisher and A. F. Lawrance. Fractal Image Compression for mass storage applications. *SPIE Image Storage and Retrieval Systems*, 1662, 1992.

[GS] T. Gevers and A.W.M. Smeulders. *An Approach to Image Retrieval for Image Databases*, volume 720. Lecture Notes in Computer Science, Springer-Verlag, Berlin.

[HLHC92] Tai-Yuan Hou, Peiya Liu, Arding Hsu, and Ming-Yee Chiu. Medical Image Retrieval by Spatial Features. In *IEEE Conference on Systems, Man, and Cybernetics*, 1992.

[Jac92] Arnaud E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transactions on Image Processing*, 1(1), 1992.

[Jac93] Arnaud E. Jacquin. Fractal Image Coding: A Review. *Proceedings of the IEEE*, 81(10), 1993.

[LQR94] S. Lepsqy, G.E. Qien, and T.A. Ramstad. Attractor Image Compression with a Fast non-iterative decoding Algorithm. In *Proceedings of ICASSP-93*, 1994.

[MAH94] M.Gharavi-Alkhansari and T.S. Huang. Fractal-Based Techniques for a Generalized Image Coding Method. In *Proceedings IEEE ICIP*, 1994.

[NBE+93] W. Niblack, R. Barker, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The qbic project: Querying images by content using color, texture, and shape. Technical report, IBM Technical Report, 1993.

[RS91] F. Rabitti and P. Savino. Automatic Image Indexation and Retrieval. In *Conference on Intelligent Text and Image Handling*, 1991.

[SC94] John R. Smith and Shih-Fu Chang. Quad-Tree Segmentation for Texture-Based Image Query. In *Proceedings of ACM Multimedia 94*, pages 279–286, San Francisco, California, October 1994.

[Slo94] Alan D. Sloan. Retrieving Database Contents by Image Recognition: New

Fractal Power. *Advanced Imaging*, 9(5):26–30, 5 1994.

[TMY78] Hideyuki Tamura, Shunji Mori, and Takashi Yamawaki. Texture Features Corresponding to Visual Perception. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(6), June 1978.

[TPF+91] A. Turtur, F. Prampolini, M. Fantini, R. Guarda, and M.A. Imperato. IDB: An Image Database System. *IBM Journal of Research and Development*, 35(1):88–96, January 1991.

[WN94] Jian Kang Wu and Arcot Desai Narasimhalu. Identifying Faces Using Mutiple Retrievals. *IEEE Multimedia*, 1(2):27–38, 1994.