IMAGE DATABASES, SCALE AND FRACTAL TRANSFORMS

Ben A.M. Schouten and Paul M. de Zeeuw

Centre for Mathematics and Computer Sciences (CWI)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{B.A.M.Schouten, Paul.de.Zeeuw}@cwi.nl

ABSTRACT

In contemporary image databases one finds many images with the same image content but perturbed by zooming, scaling, rotation etc. For the purpose of image recognition in such databases we employ features based on statistics stemming from fractal transforms of gray-scale images. We show how the features derived from these statistical aspects can be made invariant to zooming or rescaling. A feature invariance measure is defined and described. The method is especially suitable for images of textures. We produce numerical results which validate the approach.

1. INTRODUCTION

For the purpose of image recognition we are after feature invariance when images are either zoomed in or zoomed out. The operation of zooming-in can be seen as cropping followed by up-scaling (see e.g. Figure 1). The operation of zooming-out involves the addition of new information, followed by down-scaling. Therefore, feature invariance appears not feasible at all for zooming-out. However, in the special case of texture-images the additional information is similar to the information already present (see e.g. Figure 4). In this paper we consider feature invariance for zoomed textile images.

The fractal transform of an image consists of a Partial Iterated Function System (PIFS). In a PIFS, the domain for every function in the system varies and is a part of the image itself. The number of functions in the system is large, typically hundreds. In this paper we examine the relationship of the statistical properties of the fractal functions in the system before and after zooming. Such a relationship can be used to create fractal features, invariant under scaling. The paper is concerned with the use of fractal transformations as feature extractors [1, 2, 3, 4, 5, 6].

After this introduction, Section 2 briefly presents the basics of fractal feature extraction, including fractal coding schemes. The choice of our features is explained in Section 3. In Section 4 we introduce a method to make the features less invariant to zooming and present results accordingly.

B.S. gratefully acknowledges support by NWO, The Netherlands.

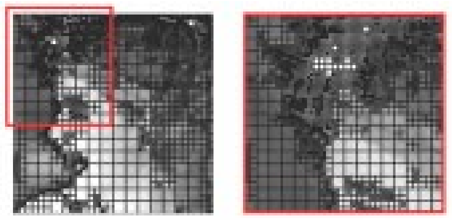

Fig. 1. Zooming in at Teeny Image.

2. FRACTAL FEATURES

For completeness we give a brief description of fractal image compression (FIC) [7]. Most of the feature extraction methods are based on the parameters used in FIC.

A given image is partitioned into non-overlapping range blocks, see Figure 1. The fractal encoder searches for parts called domain-blocks (which can be larger and overlapping) in the same image that look similar under some fixed number of affine transformations. Such an affine transformation can be written as:

$$t_i(\vec{x}) = A_i\vec{x} + \vec{o}, \quad A_i \equiv \begin{pmatrix} a_{11} & b_{12} & 0 \\ c_{21} & d_{22} & 0 \\ 0 & 0 & u_i \end{pmatrix}, \quad \text{(1a)}$$

$$\vec{x} \equiv \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix}, \qquad \vec{o} \equiv \begin{pmatrix} e_x \\ f_x \\ o_i \end{pmatrix}. \quad \text{(1b)}$$

Index i indicates the range-blocks within the image, $f(x,y)$ denotes the gray-value at position (x,y). u_i is the contrast scaling and o_i is the luminance offset. The u_i and o_i are used to match the gray-values of the domain with the gray-values of the range-block, within the limits of an imposed accuracy ϵ. Usually, a domain-block has twice the size of a range-block. The contractive nature of the transformations t_i makes the fractal encoder work. The transformation $T =$

$\bigcup_{i=1}^{N} t_i$ (where N is the total number of range blocks in the image) has a fixed point which approximates the original image. It can be restored by iterating T in the decoding phase starting with an arbitrary given image.

2.1. Features and invariances

Most of the known fractal feature extractors use the parameters, discussed in the previous section, to describe the image or object [1, 2, 3, 4, 5, 6, 8].

There is a major drawback in using fractal transformations for feature extraction. The same image (attractor) can be the result of two totally different fractal transformations, making it hard to compare two images. We proposed statistical analysis of the fractal parameters [8], assuming that well-chosen statistics of the different fractal transforms remain invariant. We strive for invariance with respect to a range of perturbations that occur in contemporary multimedia databases, like rotations, shifts, brightness adjustments [9]. In this paper, in the context of textile, zooming is considered. In the literature no such invariance is found.

3. THE FEATURES

3.1. Introduction

Here we give an outline of the features we employ, see also [8, 9]. Most of the existing fractal coding schemes use a quad-tree structure as a subdivision of the image, see Figure 2. For a given accuracy ϵ (see Section 2), the algorithm

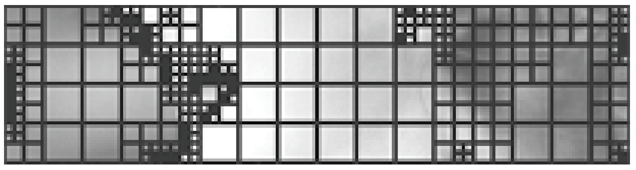

Fig. 2. Detail of Figure 1, four depths i of the quad-tree are shown.

finds a matching domain-block for the range-block in question. This is called a *success*. If there is no satisfactory match, the range-block splits into four equal parts. In this way several *depths i* of the quad-tree are created, containing range-blocks of the same size, see Figure 2. We now introduce several feature histograms. Let L be the integer signifying the maximum depth imposed in the (fractal) decomposition with quad-tree refinement, likewise l signifies the minimum depth. A domain $\Omega_{l,L;k}$ is defined as:

$$\Omega_{l,L;k} \equiv \left\{(i,j) \in \mathbb{N}^2 \mid l \leq i \leq L, 1 \leq j \leq k\right\} \qquad (2)$$

where i is associated with the depth in the quad-tree structure and k is the chosen number of *feature-bins*, see Section 3.2. A histogram h on $\Omega_{l,L;k}$ is defined as a function

$$h: \quad \Omega_{l,L;k} \to \mathbb{R}, \quad \text{with} \quad h \geq 0. \qquad (3)$$

If $(i,j) \in \Omega_{l,L;k}$ then $h_{ij} = h(i,j)$ is called the *value* of h at (i,j). A histogram h on $\Omega_{l,L;k}$ is called a *(weighted) quad-tree feature histogram* if it satisfies the following additional requirements:

$$h_{ij} = w_i v_{ij}, \qquad (4)$$

where

$$\sum_{i=l}^{L} w_i \leq 1; \quad w_i \geq 0 \qquad (5)$$

and

$$v_{ij} \geq 0; \quad \sum_{j=1}^{k} v_{ij} = 1, \quad \forall i \in \mathbb{N} \quad \text{with} \quad l \leq i \leq L. \quad (6)$$

Requirement (6) can be interpreted in the way that at each depth i we have k bins v_{ij} of which the contents add up to 1. Requirement (4)–(5) can be interpreted as weighing the contents of the bins depending on depth i. For an interpretation of the bins see Section 3.2 (next).

3.2. Description of the feature-bins

We describe two different fractal image features to recognize a texture: coarseness and contrast (we can think of more, see [9]). The definition of the second involves the first.

1. *Coarseness Feature.* At each level i in the quad-tree we record the fraction w_i of the images area that is matched by the fractal decomposition (success). These fractions are the weights in (4). In case that an image has been fully resolved by fractal decomposition then the "$\leq$" in (5) turns into an "=" sign. The w_i together ($l \leq i \leq L$) constitute a quad-tree feature histogram with $k = 1$ bins.

2. *Contrast Feature.* To match the gray-values of the range-blocks by the gray-values of the domain-blocks, a scaling factor u_i is used, see (1):

$$(t_i(\vec{x}))_3 = u_i \cdot f(x,y) + o_i, \quad 0 < ||u_i|| < 1.$$

   The range of this scaling factor is divided into 8 intervals, which leads to $k = 8$ feature-bins for this feature. Intuitively, the feature relates to the homogeneity of the gray-values within the image.

Figure 3 gives an example of a typical quad-tree feature histogram.

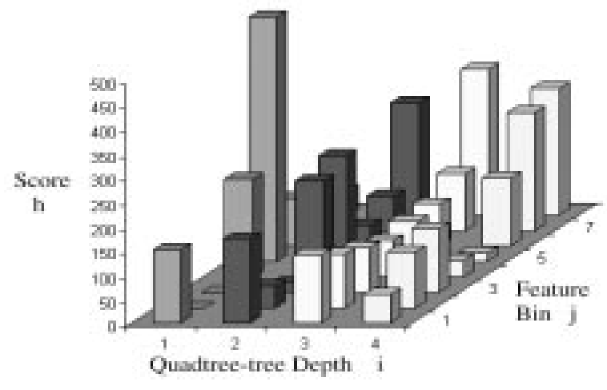

Fig. 3. Two-dimensional quad-tree feature histogram (contrast feature).

4. EXPERIMENTS, RESULTS AND DISCUSSION

Fractal feature extractors have been shown before to be effective for indexing multimedia database consisting of texture images [3, 8, 9, 10]. In [9] we demonstrated the invariance of the features, including those from Section 3.2 for rotation, translation, folding and brightness adjustments. Here we show that if an image is altered by zooming, the feature still distinguishes between images, especially if we allow the histogram to shift along the axis of the quad-tree depth before comparison.

4.1. Method

Each perturbed image is compared to all members of the database. Below we employ an invariance measure for features with respect to a database D. Let a database D count N images q:

$$q_i \in D, i = 1, \dots, N.$$

Let $p^{(n)}$ be a perturbation: an operator that perturbs an image q into an image $p^{(n)}(q)$. The collection of all perturbations of the images, is denoted by P:

$$p^{(n)} \in P, n = 1, \dots, M.$$

A quad-tree feature histogram can be interpreted as a point in $\mathbb{R}^n$ with $n \equiv k(L - l + 1)$. The distance d between two quad-tree feature histograms is defined as the 2-norm of their distance in $\mathbb{R}^n$. So $d(h(q_i), h(q_j))$ denotes the distance between the feature histograms of images q_i and q_j. We denote

$$d_{ij} = d(h(p(q_i)), h(q_j)) \in \mathbb{R}.$$

We introduce a measure for feature invariance as follows. Firstly, for an entry q_j from the database D we compute d_{ij}, $i = 1, \cdots, N$ for a given perturbation $p \in P$. Secondly, we list d_{ij} in order of decreasing size. Thirdly, let $r_j = r(q_j, D)$ be the ranking number of d_{jj} in the list. That is, $r_j = 0$ is the best possible result and $r_j = N - 1$ the worst possible. We now define the *absolute feature invariance measure* (AFIM) with respect to the database D and perturbation $p \in P$:

$$\nu(D, p) = \left(1 - \frac{\sum_{j=1}^{N} r_j}{(N-1)N}\right) \cdot 100. \qquad (7)$$

If $\nu = 100$ this implies that all queries (perturbed images) are recognized without fault by the feature.

4.2. Results

As for perturbation of an image we confine ourselves to zooming procedures. In Table 1 we present the results for the feature invariance measure with respect to the database ($N = 52$). Columns 2 & 4 (No Shift) correspond to the straightforward computation according to (7). Columns 3 & 5 (Shift $\leq \frac{1}{2}$) correspond to the case that the distance between two histograms $d(h(q'), h(q))$ is not computed as is but as

Table 1. Absolute feature invariance measure.

p	coarseness feature		contrast feature	
zoom	No Shift	Shift $\leq \frac{1}{2}$	No Shift	Shift $\leq \frac{1}{2}$
out	89.0	93.7	97.4	97.6
in	82.3	93.0	93.6	98.1

$$\min_{T \in \mathcal{T}} d(h(q'), T(h(q)))$$

instead, where T represents an operator that shifts a histogram along the axis of the quad-tree depth. We allow a histogram to shift over a maximum distance of $1/2$ in both the positive and negative direction (this defines $\mathcal{T}$). This corresponds to a zooming factor $2^{1/2}$ (in and out) of an image. We observe from Table 1 that the feature invariance benefits from taking the above shifts into account. The contrast feature appears to be very robust, even so without the shifting technique.

4.3. Discussion

In this paper, fractal transforms are employed with the aim of image recognition in multimedia-databases. We use features based on statistics stemming from fractal decomposition of images. We demonstrate that feature invariance with respect to zooming benefits from shifting the feature histogram. The thought behind our method is that the scales

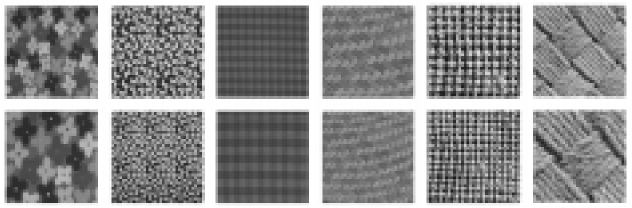

Fig. 4. Examples from the database (excerpts from VISTEX and Brodatz Collection) and the zoomed versions.

of texture in an image match with quad-tree depths in the fractal decomposition, and determine the outcome of the decomposition at each depth accordingly. That's why statistical aspects from an image move from one (quad-tree) depth to another when an image is zoomed, see Figure 5. This effect is compensated for by shifting the histogram into the opposite direction.

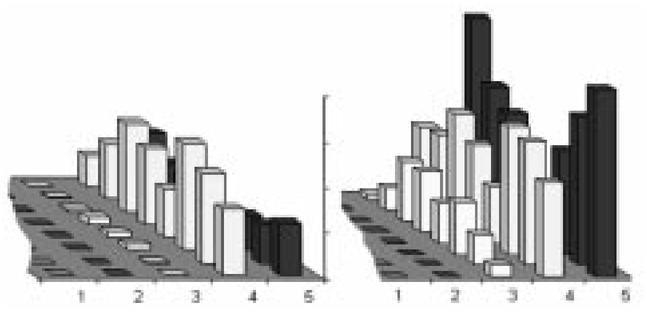

Fig. 5. Quad-tree histograms of original image (left) and zoomed-in image (right), $l = 1$, $L = 5$.

We finish up with an example, see Figure 6. The image Cloth139 (left) is presented to the database. Without compensating for zooming effects, the image Brick.000 (middle) is retrieved. The scale of both textures is confusingly similar. However, our shifting technique compensates for such effects and the original image (right) is retrieved. The features as described relate well to human perception [8, 9] and will be used for visual intelligence retrieval systems [10].

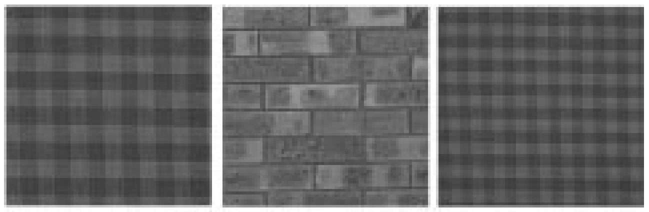

Fig. 6. An image (left) is presented to the database. Without compensating for zooming a confusingly similar image (middle) is retrieved. After compensating the correct image (right) is retrieved.

5. REFERENCES

[1] M. Baldoni, C. Baroglio, D. Cavagnino, and L. Egidi, *Learning to Classify Images by Means of Iterated Function Systems*, pp. 173–182, World Scientific Publishing Co. Pte. Ltd, Singapore, 1998.

[2] A.Z. Kouzani, F. He, and K. Samut, "Fractal face representation and recognition," in *IEEE International Conference on Systems, Man and Cybernetics*, 1997, pp. 1609–1613.

[3] J.M. Marie-Julie and H. Essafi, "Image database indexing and retrieval using the fractal transform," in *ECMAST '97; Multimedia Applications, Services and Techniques*. 1997, pp. 483–492, Springer.

[4] G. Neil and K.M. Curtis, "Scale and rotationally invariant object recognition using fractal transformations," in *ICASSP; Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1996, vol. 6, pp. 3458–3461.

[5] T. Tan and H. Yan, "Face recognition by fractal transformations," in *ICASSP; Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999, vol. 6, pp. 3537–3540.

[6] M. Vissac, J. Dugelay, and K. Rose, "A fractals inspired approach to content-based image indexing," in *ICIP; Proceedings IEEE International Conference on Image Processing*, 1999, p. CDROM.

[7] Y. Fisher, *Fractal Image Compression: Theory and Application*, Springer-Verlag, 1995.

[8] B.A.M. Schouten and P.M. de Zeeuw, "Feature extraction using fractal codes," in *Visual 99; Visual Information and Information Systems, Third International Conference*. 1999, pp. 483–492, Springer.

[9] B.A.M. Schouten and P.M. de Zeeuw, "Fractal transforms and feature invariance," in *ICPR 2000; Proceedings International Conference on Pattern Recognition*, 2000.

[10] G. Frederix, A.A.M Kuijk, E.J. Pauwels, and B.A.M. Schouten, "Show me what you mean!, *pariss*: A cbir-interface that learns by example," in *Submitted for Visual 2000; Fourth Conference on Visual Information Systems*, 2000.