Computer Vision and Image Understanding **77,** 371–387 (2000)
doi:10.1006/cviu.1999.0823, available online at http://www.idealibrary.com on IDEAL®

Use of IFS Codes for Learning 2D Isolated-Object Classification Systems

Matteo Baldoni, Cristina Baroglio, and Davide Cavagnino

Dipartimento di Informatica, Università degli Studi di Torino, Corso Svizzera, 185, 1-10149 Torino, Italy
E-mail: baldoni@di.unito.it; baroglio@di.unito.it; davide@di.unito.it

Received May 21, 1999; accepted November 29, 1999

Automatic recognition of complex images is a hard and computationally expensive task, mainly because it is extremely difficult to capture in an automatic way and with a few features the necessary discriminant information. If such features were available, a proper learning system could be trained to distinguish images of different kinds of objects, starting from a set of labeled examples. In this paper we show that fractal features obtained from Iterated Function System encodings capture the kind of information that is needed by learning systems and, thus, allow the successful classification of 2-dimensional images of objects. We also present a fractal feature extraction algorithm and report the classification results obtained on two very different test-beds by applying Machine Learning techniques to sets of encoded images. © 2000 Academic Press

1. INTRODUCTION

One of the most challenging research topics currently being investigated is the *automatic recognition* of objects from their visual representation, a problem of great importance due to the many repercussions automatic recognition systems would have in different applicative fields. For instance, image recognition systems could be exploited to implement *graphical interfaces*, where a human operator draws sketches that the computer will interpret. So, for example, an operator could use a magnetic pen to write algebraic formulas, whose components are automatically recognized by the computer [36]. Similar interfaces would allow the development of easier-to-use CAD systems that would—in turn—simplify design tasks.

More importantly, automatic recognition tools may be used to develop innovative indexing systems and retrieval engines for large repositories of images and documents [32]. In the case of image databases this sort of system would allow the retrieval of images containing instances of a given class of objects rather than retrieving those images that contain a specific object (see, for instance, [14])—e.g., images containing “tulips” but not a specific

1077-3142/00 $35.00
Copyright © 2000 by Academic Press
All rights of reproduction in any form reserved.

one. This can be done because by using automatic learning tools it is possible to acquire generalized descriptions of objects that fit a wide variety of cases.

According to O'Gorman [34], recent interest in the use of automatic recognition tools has been shown also by the *image compression* community (see, for instance, [47]). The reason is that very few further improvements can be gained by the current signal processing approaches: if recognition techniques could be integrated with compression methods, in some cases it would be possible to automatically identify image components and, then, store (or transmit) them by storing (transmitting) their identifiers and their deviation from a canonical example.

One of the problems that plays a central role in the applications mentioned above is the automatic classification of *isolated objects*, not an easy task. As an illustration, let us consider a gray-level image made of 256×320 pixels. Working at a raw data level, our system should be able to handle $81,920$ inputs that are not correlated to each other. Of course, since the smaller the number of input features the more tractable the classification task, it is better to work at a *more abstract* level of detail, *extracting* from the raw data the most meaningful characteristics of the represented object. These characteristics usually correspond to properties of parts of the image or of the image as a whole (colors, brightness, and many others) and to relations between the different parts (position, similarity, and so forth).

In other words, it is necessary to capture the discriminant information that is contained in the image by means of a *few* features. The ideal features one would like to use are *easy* to extract, *robust*, and *well-suited* to a *fast* and *reliable interpretation*. They should also be *invariant* w.r.t. the image size and have a *high discriminatory power*, i.e., be different for different classes of objects. Well-known approaches to the problem suggest using template matching [42, 1] or model-based methods [12, 15, 22]. In our work we have investigated a simpler approach that does not need any such knowledge, being based on the direct extraction of a set of characterizing features. Such features are derived from a fractal description of an image by means of an *Iterated Function System* (IFS).

Image encoding by means of an IFS is well known in the literature. It has been, first, proposed by Barnsley [10] for image transmission. Since then, it has been widely used for image compression and for efficient browsing of images using thumbnails [17]. Nevertheless, to our knowledge, IFSs were never used for classification purposes, and there are very few papers that are related to this topic [21, 18, 23]. A description of these works and their differences from the method we propose can be found in Section 5.

The main purpose of this paper is to give a unified view of the results of our research, which have already been presented separately [5, 2], and to draw some general conclusions. More specifically, we want to show that it is possible to represent an isolated object's *characteristic information*, i.e., the information that allows one to recognize it as belonging to some class of objects (mugs, flowers, oaks, etc.), by means of an IFS. We will empirically show that not all IFS encoding methods are good for capturing this kind of information, but when the right ones are used, the encodings of the images of a set of objects of the same class are actually *more similar* than those of objects of different classes. Last but not least, we will describe a fractal feature extraction algorithm (based on the original formulation of the *Collage Theorem* [7]) that we designed and implemented to extract characteristic IFSs in an *automatic way*. Given a system like the one that we propose, one can represent a set of labeled instances by means of a few features and, then, train a *Machine Learning* tool to distinguish between objects belonging to different classes. The task of building a recognition system is, then, turned into the task of learning a classifier. Note that both the feature extraction step (that corresponds to IFS-based encoding) and the building of the recognition system (that in our case corresponds to training a learning system) are automatic. This is, in our opinion, the real novelty and advantage of the proposed approach: the compactness of representation given by IFSs allows automatic learning systems to be used for avoiding the prior creation of prototype models of the relevant classes, which is a critical step in the current approach to image classification. On the other hand, learning systems are used as a means for producing approximate solutions to a generalized version of the *Inverse Problem* of IFS, i.e., find the IFS whose perturbations produce the instances of a given class of images (see Section 2).

The system that we implemented, *XFF*, was tested on two very different classification tasks: the first consists of the recognition of instances of three different species of *trees*, while the second consists of the recognition of handwritten instances of *ten digits*. Tree classification is aimed at testing the effectiveness of our feature extraction algorithm in capturing discriminatory information about shapes. It is quite a hard task for different reasons. First of all, trees have *complex shapes* that are very difficult to represent by means of a small set of features (think, for instance, of the difficulty of identifying the edges of every branch). In particular, we used three species (oak, lime-tree, and elm), whose shapes are pretty similar, as one can see in Fig. 4. Another—already mentioned—difficulty resides in the size of the images, which are made of 320×240 pixels.

On the other hand, the digit classification problem is aimed at testing the generality of the method on images which do not have an intrinsic complex structure but, at the same time, do not seem to be easily representable by means of fractal features. Digits are, in fact, *linear*, one-dimensional patterns, spread across a two-dimensional region (the surface on which the digit is written). The digits test-bed was chosen also because it is well known in the Machine Learning literature [27] and allows a comparison with the performances achieved by other approaches. Last but not least, this is an optical character recognition task, in which real-world data are used.

This paper is organized as follows. In Section 2 the basic notions concerning IFSs (with special attention to the Collage Theorem), fractal features, and their differences from other feature extraction methods are provided. In Section 3 we describe XFF, the system for the eXtraction of Fractal Features that we developed, and explain its theoretical background. Then, in Section 4 we describe the two learning problems and the experiments we conducted, reporting the results. Finally, Section 5 comments on the experimental results and reports some points that we intend to investigate in the future.

2. ITERATED FUNCTION SYSTEMS AND IMAGE REPRESENTATION

After the work of Mandelbrot [26], fractals and IFS have been widely used in compression, see for instance [10, 24, 29, 9, 43, 11, 20, 6, 46], and for representing natural shapes [38]. The basic idea underlying *fractals* is that a fractal is at any scale *self-similar*. One way of generating fractal images in the plane is to iterate a *contractive* geometrical transformation: the fractal is the limit figure obtained when the number of iterations goes to infinity. In principle, the mapping between the original image and each of the smaller copies can be any transformation; in practice *affine* transformations are preferred. More specifically, let us consider the bidimensional space $\mathbf{R}^2$ (the case of interest when dealing with black and white 2-D images) and let $\vec{x}$ be a point in $\mathbf{R}^2$. An affine transformation has the general form:

$$M(\vec{x}) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \vec{x} + \begin{bmatrix} e \\ f \end{bmatrix}, \qquad (1)$$

where a, b, c, and d express a combination of *rotations*, *scalings*, and *shears* while the values e and f encode the *translations*.

An Iterated Function System is a set of *contractive* transformations $\{M_1, \ldots, M_m\}$ on compact sets that define a set-valued mapping S, such that, for any nonempty compact set E, $S(E) = \bigcup_{i=1}^{m} M_i(E)$. S must be contractive in the Haussdorf metric. The property of IFSs that is more relevant to this work is that for each IFS defined on some $E \subseteq \mathbf{R}^n$ there *always exists* a *unique* nonempty compact set $F \subseteq E$ such that $F = \bigcup_{i=1}^{m} M_i(F)$ (*Fixed Point Theorem* [19, Chap. 9]). Here, F is the *invariant set* of the IFS.

Another important result, which is at the basis of our work, is a consequence of the well-known Collage Theorem [8]: given any compact subset E of $\mathbf{R}^n$, it is always possible to find an IFS whose invariant set approximates E as closely as desired. As a special case, arbitrarily complex images (like those of many objects in nature such as trees or clouds [38]) can be represented and/or reproduced by means of a set of transformations that cover the initial image with many contracted copies of itself. The problem of finding the IFS that produces a picture is still open and is known as the Inverse Problem of IFS. Despite the difficulty of finding it, following from the Collage Theorem, each image can be considered as the fixed point of an IFS $= \{M_1, \ldots, M_m\}$. Since this is uniquely defined by the parameters of its transformations, we can consider the set as a representation of the image. The number of parameters used to represent a bilevel image will be $n = m \times 6$, where m is the number of the transformations of the IFS and 6 is the number of parameters of each transformation (see Eq. (1)).

One further characteristic of IFSs that is fundamental to this work is that they are *robust*, which means that small changes in the transformations produce small changes in the invariant sets. Informally speaking, the shape of an invariant set changes in a continuous way when we modify the parameter values in a continuous way. Then, it is reasonable to suppose that the IFS-based encodings obtained for representing similar images will be similar, although one cannot guarantee the continuity of the parameter values; i.e., it is unlikely to find a single model for each class of objects. Starting from this assumption we claim that the IFSs that encode the images of a set of objects of a same class are "more similar" than those of a set of objects belonging to different classes, in the sense that they allow a set of models for each class to be found, and, thus, such encodings can be used as a set of *descriptive features* in image classification tasks. As a consequence, if the IFS-based encoding of a particular instance is very distant from the prototypical cases, we will suppose that it cannot be considered statistically relevant. From now onward, we will call the IFS-based features *fractal features*. This paper reports the results that we obtained in our attempt to prove our claim with the help of Machine Learning techniques. The use of these techniques was made possible by the characteristics of robustness and stableness of the invariant sets together with the conciseness of representation typical of IFS-based image encodings. The number of features required is independent from the width and height of the image [25, 11].

In contrast to our direct approach, the classical approach to feature extraction consists of making the pixel matrix undergo a sequence of successive processing steps that are aimed at representing the information contained in it by means of more and more abstract descriptors (see [34] for a survey). Skipping the lowest levels of processing [45, 33, 42], image analysis can be done by using *line features, shape descriptors, critical points, geometrical models,* and/or *intensity functions*. Line features describe lines, curves, polygons, and circles and are used to produce a geometrical sketch of the image, although no label has yet been assigned to its components. Critical points are higher-level features which allow the location of the geometrical components of a given image. Shape descriptors are application-dependent features, such as the shape area, the length of its contour, or the moments of the shape. Moments are particularly interesting because they can be combined into functions that are *invariant* w.r.t. image scaling and rotation, but they cannot be used to represent shapes characterized by concavities.

A different approach consists of using *geometrical models*, in which numerical vectors represent relative positions of characteristic parts of the shape. Unfortunately, the construction of models requires a lot of a priori knowledge about the objects in the image, and a matching procedure that can overcome problems like distance and rotation is yet to be found. Recently, *photometric representations*, which are based on the brightness of the pixels, have also been proposed [30, 37].

A method that is *an alternative* to all those mentioned above, and that can reasonably be considered a low-level processing technique, is the one that was used in the StatLog project for representing the digits [27], which consists of lowering the resolution of the image. The obvious disadvantages of this technique are that the number of resulting features depends on the size of the original image and that a loss of meaningful details is possible.

3. EXTRACTION OF FRACTAL FEATURES

In our first attempt at extracting fractal features we used a well-known fractal compression system, i.e., the *enc* program by Fisher [20]. The motivation was twofold. On one hand, to the best of our knowledge, fractal compression systems were the only systems that allowed image fractal encoding; *enc* was chosen because it is representative of a wide class of such systems. It is widely used in practice and mentioned in the literature. Furthermore, we thought that, by finding image encodings which allow precise image reconstructions the compression systems captured some “essential traits” in the images that could be useful for classification. We wanted to verify whether it was possible to induce a body of classification knowledge from this kind of encoding without developing a new fractal feature extraction method. Soon, however, the need for a different approach emerged. In fact, even though *enc* allowed some interesting results to be achieved, as we will show in Section 4, the overall performance of the learned classifiers was quite poor, and, as we later found out, the same would happen when any other encoder based (like *enc*) on the Collage Theorem for Local IFSs was used [9]. The reason is that these systems, which are widely used for compression, do not show the robustness property that assures that similar IFSs have similar invariant sets (see previous section), a fundamental property when we want to use the obtained encodings for inducing classification knowledge. This is why we later developed a different system, based on Barnsley’s original formulation of the Collage Theorem and not the Collage Theorem for Local IFS, which is a specification of the former theorem developed for obtaining better performance in compression tasks.

3.1. XFF: A System for eXtracting Fractal Features

In this section we present XFF, the system that we developed to extract 2-D image discriminant fractal encodings. The kinds of images we focused on represent isolated objects, i.e., objects that are extracted from scenes, for instance, by means of segmentation [39] (many examples are reported in the figures in Section 4). This choice is common to all those cases when the target is classification, and, at least in this stage of our work, it is a necessary choice although we are aware that this is an open research problem.

The system XFF was applied to black and white bilevel images as well as to gray-level image test-benches. We also assumed that the objects were correctly oriented.

The principle underlying XFF is the same as that underlying fractal compression systems, looking for a set of transformations that encode the given image. The main difference stands in the fact that it is inspired by the Collage Theorem as originally stated by Barnsley and not by the Collage Theorem for Local IFS. In particular, while the compression systems encode a *whole* image by searching for a set of transformations that turn parts of an image into other parts of it, our system focuses *only* on the shape that is represented; i.e., it looks for an IFS that describes the shape discarding the background. As a consequence, the representations we obtain are more concise than those obtained in the case of compression.

Before giving more details, we briefly describe the algorithm. In Section 2 we said that an image is uniquely identified by the parameters of the IFS whose fixed point is the image itself. Our idea is to look for an *approximation,* IFS_{appr}, of such an IFS. Note that since our purpose is classification and not reconstruction the approximation is not required to be very precise. The parameters of the transformations that belong to IFS_{appr} will be used as descriptive features for the processed image.

ALGORITHM 1 (Top Level).

```
  **Load** image.
  **Scale** the loaded image.
  **Build** the proto-transformations.
  **Build** a covering for the initial image.
  **Save** the produced covering.
```

IFS_{appr} is obtained by covering the shape at hand with a number of contractions of it. First, a set of transformations that are parametric w.r.t. e and f (see Eq. (1)) is built. We call them *proto-transformations*. The proto-transformations are obtained either by rotating a scaled copy of the original image or by flipping a previously built proto-transformation around the X or around the Y axes. The scaling factor, the number of proto-transformations, and the angles that define the rotations are decided by the user. The same scaling factor is used to reduce both the width and the height of the image.

Once the proto-transformations are ready, the encoding phase can start.

ALGORITHM 2 (Cover Image (*width, height*)).

```
1. begin
2.   $\mathrm{IFS}_{appr} := \emptyset$;
3.   $currBest := null$;
4.   while the stop criterion is not verified do
5.     for $x \in [1, width]$ and $y \in [1, height]$ do
6.           instantiate each proto-transformation by setting parameter $e$ to $x$ and
                 parameter $f$ to $y$, thus obtaining a set of candidate transformations;
7.           evaluate each candidate transformation and select the best one;
8.         if it scores better than the current best candidate transformation
```

```
 9.             then $currBest$ := selected transformation;
10.             end if
11.              increment $x$ and $y$ of a predefined step;
12.        end for
13.    $\text{IFS}_{appr} := \text{IFS}_{appr} \cup \{currBest\}$;
14.   end while
15.  end.
```

Here *width* and *height* are the width and the height of the original image.

In words, at each iteration a set of candidate transformations is produced on a regular grid of points, then a scoring criterion is applied to each candidate transformation, the best one is selected and added to IFS_{appr}, and the pixels that are covered by the selected transformation are marked. Note that IFS_{appr} can contain different instantiations of the same proto-transformation.

The *stop criterion* is, alternatively, a test on the number of iterations (go on until N transformations have been selected) or a test on the amount of foreground that has been covered (go on until M% of the foreground pixels have been covered).

Finally, the *scoring procedure* works as follows. Since any gray-level image can be represented by a *matrix*, whose elements are the values of the pixels, we decided to use a *weighted mean square distance*. The scoring is done by comparing two matrices, matrix *currTransf*, corresponding to the transformation at issue, and a part of the matrix *image*, corresponding to the original image. The portion of image considered has the same size as *currTransf*, and its top-left vertex is defined by the parameters e_{curr} and f_{curr} of *currTransf*. Calling $w1$ the width of such a matrix and $h1$ its height, the scoring is the sum, pixel by pixel, of $s_{ij} = ((image[i'][j'] - currTransf[i][j])^2 + 1) * punish[i][j]$, where $i \in [1, w1]$, $j \in [1, h1]$, $i' = e_{curr} + i$, and $j' = f_{curr} + j$. The "+1" in the formula had to be added for a practical reason, that is, to be able to take into account the information given by $punish[i][j]$ in the case of a perfect covering, i.e., when $\forall i, j(image[i'][j'] = currTransf[i][j])$. Here, *punish* is a matrix of weights, of the same size as *image*, that is used to spread the transformations all over the shape, by forcing the encoding procedure to prefer pixels that do not belong to the background of the original image and, as a second requirement, that have not been covered yet. Since IFS_{appr} is built by means of an iterative process, the values contained in *punish* change over time to take into account the increasing part of image that is being covered. The s_{ij} are computed only for those pixels which belong to the *foreground* of the transformation.

The scoring procedure is necessary to allow the comparison and the selection of candidate transformations. When two candidates are to be compared, first their scores are computed and then the following criterion is applied: the less the $\sum_{ij} s_{ij}$ the better the evaluation. Then, the transformations which minimize their difference with a part of the original image that is left to cover are preferred.

The outcome of XFF is an IFS $\{M_1, \ldots, M_m\}$. Each transformation M_k is coded by three numbers, an identifier of the proto-transformation it derives from (this is possible because the number of proto-transformations is finite, see next section) and parameters e and f. Thus, the number of features required to encode an image is $m \times 3$.

3.2. *Constraining the Search*

A problem faced by all the algorithms that try to obtain an IFS from a picture is the great size of the search space. Even though many heuristics have been proposed ([31], for instance, discusses in detail the issue of reducing the search space, by defining a set of constraints on the parameters of the transformations, proving their effectiveness, and giving a method for implementing them), no *efficient* solution has been developed yet. This problem is particularly relevant in a learning and classification task, where many images are to be encoded before learning takes place. We decided to tackle it by building IFS$_{appr}$ one transformation at a time, through an iterative process, and by reducing the search space for every transformation. This is done by applying the same scaling factor (which is fixed, decided a priori, and is the same for all the proto-transformations) to both image sides and allowing only a fixed number of alternative rotations. In the experiments on the trees only rotations that were multiple of $\pi/2$ were allowed while in the experiments on the digits only rotations belonging to set $\{0, \pi/4, \pi/2, 3\pi/4\}$ were used. This choice is along the line of what is done, for instance, by *enc* [20], which allows only eight proto-transformations: the identity, $\pi/2$, π, and $3\pi/2$ anticlockwise rotations, and the reflection in the two diagonals and in the two axes.

Our method is rougher than others (such as the one proposed in [31]), but it is quick and simple, and sufficient for our purposes, even though we are aware that in order to obtain better approximations (and higher recognition rates) it will be necessary to improve it. The constraints we apply have, however, the advantage of allowing the reduction of the number of coefficients necessary to represent a transformation. In fact, since only a limited number of scalings and rotations, say q, is allowed, it is possible to map each proto-transformation into a *single* integer. A numerical identifier will, then, be sufficient to encode the proto-transformations. Consequently, XFF encodes each image of the test-beds that we used in the experiments in a time on the order of 1 min. Efficiency of encoding is important when dealing with Machine Learning applications, where usually many hundreds, or even thousands, of instances are to be processed.

Another topic to discuss is the stop criterion. At present, knowing when to stop applying proto-transformations is an open problem because every image has an infinite number of IFS approximations. Imposing a fixed number of transformations is a very simple criterion: its only disadvantage is that it forces the use of a heuristic to find out the "best" number of transformations for a given application. In our experience, a few attempts were enough but, in general, a more "intelligent" criterion could be necessary.

The complexity of the algorithm is polynomial in the number of pixels; see [3].

3.3. *Postprocessing*

Before learning took place, the transformations in each IFS were sorted according to a *polar ordering* w.r.t. the point of coordinates $(0.5{*}width, 0)$. Let us explain the need for this preprocessing step by means of an example. Consider two hand-written "4's," the first with a thick *vertical* line and the second with a thick *oblique* line. Since the encoding algorithm selects first those transformations that cover *more pixels*, it is very likely that the first transformation generated to encode the first 4 covers the vertical line, whereas that created to cover the second 4 covers the oblique line. The transformations contained in the two IFSs will be analogous, the main difference being their position in the generation sequence. An ideal *learning system* should be able to compare sets of transformations; that is, it should be able to handle *first-order* representations, in which sets of objects are learned without taking into account the order of the objects. Unfortunately, most *numerical* learning systems (neural networks) only work at a *propositional* level. The ordering preprocessing step was introduced to fill this gap. Other simple orderings were also tried, but they were less effective.

4. EXPERIMENTS

This section describes the two test-beds on which we worked, the trees and the hand-written digits, and reports the results that we achieved in various experiments, each followed by a brief comment. For both test-beds the experiments have been organized as follows. First, we extracted the IFS-based features by hand-coding the examples (i.e., we played the part of the fractal encoder, covering each instance with a set of self-affine copies) with the help of a graphical program, *Fractal Vision* [32]. The aim of these preliminary experiments was to verify whether fractal features are able to capture discriminant information. Due to the excellent results obtained, we then tried to automatize the encoding process. As mentioned, the first attempt was to use *enc* [20], a fractal compression system. This algorithm exploits the Collage Theorem for Local IFSs [9]: instead of covering the whole image with contracted affine copies, it covers the image with contracted affine copies of *parts* of it. The *enc* system imposes some constraints in order to reduce the search, allowing only the eight transformations seen in Section 3.2. Each transformation is specified by five numbers: t between 0 and 7, which codes the selected geometrical transformation; e and f, for the translation; and two real numbers, s and o, indicating the luminance transformation. Afterward, we developed XFF.

The results reported in the next sections were mostly obtained using, as a learner, feed-forward neural networks[1] trained by means of the *Scaled-Conjugate Gradient* (SCG) rule [28]. This is basically a variant of gradient descent that exploits second-order derivatives for reducing the number of learning epochs. This model is interesting also because its convergence is proven to be independent of the values of the learning parameters (such as the learning factor), which only affect the speed of convergence. Training always took place in a few minutes, the number of epochs equaling at most 1000. For each test-bed, over 400 experiments were carried out, partly to test different kinds of encodings and partly to cross-validate the results. When not stated otherwise, six-fold cross-validation was applied; i.e., the results reported are the averages of those obtained on six different partitions of the given examples into learning/test set pairs. The experiments on each learning/test set pair were repeated from three to five times.

4.1. Digits

This test-bed is a collection of hand-written instances of the ten digits, acquired with a resolution of 16×16 pixels and 256 gray levels, gathered by the German Federal Post. (See Figs. 1–3.)

Our first goal was to verify whether fractal features are able to capture discriminant information about linear images. This goal was achieved by hand-coding the examples and working only on three digits, chosen for their similarity in terms of handwriting: 1, 4, and 7 (see Fig. 2). Each digit was covered with four self-affine copies (24 features). The network had 80 hidden neurons. The learning set contained 30 instances (10 per class) while the test

[1] All the neural networks were implemented by means of the Stuttgart Neural Network Simulator, developed at and made available by the University of Stuttgart, Germany.

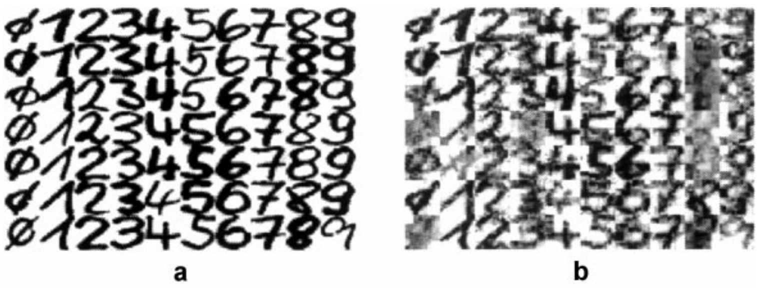

FIG. 1. (a) Some images used in our experiments; (b) Their reconstruction with 16 transformations.

set contained 40 instances per class. A 100% recognition rate was achieved. These results show that if a proper method for placing the transformations is found, fractal features can be used to define shapes through class learning. Thereafter we tried to automatize the process by coding the examples by means of *enc* and by means of XFF.

When we used *enc*, each digit was approximated by 16 transformations[2] (80 features); Fig. 1 reports some examples. Three neural networks had to be used: the first (250 hidden neurons) was trained to recognize digits from 0 through 3 using as counterexamples also all digits from 4 through 9. The second (200 hidden neurons) was trained to classify digits 4, 5, and 6. The third (250 hidden neurons) learned to classify the remaining digits. The learning set was made of 500 digits (50 per class) while the test set was made of 2500 instances (250 per class). Results are reported in Table 1. We can observe that for 5 classes out of 10, a recognition rate higher than 70% was achieved, but the average low performance suggests that better ways for extracting the IFS should be found. On the other hand, this result should be expected since, as already mentioned, the fractal encoding by means of Local IFS is not robust. Furthermore, the learning systems showed no generalization capability when the training sets contained less than 50 instances per class.

Afterward, we applied XFF. One thousand examples (100 per digit) were used. The learning set was made of 300 (30 per class) examples while the test set was made of 700 examples. Threefold cross-validation was applied. In some experiments the digits were binarized. Since each digit is made of a thin line, we have used a set of contractions of the *square-box* containing the digit. This can be done because the structure of an object is information that is intrinsic in the object itself and, as long as we are *not* interested in fine reconstruction, it *does not* depend on the shapes used for covering. An example is reported in Fig. 3. The rotations used belong to the set $\{0, \pi/4, \pi/2, 3\pi/4\}$; the scaling is fixed and depends on the experiment.

The results of this series of experiments are reported in Table 2. Rows (a), (b), and (c) contain the average recognition rates obtained on the test sets digit per digit. All the NNs used had 500 hidden neurons. The only difference between these experiments consists in the values of the parameters of XFF: the results of row (a) were obtained on digits covered by means of five transformations, where the 16×16 pixel frame square box is scaled to a 7×3 pixel rectangle (15 inputs, i.e., five transformations times three parameters each); those of row (b) were covered by means of six transformations with a scaling to a 5×3 pixel

[2] Fewer transformations were not sufficient while more transformations produced too many input features. Remember that since a quad-tree partitioning is used, increasing or decreasing the partition levels number *drastically* modifies the number of transformations applied.

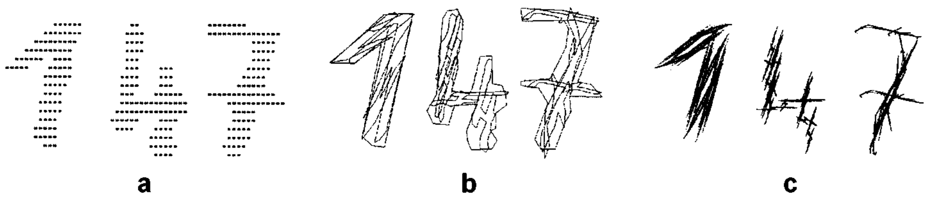

FIG. 2. (a) Binarized instances of the digits 1, 4, and 7. (b) Digit covering with four self-affine transformations. (c) Digit fractal reconstruction computed by Fractal Vision.

rectangle (18 inputs); those of row (c) by six transformations with a scaling to a 5 × 3 pixel rectangle (18 inputs), the difference with the previous case being that the data were not binarized any more; i.e., the 256 grey levels were maintained, and the search for the best matching transformations took into account also the intensity level of the pixels.

In our opinion, the most interesting result of these experiments is that a single SCG neural network achieved an average recognition rate that is higher than 84% for 7 classes out of 10. This result accounts for the fact that fractal features can be produced so as to capture discriminant information.

Another reason for working on digits is that they are well known in the Machine Learning literature. In particular, they were used in the *StatLog* project [27], in which the performance of many learning systems was measured. The comparison we will do here is aimed at having an idea of the general goodness of our results and is limited to the recognition rates reported as a result of StatLog and our project. We do not mean to use these results to state that XFF is better than other character feature extraction methods. This kind of conclusion would require a much deeper investigation that is out of the scope of the present research.

The feature extraction technique used in StatLog consisted of reducing each 16 × 16 matrix to a 4 × 4 matrix in the way described in Section 2. Eighteen learning systems were trained and tested on these data. Their performance varied between 78.0 and 98.4% on the learning set (against the 100% recognition rate obtained using XFF) and between 71.5 and 95.5% on the test set; in the latter case only 8 systems out of 18 obtained an average recognition rate higher than 88%. Using six transformations, we achieved an average 85% on the test sets, a result that is very close to those obtained by AC2 (84.5%), INDCART [13] (84.6%), C4.5 [40, 41] (85.1%), NewID (85.5%), and CN2 [16] (86.6%), although it is still far from the 95.5% of k-N-N (the best-performing system on this test-bed). The use of

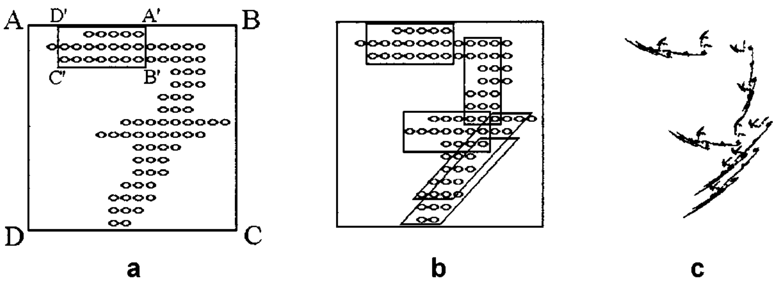

FIG. 3. (a) Sample digit inside its square box *ABCD* (circles represent black pixels): application of the first affine contraction resulting in $A'B'C'D'$. (b) Final coverage using five contractions. (c) Fractal reconstruction: note how the digit structure is captured.

TABLE 1
Recognition Rates Obtained Encoding the Digits with *enc*

	0	1	2	3	4	5	6	7	8	9	Avg.
%	79.6	78.0	71.6	60.8	65.6	81.6	72.4	69.2	26.0	47.2	65.2

fractal features has, however, one considerable advantage w.r.t. the other kinds of encoding, such as the one used in StatLog: the number of features necessary to represent an image, besides being small, does not depend on the image size.

Along the lines of what done in StatLog, we did also some experiments in order to check the dependence of the recognition rates on the particular learning system used. The results of this work can be found in [4].

4.2. Trees

The other considered task is the recognition of *synthetic* images of three different kinds of trees (see Fig. 4). The three classes were called *lime-tree*, *oak*, and *elm*. The difficulty lies partly in the similarity between the three classes, partly in their complexity, and partly in their size. No specific shape descriptor (such as the inclination of the branches) was supplied to the automatic learner.

One hundred and fifty instances were available, 50 per class. All images were 320×240 pixels. We generated each tree by using a recursive procedure which, starting from the trunk and applying scalings and rotations of it, produces the branches. A wide variety of trees can be generated from a few prototypes for each class by applying random perturbations to the parameters (scalings and rotations). Observe that different random perturbations were applied at every branch, so trees were not generated by the application of IFSs.

Also in this case the first experiments were aimed at verifying the possibility of capturing discriminant information by means of fractal features. Each tree was encoded by 42 features, produced by hand with the help of Fractal Vision [35]. A neural network with 150 hidden neurons was used; the learning set contained 75 instances while the test set contained 105. A 100% recognition rate was obtained.

Afterwards we applied *enc*. Each instance was approximated by 16 transformations (80 features). In the best experiment an average 68% recognition rate was achieved; cross-validation resulted in an average 64% recognition rate.

At last we applied XFF. Depending on the experiment, each tree was encoded using 3, 4, or 5 transformations (i.e., respectively, 9, 12, or 15 numbers). These encodings are very compact w.r.t. those obtained by means of other common techniques. For instance, averaging squares of 16 pixels, we would obtain a matrix made of 80×60 pixels (4,800

TABLE 2
Average Recognition Rates Achieved Working on the Digits Encoded by XFF

	0	1	2	3	4	5	6	7	8	9	Avg.
(a)	89.05	99.05	86.67	86.67	84.95	78.57	80.00	96.67	47.62	79.52	82.86
(b)	84.95	96.67	91.90	84.19	81.43	86.67	90.19	92.38	62.38	79.00	85.00
(c)	79.52	97.14	90.48	82.38	83.33	86.66	83.33	96.19	62.86	80.00	84.19

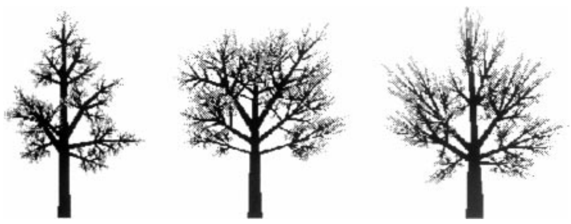

FIG. 4. Instances of elm, oak, and lime-tree.

features). We could have obtained smaller encodings; nevertheless, each square would have summarized too big of an area to be meaningful.

The best results were obtained using four transformations with a scaling factor equal to 0.35. The number of hidden neurons varied depending on the experiment; the best results obtained using four transformations were achieved using 100 hidden neurons. The networks were trained in a number of epochs in the range [500, 1000].

Some outcomes of these experiments are reported in Table 3. They were obtained using disjunct learning and test sets made of 75 instances each. Columns (a) and (b) report the results obtained using the best setup described above. The results of column (a) were obtained on some specific runs and were reported to show that in some cases a perfect recognition was obtained. Column (b) reports the average results obtained by cross-validation. All these results were obtained using 100 hidden neurons; different numbers of neurons were also tried, always obtaining average recognition rates higher than 80%. Columns (c) and (d) report the average results obtained when the trees were encoded, respectively, by means of three and five transformations (9 and 15 input units, respectively). Different numbers of hidden neurons were tried also in this case; in particular, when five transformations were applied, the best results were obtained using 200 hidden neurons. A 100% recognition rate was always achieved on the learning set (column (e)), independently from the number of instances contained in it and from the setup. Recognition rates between 75 and 80% were achieved using smaller learning sets.

A first observation is that the above-mentioned results look very good, especially if we consider the number of features used and also the fact that the most discriminant information that allows trees of different species to be distinguished (the inclination of the branches) was not used due to the limitations posed on the rotations. Twelve numbers were a sufficient encoding for a tree image of size 320×240, allowing an average 85.12% recognition rate to be achieved. We think that this result is sufficient proof of the power of discrimination that can be summarized by fractal features. The optimal performance obtained in some cases, then, strengthens this result.

TABLE 3
Recognition Rates Obtained Encoding the Trees with XFF

	(a)	(b)	(c)	(d)	(e)
Elm	100%	88.61%	86.00%	90.18%	100%
Oak	100%	91.38%	95.34%	83.64%	100%
Lime-tree	100%	75.38%	67.34%	61.45%	100%
Average	100%	85.12%	82.89%	78.42%	100%

A second appealing characteristic is that the fractal features were extracted in an automatic way by a program. We believe that higher average recognition rates could be achieved by widening the range of possible transformations. Nevertheless, relaxing the constraints that were imposed would also have the effect of increasing the computational complexity of the search.

5. CONCLUSIONS

Since Mandelbrot defined them, fractals have been widely addressed in the literature, and, in strict connection with them, IFSs have been developed and used in various domains. However, the only works that have used fractals in classification tasks, that we could find in the literature, are [21], where a neural network is trained to distinguish fractal images from nonfractal images, [18], where fractals are used to build the belief functions of a Bayesian classifier, and [23], where a metric for planar self-similar forms is defined. In this last work, which is probably the closest to the topics we explored, a metric is defined and used for measuring the distance between IFSs: the smaller the distance the more similar the represented shapes. Nevertheless, the authors do not describe any method for using this distance measure in classification tasks unless, as implicitly stated, some models are given a priori for comparison. Even if such models (which are, anyway, to be built or learned somehow) were available, in order to compare them to an image it would still be necessary to find an IFS encoding for the image itself. Only then would it be possible to apply this metric (or a similar one) to compute the similarity between different instances and/or between some instance and the prototypes.

The main difference between the works of the kind described above and the method that we propose is that we focus on finding (in an automatic way) the IFS representations of a set of given images, and, then, we use these representations to learn (again, in an automatic way) classification knowledge. Thus, by attacking the problems of image feature extraction and of the acquisition of classification knowledge, our method is to some extent complementary to [23]. This is also the greatest novelty of our proposal. Along the same lines of the research presented here, a work by Schouten *et al.* [44] was recently published which tackles the problem of extracting information about the structural content of an image from the coefficients that describe the fractal code of it. Schouten *et al.* do this by applying simple statistics to the fractal codes.

With our work we have proved that some kinds of IFS-based encodings capture discriminant information of images of different objects and can, then, be used also in classification tasks. However, since many of the current fractal encoders—that were developed for image compression—are based on quad-tree (or nona-tree, where the image is divided in nine parts instead of four) partitioning, they cannot grant that similar objects will have similar encodings. For this reason it is not possible to achieve significative performance by working on the codings that they produce. The actual problem is, then, to develop algorithms which show this property and, then, allow fractal discriminant features to be extracted in an automatic way.

In our research, we have implemented and used a very simple fractal feature extraction system, XFF, that can be applied to any isolated shape. Experiments immediately showed a remarkable increase of the recognition rates, achieving an average 85% on both test-beds. We believe that the results we have obtained are a sufficient motivation for a deeper investigation of this kind of feature extraction method, with particular attention to the constraints that decide which transformations are to be taken into account. In fact, as we have shown, when no restriction was imposed on the transformations used (i.e., when instances were hand-coded), a neural network could perfectly distinguish instances of different classes of objects. The open problem is, then, to find a *better criterion* for guiding transformation selection.

One of the characteristics of IFSs is that their use reduces both the number of input space dimensions and the set of possible values for each input feature. So, for instance, in the experiments which gave the best results each tree image (sized 320×240) was represented by just 12 numbers. Moreover, the number of fractal features used to represent a shape is *independent* from the image width and height, being related to the shape intrinsic complexity only. These two characteristics make IFS representations extremely interesting in the case of image classification. In fact, according to the results of StatLog [27], many learning algorithms cannot deal with applications having hundreds of dimensions while those that can take too much time before converging. The use of IFSs, then, widens the range of learning systems that can be applied to image classification tasks, decreasing the time required by the training phase.

As explained in the introduction, the possible applications are many and range from graphical interfaces for design systems to the recognition and retrieval of images in large repositories (e.g., in Web browsing applications) or in virtual reality systems. In particular, it would be interesting to study the performance of IFS-based encoding for classification in the case of binary trademark and logo interpretation.

ACKNOWLEDGMENTS

We thank Professor Lorenza Saitta, Dr. Giuseppe Lo Bello, and Dr. Lavinia Egidi for the helpful discussions and Professor Charles Taylor for all the information he gave us about StatLog. We also thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

1. R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, Analyzing images containing multiple sparse patterns with neural networks, in *Proceedings of IJCAI-91*, Sidney, Australia, 1991.
2. M. Baldoni, C. Baroglio, and D. Cavagnino, XFF: A simple method to eXtract fractal features for 2D object recognition, in *Advances in Pattern Recognition, Joint IAPR International Worshops SSPR'98 and SPR'98* (A. Amin, D. Dori, P. Pudil, and H. Freeman, Eds.), pp. 382–389, Springer-Verlag, Berlin/New York, 1998.
3. M. Baldoni, C. Baroglio, D. Cavagnino, and L. Egidi. Learning to classify images by means of iterated function systems. in *Fractals and Beyond, Complexities in the Sciences, Proceedings of Fractal'98* (M. M. Novak, Ed.), pp. 173–182, World Scientific, Singapore, 1998.
4. M. Baldoni, C. Baroglio, D. Cavagnino, and G. Lo Bello. Extraction of discriminant features from image fractal encoding, in *Proceedings of AI*IA 97: Advances in Artificial Intelligence* (M. Lenzerini, Ed.), Vol. 1321 of *LNAI*, pp. 127–138. Springer-Verlag, Berlin/New York, 1997.
5. M. Baldoni, C. Baroglio, D. Cavagnino, and L. Saitta, Towards automatic fractal feature extraction for image recognition, in *Feature Extraction, Construction and Selection—A Data Mining Perspective* (H. Liu and H. Motoda, Eds.), Chap. 22, pp. 357–373. Kluwer Academic, Dordrecht/Norwell, MA, 1998.
6. N. Balossino, D. Cavagnino, G. Magliacani, and M. T. Reineri, Development of eidethical data base for hospital surgical unit by using fractal compression, in *Proceedings of the Health Telematics '95* (M. Bracale and F. Denoth, Eds.), Naples, Italy, 1995.
7. M. Barnsley, *Fractals Everywhere*, Academic Press, San Diego, 1988.
8. M. Barnsley and S. Demko, Iterated function systems and the global construction of fractals, in *The Proceedings of the Royal Society of London*, Vol. A399, pp. 243–275, 1985.

9. M. Barnsley and L. P. Hurd, *Fractal Image Compression*, AK Peters, Wellesley, MA, 1993.
10. M. Barnsley and A. Sloan, A better way to compress images, *BYTE Mag.*, **13**(1), 1988, 215–223.
11. T. Bedford, F. M. Dekking, M. Breeuwer, M. S. Keane, and D. van Schooneveld, Fractal coding of monochrome images, *Signal Process. Image Comm.* (6), 1994, 405–419.
12. P. Besl and R. Jain, Three dimensional object recognition, *ACM Comput. Surveys* (17), 1985, 75–154.
13. L. Breiman, J. Friedman, J. Ohlsen, and C. Stone, *Classification and Regression Trees*, Wadsworth & Brooks, Pacific Grove, CA, 1984.
14. B. Cheng, A. Zhang, R. Acharya, and C. Sibata, Using fractal coding to index image content for a digital library, Technical Report 95-05, SUNY, Buffalo, NY, 1995.
15. R. Chin and C. Dyer, Model-based recognition in robot vision, *ACM Comput. Surveys* (18), 1986, 67–108.
16. P. Clark and T. Niblett, The CN2 induction algorithm, *Machine Learning* **3**(4), 1989, 261–283.
17. H. A. Cohen, Retrieval and browsing of images using image thumbnails, *Journal of Visual Communication and Image Representation* **8**(2), 1997, 226–234.
18. A. M. Erkmen and H. E. Stephanou, Information fractals for evidential pattern classification, *IEEE Trans. Systems Man Cybernet.* **20**(5), 1990, 1103–1114.
19. K. Falconer, *Fractal Geometry, Mathematical Foundations and Applications*, Wiley, Chichester, UK, 1990.
20. Y. Fisher, *Fractal Compression: Theory and Application to Digital Images*, Springer Verlag, Berlin/New York, 1994.
21. B. Freisleben, J. H. Greve, and J. Löber, Recognition of fractal images using a neural network, in *New Trends in Neural Computation, IWANN '93* (J. Mira, J. Cabestany, and A. Prieto, Eds.), Vol. 686 of *LNCS*, pp. 632–637, Springer-Verlag, Berlin/New York, 1993.
22. W. Grimson, T. Lozano-Pérez, and D. Huttenlocher, *Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge, MA, 1990.
23. A. Imiya, Y. Fujiwar, and T. Kawashima, A metric of planar self-similar forms, in *Advances in Structural and Syntactical Pattern Recognition, SSPR'96* (P. Perner, P. Wang, and A. Rosenfeld, Eds.), Vol. 1121 of *LNCS*, pp. 100–109, Springer-Verlag, Berlin/New York, 1996.
24. A. Jacquin, Image coding based on fractal theory of iterated contractive image transforms, in *Proceedings of SPIE, Visual Communications and and Image Processing '90*, Vol. 1360, 1990.
25. G. Lu, Fractal image compression. *Signal Process. Image Comm.* (5), 1993, 327–343.
26. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, San Francisco, CA, 1982.
27. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Series in Artificial Intelligence, Prentice Hall, New York, 1994.
28. M. F. Moller, A scaled conjungate gradient algorithm for fast supervised learning, *Neural Networks* **6**, 1993, 525–533.
29. D. M. Monro and F. Dudbridge, Fractal block coding of images, *Electron. Lett.* **63**(11), 1992, 1053–1055.
30. O. Nakamura, S. Mathur, and T. Minami, Identification of human faces based on isodensity maps, *Pattern Recognition* (24), 1991, 263–272.
31. D. J. Nettleton and R. Garigliano, Reductions in the search space for deriving a fractal set of an arbitrary shape, *J. Math. Imaging Vision* **6**, 1996, 379–392.
32. H. Nishida, Structural shape indexing with feature generation models, *J. Comput. Vision Image Understand.* **73**(1), 1999, 121–136.
33. L. O'Gorman, Binarization and multi-thresholding of document images using connectivity, *CVGIP Graphical Models Image Process.* **56**(6), 1994, 494–506.
34. L. O'Gorman, Basic techniques and symbol-level recognition—An overview, in *Graphics Recognition, Methods and Applications* (K. Tombre and R. Kasturi, Eds.), Vol. 1072 of *LNCS*, pp. 1–12. Springer-Verlag, Berlin/New York, 1995.
35. D. Oliver, *Fractal Vision, Put Fractals to Work for You*. Sams Publishing, Indiana, 1992.
36. A. R. Pearce and T. Caelli, Interactively matching hand-drawings using induction, *J. Comput. Vision Image Understand.* **73**(3), 1999, 391–403.

37. A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition, in *Proceedings of Conference on Computer Vision and Pattern Recognition*, 1994.

38. A. P. Pentland, Fractal-based description of natural scenes, *IEEE Trans. Pattern Anal. Machine Intelligence* (PAMI-6), 1984, 661–674.

39. W. K. Pratt, *Digital Image Processing*, Wiley, New York, 1978.

40. J. R. Quinlan, Induction of decision trees, *Machine Learning* (1), 1986, 81–106.

41. J. R. Quinlan, *C4.5 Programs for Machine Learning*, Kaufmann, Los Altos CA, 1993.

42. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Vols. 1 and 2, 2nd ed., Academic Press, San Diego, 1982.

43. D. Saupe and R. Hamzaoui, A review of the fractal image compression literature, *Comput. Graphics* **28**(4), 1994, 268–273.

44. B. A. M. Schouten and P. M. de Zeeuw, Feature extraction using fractal codes, in *Proceedings of Visual-99, Amsterdam, The Netherlands, June 1999*, Kluwer Academic, Dordrecht/Norwell, MA, 1999.

45. W. H. Tsai, Moment-preserving thresholding: A new approach, *Comput. Vision Graphics Image Process.* **29**, 1985, 377–393.

46. N. Wadströmer, A solution to the inverse problem of iterated function systems, in *Proceedings of the Picture Coding Symposium*, Melbourne, 1996.

47. I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand–Reinhold, New York, 1994.