XFF: A Simple Method to eXtract Fractal Features for 2D Object Recognition

Matteo Baldoni, Cristina Baroglio and Davide Cavagnino

Dipartimento di Informatica — Università degli Studi di Torino
Corso Svizzera, 185 — I-10149 Torino (Italy)
E-mail: {baldoni,baroglio,davide}@di.unito.it
URL: http://www.di.unito.it/~baldoni/fractals

Abstract. Automatic *recognition* of objects from their visual representation is a hard and computationally expensive task, mainly because it is difficult to extract the necessary discriminant information from the raw data. The current approaches to image classification commonly exploit some geometrical models of the objects of interest. The classification process is based on the comparison between the image at hand and the models for the classes: the object's class will be the one of the closest model. In this paper we present *XFF*, a new method for representing 2-D images, based on the extraction of a set of *fractal features* which exploits the approximation of an image with an *Iterated Function System*, a technique that is already at the basis of many successful image compression tools. One of the advantages of these features is that they can be used directly to train an adaptive classifier, without the need of any a priori knowledge.

1 Introduction

Automatic recognition of images from their visual representation is a problem of great importance due to the many potential applications of automatic recognition systems (see [17] for a survey). The main problem with image classification is its high computational complexity, due to the large amount of raw data to be handled. Hence the need for applying feature extraction techniques, which allow to work at a *more abstract* level of detail [11]. The ideal features are *easy* to extract, *robust*, *invariant* w.r.t. the image size, and must have a *high discriminatory power*, i.e., be different for different classes of objects. Well-known approaches to image classification suggest to use model-based methods [5, 6]; in our work we have, instead, investigated an approach that does not need any a priori knowledge, being based on the extraction of a set of characterizing features, based on *Iterated Function Systems* (IFSs) [4], that we call *fractal features*. Image encoding by means of IFSs has been widely investigated to develop image compression systems, however, to our knowledge, IFSs were never proposed before for extracting features to be used in characterization and/or classification tasks.

The purpose of this paper is to show that, given an isolated image classification problem, the fractal encodings of the images of objects that belong to the same class are *more similar* than those of objects of different kinds. This will be done both reporting the theoretical backgrounds of the fractal features (Section 2) and describing a system, XFF (*eXtraction of Fractal Features*), that extracts 2-D *isolated image* fractal encodings to be used for training adaptive classifiers (Section 3). XFF derives from the experience of [1], where a prototype system was developed for the specific application of hand-written digit classification. Differently than the prototype, XFF can deal with *any* kind of shape, being particularly suitable to handle images with an intrinsic complex structure. For showing an example, we report the results obtained on a complex and irregular image classification task, that consists in recognizing different kinds of trees (Section 4).

2 IFS-based Fractal Features

The basic idea underlying *fractals* is that, at any scale, they are *self-similar*. In the plane, fractal images can be generated simply by iterating a geometrical transformation that maps the given image in smaller copies of itself. The fractal is the limit obtained when the number of iterations goes to infinity. Even though, in principle, this mapping can be any transformation, in practice, *affine* transformations, which preserve the object's shape, are preferred. The only necessary condition imposed on the transformations is that they must be *contractive*, i.e. they must move points closer together.

An *affine transformation* in a bidimensional space has the general form:

$$M(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} . \tag{1}$$

Notice that the parameters e and f encode the *translation* component of the transformation.

In general, an *Iterated Function System* (IFS for short) is any collection $\{M_1, \ldots, M_r\}$ of contractions; here we focus on collections of affine contractive transformations. For every IFS in $\mathbf{R}^n$ there exists a set F such that $F = \bigcup_{i=1}^{m} M_i(F)$. F is called *invariant set*; it is *unique* and *non-empty* and can be a *fractal*. Moreover, as a consequence of the *Collage Theorem* [2, 8], given any compact subset E of $\mathbf{R}^n$ and an arbitrary precision of approximation, there always exists an IFS, whose invariant set approximates E as finely as desired, although there is currently no method that allows to find such approximations in a fully automatic way. A particular case is that of images, which can be considered as compact subsets of the plane; the Collage Theorem helps in reconstructing an image. The method that is used consists in reconstructing an image by covering it with a set of contracted affine copies of itself. Other characteristics of the IFS are that it is *robust* and *stable*, i.e., small changes in the transformations produce small changes in its invariant set; hence, varying the coefficients in a continuous way, the shape of the invariant set also changes in a continuous manner. Details can be found in [8, 4].

Following from the Collage Theorem [3], each image can be approximated to any desired precision by computing the invariant set of some IFS. The advantage of these approximations is that they allow images with an intrinsic complex structure to be encoded in a very *concise* way, the size of the representation being *independent* than the image size. In this paper we use the parameters of the transformations M_i as a set of *descriptive features*, called *fractal features.* Since IFSs are robust and stable, it is reasonable to suppose that the IFSs that approximate the images of a set of objects of a same class will be more similar than those approximating a set of objects belonging to different classes. Furthermore, since we tackle classification problems, we show that it is not necessary that the approximations be fine; rough approximations are enough to our purposes.

Then, given an image we look for an IFS whose invariant set approximates it and use the parameters of such an IFS to represent the original image. The obtained encodings are used to feed a *learning* system to produce a classifier (a Neural Network). This approach is quite different than the common approach to feature extraction (see [17, 11]), which consists in making the pixel matrix undergo a sequence of processing steps, aimed at representing the image by means of more and more abstract descriptors. Differently than these methods, the *fractal features* are simple to extract and they allow the use of adaptive classifiers for building the classification knowledge in an automatic way. Moreover, fractal features allow to represent in a very concise way extremely irregular objects, such as landscapes, clouds, trees [18] (see Section 4).

3 XFF: automatic extraction of fractal features

This section presents XFF, the system we developed to extract 2-D image discriminant fractal encodings of isolated objects. XFF is an operationalization of the Collage Theorem as stated in [3]. Our idea is to look for an IFS, whose invariant set *approximates* the image at hand. In the following we call it IFS$_{appr}$. The parameters of the transformations in IFS$_{appr}$ will be used as the descriptive features of the processed image.

```
Top level:
    1. load image of size $width \times height$;
    2. scale the loaded image;
    3. build the proto-transformations;
    4. build a covering for the initial image;
    5. save the produced covering;
```

IFS$_{appr}$ is obtained by covering the image with a set of contractions of it, a process divided in two phases. First, a set of transformations that are parametric w.r.t. e and f is built. We call them *proto-transformations* and they are obtained either by rotating a scaled copy of the original image or by flipping a previously built proto-transformation around one of the axes. The second phase corresponds to step 4 of the *Top level* and is as follows:

```
Cover image:
  while the stop criterion is not verified, do the following:
    1. for $x \in [1, width]$ and $y \in [1, height]$:
```

   - instantiate each proto-transformation by setting parameter e to x and f to y, thus obtaining a set of candidate transformations;
   - evaluate each candidate transformation and select the best one;
   - if it scores better than the current best candidate transformation, memorize it instead of the previously best scoring one: *currBest* := selected transformation;
   - increment x and y of a predefined step;
2. $\text{IFS}_{appr} := \text{IFS}_{appr} \cup \{currBest\}$;

where IFS_{appr} and *currBest* are initially empty. This procedure produces IFS_{appr} by selecting a set of best scoring candidate transformations, each of which is a copy of one of the proto-transformations built at step 3 of the *top level.* The selection is done according to a *scoring procedure*, that exploits a variant of the *Hamming distance.* This procedure compares two matrices: *transform*, the pixel matrix corresponding to the transformation at issue, and a part of the matrix *image*, corresponding to the original image pixel matrix. The portion considered has the same size as *transform* and its top-left vertex is the point defined by the parameters e and f of the transformation at hand. Calling $w1$ the width of such a matrix and $h1$ its height, the scoring is the sum, pixel by pixel, of $s_{ij} = ((image[i'][j'] - transform[i][j])^2 + 1) * punish_{ij}$, where $i \in [1, w1]$, $j \in [1, h1]$, $i' = x + i$ and $j' = y + j$. The "+1" in the formula had to be added to take into account the information given by $punish_{ij}$ in the case of a perfect covering, i.e. when $\forall i, j(image[i'][j'] = transform[i][j])$. $punish_{ij}$ forces the covering operation to prefer pixels that belong to the foreground of the original image and, as a second requirement, that have not been covered yet. The value of $punish_{ij}$ varies pixel by pixel and changes over time to take into account the part of image already covered by the transformations added to IFS_{appr}. s_{ij} is computed only for those pixels which belong to the foreground of the *transformation.* When a transformation is to be selected out of a set, the score of each element is computed. The transformation that is preferred is the one which minimizes the score, i.e. the *difference* with a part of the original image.

The *outcome* of XFF is an IFS $\{M_1, ..., M_r\}$. Each M_i is coded by *three numbers*: one encoding the applied *proto-transformation* and two respectively equal to x and y (the coordinates of the *point* on which the proto-transformation is instantiated, i.e., parameters e and f). The image is, then, encoded by $r \times 3$ numbers. Each proto-transformation can be encoded by just one number because, since the allowed scalings and rotations are given, the possible alternatives are limited in advance. This fact is due to a problem that all systems that encode images by means of IFSs must face: in principle, the search space they explore is infinite as well as the number of possible solutions, see [9]. In order to make the problem tractable, all such systems *constrain the search* of the IFS by narrowing the range of all the coefficients a, b, c, d of 1. XFF applies the following constraints. The same scaling factor (which is fixed, decided a priori and is the same for all the proto-transformations) is used to reduce both image sides. Furthermore, XFF is parametric w.r.t. the angles that define the rotations, i.e. the user specifies which rotations XFF must use.

Another interesting point to discuss is how to decide when to stop applying proto-transformations, i.e. how many transformations are necessary for capturing the structure of an object. This is, actually, an open problem. Generally speaking, even images of different objects belonging to the same class may require different numbers of transformations for getting a sufficiently informative approximation. However, if we are to use automatic learners for extracting the classification knowledge, it is necessary to find a homogeneous representation for all the instances. In fact, when different length encodings are used, a learning system that can deal with first-order logics should be adopted, however, the development of first-order numerical learners is at the beginning and there are just a few examples of such systems. We have recently started a series of experiments of this kind but they are still in a preliminary phase. Our *stop criterion* is very simple and consists in deciding a priori the number of transformations to use.

4 An example application

The test-bed we chose for presenting XFF is a "complex image" classification problem: *tree recognition* (see also [12]). The classification problem consists in learning to distinguish three kinds of trees, namely *lime-tree*, *oak* and *elm* (Figure 1). All images size is 320×240 pixels. 150 instances were available, 50 per

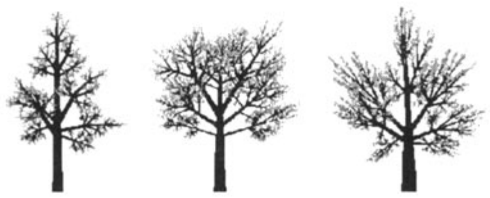

Fig. 1. Instances of Elm, Oak and Lime-tree.

class. Each experiment was repeated from three to five times and 6-fold cross-validation was applied. Depending on the experiment, each instance was encoded simply by using 3, 4 or 5 transformations, i.e. using alternatively 9, 12, or 15 numbers. These encodings are very compact w.r.t. those obtained by means of other common techniques. For example, in the tree classification problem the use of chain codes is not a good choice because of the extreme irregularity of the shapes. Note that shape extreme irregularity is, however, typical of many image classification tasks, especially in biology, such as lichen classification [13, 16] and bacteria colonies recognition.

About 450 experiments were carried on. The proto-transformations were obtained using only rotations that are multiple of $\pi/2$. The best results were obtained using 4 transformations with a side scaling factor equal to 0.35. The classifier was implemented as a feed-forward neural network, trained by means of the Scaled Conjugate Gradient (SCG) rule [15]. The networks had about 100 hidden neurons and were trained in a number of epochs in the range [500, 1000], then training took always place in a few minutes.

Table 1. Recognition rates obtained encoding the trees with XFF.

class	(a)	(b)	(c)	(d)	(e)	(f)
elm	100 %	88.61 %	86.00 %	90.18 %	100 %	56.00 %
oak	100 %	91.38 %	95.34 %	83.64 %	100 %	76.00 %
lime-tree	100 %	75.38 %	67.34 %	61.45 %	100 %	60.00 %
average	100 %	85.12 %	82.89 %	78.42 %	100 %	64.00 %

The outcomes of the experiments are reported in Table 1. 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 are not averages. They were reported just to show that in some cases a perfect recognition was obtained also for class "lime-tree", which is always more difficult to capture than the others. Column (b), instead, 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%. Recognition rates in between 75% and 80% were achieved using smaller learning sets. Columns (c) and (d) report the average results obtained when encoding the trees respectively by means of 3 and 5 transformations. Different numbers of hidden neurons were tried also in this case; in particular, when 5 transformations are applied, the best results were obtained using 200 hidden neurons. A 100% recognition rate was always achieved on the learning set, see column (e), independently from the setup. Column (f) reports the results obtained encoding the instances by means of *enc* [9], a grey-scale image compression tool.

A main observation is that *12 numbers* resulted to be 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 a sufficient proof of the *power of discrimination* that can be summarized by an IFS. The optimal performance obtained in some cases, then, strengthens this result. We believe that higher average recognition rates could be achieved by widening the range of transformations; nevertheless, relaxing the constraints that were imposed would also increase the computational complexity of the search. This is the same problem that was faced by fractal compression system developers.

5 Conclusions

Even though IFSs have been extensively studied and used for image compression, they were never taken into account as a means for characterizing visual representations of objects for recognition purposes. In the literature, in fact, there are very few examples of works where fractals have been used in image classification tasks. The only ones that we could find are [10], where a neural network is trained to distinguish fractal images from non-fractal images, [7], where fractals are used to build the belief functions of a Bayesian classifier, and [12], where a metric for planar self-similar forms is proposed.

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. Note that image classification is a natural application of similar metrics, however, in order to use this metric in recognition tasks some models are to be given for comparison. Furthermore, it is necessary to find a technique for producing IFS encoding for the learn/test instances. Only then, it would be possible to apply a metric to compute the similarity between instances and/or between instances and prototypes.

The greatest novelty of our work is that we focused on finding the IFS representations of a set of given images and, then, we used these representations to learn a classification knowledge. Then, by attacking the problems of image feature extraction and classification knowledge acquisition, our method is to some extent *complementary* to [12].

With our work we have shown that IFS encodings can be used to discriminate between images of different kinds of objects and can, then, be used also in classification tasks. The only problem is to find good techniques for making fractal feature extraction automatic. In fact, as shown in [1], when a perfect encoding is produced the recognition rates achieved are optimal. In order to show that such algorithms *can* be developed, we implemented XFF, a simple fractal feature extraction system, that can be applied to any isolated shape. Experiments showed that recognition rates obtained using XFF are twenty percentage points better than those obtained using fractal compression systems (see table 1, column (f)). Furthermore, the encodings are generally much more compact.

We think that these results are a sufficient motivation for a deeper investigation of IFS-based fractal feature extraction methods, with particular care 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 to use, the recognition rates were optimal [1]. The new open problem is, then, to find an *optimal criterion* for guiding transformation selection. We think that this problem is worthwhile a deeper investigation because of the very appealing characteristics that are shown by IFS-based fractal features. One of the main characteristics is that their use reduces both the number of the 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 (whose size is 320×240) was represented by just 12 numbers. Moreover, the number of fractal features used to represent a shape is *independent* from the size, being related to the intrinsic shape complexity only. These two characteristics make IFS representations extremely interesting in the case of image classification. In fact, according to the results of StatLog [14], many learning algorithms *cannot* deal with applications having hundreds of dimensions while those that can, take a *too long 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.

References

1. M. Baldoni, C. Baroglio, D. Cavagnino, and G. Lo Bello. Extraction of Discriminant Features from Image Fractal Encoding. In M. Lenzerini, editor, *Proc. of AI*IA 97: Advances in Artificial Intelligence*, volume 1321 of *LNAI*, pages 127–138. Springer-Verlag, 1997.
2. M. Barnsley. *Fractals Everywhere.* Academic Press, San Diego, 1988.
3. M. Barnsley and S. Demko. Iterated function systems and the global construction of fractals. In *The Proc. of the Royal Society of London*, vol. A399, 243–275, 1985.
4. M. Barnsley and L. P. Hurd. *Fractal Image Compression.* AK Peters, Ltd., Wellesley, Massachusetts, 1993.
5. P. Besl and R. Jain. Three dimensional object recognition. *ACM Computing Surveys*, (17):75–154, 1985.
6. R. Chin and C. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, (18):67–108, 1986.
7. A. M. Erkmen and H. E. Stephanou. Information Fractals for Evidential Pattern Classification. *IEEE Trans. on Sys., Man, and Cyb.*, 20(5):1103–1114, 1990.
8. K. Falconer. *Fractal Geometry, Mathematical Foundations and Applications.* John Wiley & Sons Ltd., Chichester, UK, 1990.
9. Y. Fisher. *Fractal Compression: Theory and Application to Digital Images.* Springer Verlag, New York, 1994.
10. B. Freisleben, J. H. Greve, and J. Löber. Recognition of Fractal Images using a Neural Network. In *New Trends in Neural Computation, IWANN '93*, volume 686 of *LNCS*, pages 632–637. Springer-Verlag, 1993.
11. R. C. Gonzales and R. E. Woods. *Digital Image Processing.* Addison - Wesley Publishing Company, 1992.
12. A. Imiya, Y. Fujiwara, and T. Kawashima. A Metric of Planar Self-Similar Forms. In P. Perner, P. Wang, and A. Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition, SSPR'96*, vol. 1121 of *LNCS*, pages 100–109. Springer-Verlag, 1996.
13. A. De Marchi and Cassi. Modelli matematici di colonizzazione lichenica ottenuti mediante l'esame di equazioni frattali. *Notiziario della Società Lichenologica Italiana*, 5, 1992.
14. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood Series in AI. Prentice Hall, 1994.
15. M. F. Moller. A scaled conjungate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525:533, 1993.
16. D. R. Morse, J. H. Lawton, M. M. Dodson, and M. H. Williamson. Fractal dimension of vegetation and the distribution of anthropod body lengths. *Nature*, 314:731–733, 1985.
17. L. O'Gorman. Basic techniques and symbol-level recognition – an overview. In K. Tombre R. Kasturi, editor, *Graphics Recognition, Methods and Applications*, volume 1072 of *LNCS*, pages 1–12. Springer-Verlag, 1995.
18. A. P. Pentland. Fractal-Based Description of Natural Scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, (PAMI-6):661–674, 1984.