Tamil Online handwriting recognition using fractal features

Rituraj Kunwar and A G Ramakrishnan
MILE Lab, Dept of Electrical Engineering, Indian Institute of Science, Bangalore.

Abstract

We present a fractal coding method to recognize online handwritten Tamil characters and propose a novel technique to increase the efficiency in terms of time while coding and decoding. This technique exploits the redundancy in data, thereby achieving better compression and usage of lesser memory. It also reduces the encoding time and causes little distortion during reconstruction. Experiments have been conducted to use these fractal codes to classify the online handwritten Tamil characters from the IWFHR 2006 competition dataset. In one approach, we use coding and decoding process. A recognition accuracy of 90% has been achieved by using DTW for distortion evaluation during classification and encoding processes as compared to 78% using nearest neighbor classifier. In other experiments, we use the fractal code, fractal dimensions and features derived from fractal codes as features in separate classifiers. Whereas the fractal code was successful as a feature, the other two features are not able to capture the wide within-class variations.

Introduction

Fractal codes are the compressed representation of patterns, based on iterative contractive transformations in metric spaces proposed by Barnsley. A simplified version of the fractal block coding technique for digital images has been used to encode the 1-D ordered online handwritten character patterns. A novel partitioning algorithm has been proposed to reduce the computation complexity of encoding and decoding, with a minor fall in recognition accuracy.

Building fractal codes for handwritten characters

We need to find the collection of affine transforms of the online handwritten character. The raw online handwritten character is first preprocessed using three steps: (i) smoothing (ii) re-sampling the variable number of points in each character to 60 points. (iii) normalizing the x and y coordinates between 0 and 1. The

handwritten character locus is divided into non-overlapping range segments. Each range segment has a fixed number of points (R) in it. Last point of each range becomes the first point of the next range, except in the case of the last range.

Creating a pool of domain segments

The domain pool is formed for each character locus. Domain pool is the collection of all possible domain segments. The number of points in each domain segment is chosen to be double that in each range segment, D = 2R. Domain pool can be obtained by sliding the window containing D points at a time. The window is first located at the beginning of the stroke. The window is moved along the stroke by δ points, in such a way that it does not cross the end point of the stroke. The step δ has been chosen as R/2 in our experiments.

Constructing transformed Domain pool

Transformed domain pool is constructed by multiplying each domain segment with the eight isometrics that involve reflection and rotation about different axes. To begin with, each domain is translated to its centroid and scaled down by the contractivity factor (s=0.5). The following transformations are then applied to each of the candidate domain segment.

$$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix},\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix},\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix},\begin{bmatrix}0 & 1\\-1 & 0\end{bmatrix},\begin{bmatrix}-1 & 0\\0 & 1\end{bmatrix},\begin{bmatrix}0 & -1\\1 & 0\end{bmatrix},\begin{bmatrix}-1 & 0\\0 & -1\end{bmatrix},\begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix}$$

The above transformations produce a whole family of geometrically related domain segments. In domain pool, matching blocks will be looked for encoding the online handwritten character.

Searching for the most similar domain segment for each range: Each affine transformed domain segment is re-sampled into R points and then its centroid is translated to that of the concerned range segment. Distance between them is found. Similarly, distances w.r.t to the all the domain segments is calculated. The most similar domain segment corresponding to each range segment is identified and fractal code is stored corresponding to the particular range. The fractal codes are similarly obtained for all the online handwritten characters.

Fractal codes corresponding to each range segment consists of (1) the range segment index, (2) the range segment centroid, (3) index of the most similar domain segment and (4) the index of transformation used out of the 8 transformations.

Issue related to constructing fractal codes

The whole character is divided into range segments of equal number of points. Smaller the number of points in each range segment, the more minutely we can capture the complexity in any region of the character. The number of range segments per character is thus inversely proportional to the number of points in each range. Again, the encoding speed is inversely proportional to the number of range segments per character. It has been noted that there are certain region in a character where the curliness is minimal so in those area the range segment size could be increased still encoding the region precisely.

Steps to encode a handwritten character where the number of points in each range is variable:

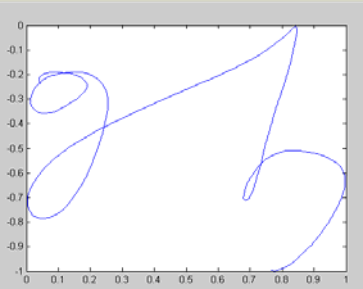

Fig 1. Tamil handwritten character ‘aa’

Cumulative angle ‘θ_C’ is calculated starting from the first point and traversing the character stroke till it crosses an empirically set threshold of θ_T. Smaller the threshold, finer is the encoding. Figure 1 shows a sample of the handwritten character /aa/ in Tamil. Figure 2 shows the effect of the choice of the angle threshold on the reconstruction error.

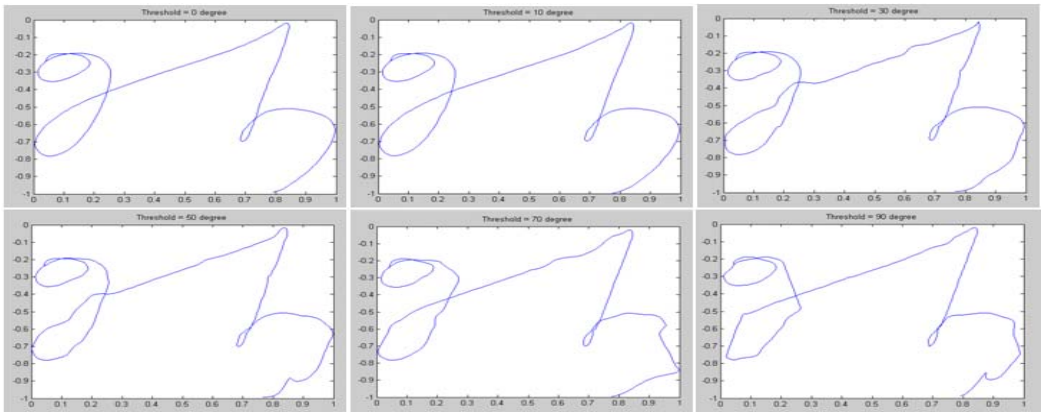

Fig 2. Illustration of the distortion in reconstruction with different angle thresholds ϕ_T. Here reconstruction is performed from the fractal codes of the character /aa/ shown in Fig 1. For encoding, different values of range sizes are used, namely, 4, 8, 12, 16 and 20.

Algorithm for partitioning

1. Domain pool of different sizes (namely 8, 16, 24, 32, 40) are constructed corresponding to the range sizes of 4, 8, 12, 16 and 20. By size, we mean the number of points in each domain.
2. Start from the first point and move along the character from one point to the next and calculate the cumulative change in angle θ_C.
3. The No. of points (K) till the point penultimate to the one, where θ_C crosses the threshold ϕ_T is noted.
4. The range size closest to and less than K is chosen. The most suitable domain is chosen from the corresponding domain pool and the fractal codes are stored.
5. The last point of the present range is then considered as the first point of the new range and the process repeats starting from step 2.

Along with the fractal codes, the size of the range chosen is also stored. If the end point is reached with $\theta_C < \phi_T$, then step 4 is followed, where K includes the last point also since $\theta C < \phi T$). If at the end, few points are left which is less than the smallest range, they are discarded else step 4 is repeated.

Algorithm for reconstruction: Banach's contractive mapping theorem states: "If a contractive mapping 'W' (which are the fractal codes here) is defined, then iterative

application of the mapping on any sequence of the same space will lead to a Cauchy's sequence which will converge to a fixed, unique point.

CASE I: Range having fixed number of points

A random initial pattern having the same number of points is taken or generated. A domain pool of size double that of the range is created in a manner similar to the encoding process. First fractal code is taken corresponding to the first range of the pattern, and operations are performed on the corresponding domain indicated by the domain index in the code of first range. The indicated domain segment's origin is shifted to its origin and then it is scaled down by the contractivity factor (0.5 here). Then the affine transformation as indicated in the code is applied on the scaled domain segment. Finally the transformed domain's centroid is shifted to the range segment centroid as present in the fractal code. The above steps are repeated to decode all the range segments. Then the whole decoded locus is smoothened. The above steps are repeated till the termination condition is satisfied to finally converge to a fixed and unique pattern. Termination condition could be (i) an empirically set fixed number of iterations, sufficient for convergence or (ii) minimal or no distortion between two consecutive patterns produced by 2 consecutive iterations.

CASE II: Range having variable number of points in each range

In this case, multiple domain pools are created out of the random pattern taken for the reconstruction. Using the extra information given in the fractal code pertaining to the range size to be chosen so that domain segment is picked up from the appropriate domain pool. The rest of the steps are same as the case for the reconstruction with fixed range size. Using above two methods, fractal codes of any give pattern can be created and the same pattern could be decoded using any random pattern after applying this reconstruction algorithm iteratively for few times.

1. Classification by using fractal codes in construction and reconstruction:

The above fractal encoding and decoding method has been used for classification of characters. The assumption behind this classification is that if a sample of a class (say /a/) is encoded and fractal codes are obtained. The following process of reconstruction if started in 2 ways i.e. firstly by applying the reconstruction algorithm

iteratively on a random pattern of any different class (anything other than ‘A’) and secondly doing the same on any random pattern of the same class (i.e. ‘A’) then the distortion between the initial pattern and the pattern obtained after first iteration of reconstruction is relatively much more in the first case than in the second. The reason behind this is that reconstruction process leads to convergence to the pattern whose code is used for reconstruction. And since the class of the initial pattern in the second case and the fractal code is same, the distortion in the second case is smaller than the first case.

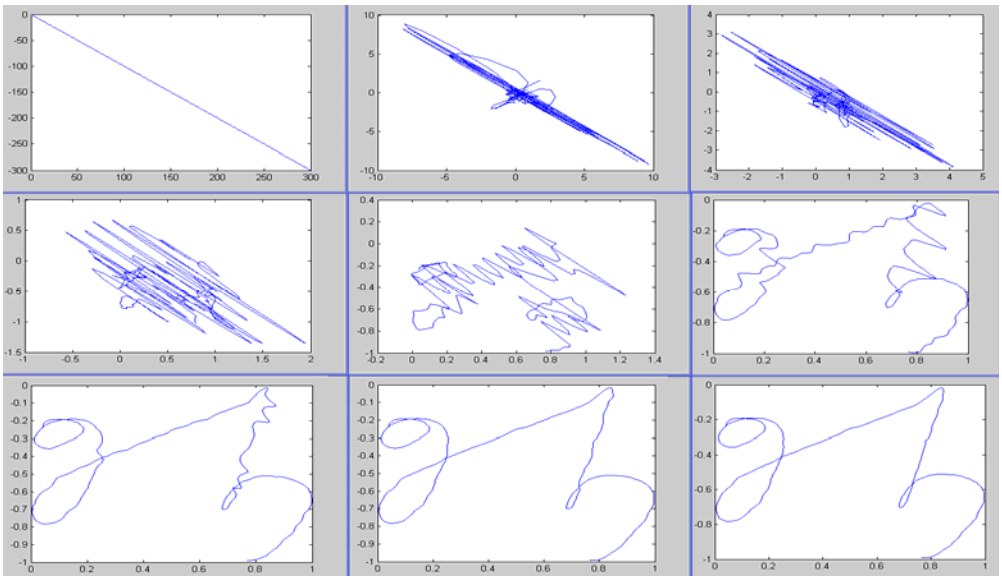

Fig 3. In the above image, reconstruction process is shown which starts from a random straight line and finally converges to a pattern which is very close to the original pattern after 8 iterations. The original pattern (in Fig 1) was encoded using different range sizes of 4, 8, 12, 16 and 20 with the threshold angle of 30 degrees.

Character classification using fractal codes:

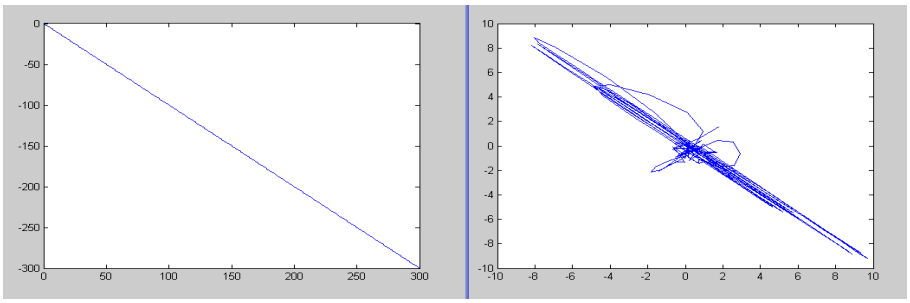

Fig. 4. The above image shows the distortion created, when an iteration of reconstruction was performed. This shows that the distortion is huge if the

starting pattern (above left) is very different from the original pattern, whose fractal codes are used for reconstruction (in this case Fig 1).

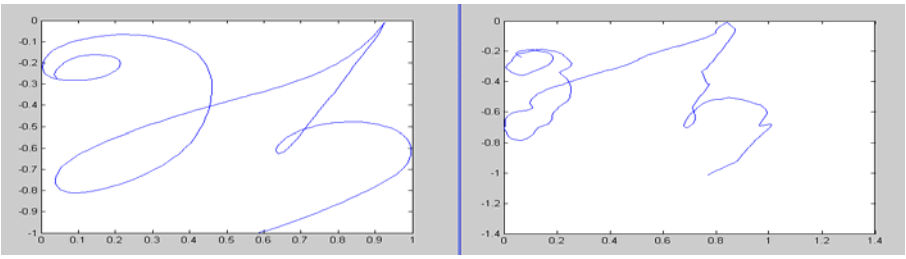

Fig. 5. The above image shows the distortion created, when an iteration of reconstruction was performed. This shows that the distortion is much less if the starting pattern (above left) is not very different from the original pattern, whose fractal codes are used for reconstruction (in this case Fig 1).

Classification Algorithm

In the present research, fractal codes of 'N' samples of entire 156 classes are computed and stored. To classify a test sample, following steps are taken.

a) An iteration of the reconstruction algorithm is applied on the test sample using the fractal code of each sample of each class.

b) The distortion 'D' is calculated between the initial pattern and the pattern obtained after one step of reconstruction. Thus the distortion matrix of size 156*N is obtained.

c) The row number of the minimum value of the distortion is found from the distortion matrix and assigned to the test sample.

Distortion evaluation:

The distortion between two patterns can be evaluated by finding the distance between them. The distance between 2 patterns is measured by Nearest Neighbor (NN) method. The issue with NN is that the matching is done point by point which increases the distance unusually and thus decreases the classification accuracy (evident from the result table). This drawback in distance evaluation of NN is addressed by DTW pattern matching which is more intuitive. This intuitive matching takes place because DTW matches similar subsections of the patterns thus producing a reasonable distance between patterns. This method hence produces a remarkable increase in accuracy as shown Table 1.

Classification using the fractal codes as a features in the Nearest Neighbor (NN)

In this method, the fractal codes are used as features and fed into a NN classifier. An accuracy of approximately 65% is obtained.

Classification using fractal dimension or features derived from the fractal codes

Fractal dimension is a unique identity of any pattern or object. However, because of the mere nature of handwritten character recognition (i.e. large variation within every class), it fails completely to classify any random sample. The features derived from the fractal codes like MMVA and DRCLM, though successful in problems like face recognition and signature verification, fail in recognizing handwritten characters.

Results

Table 1: Results show how the DTW comparison during reconstruction impacts the recognition accuracy.

Fixed Range Size	DTW used?	No. of Training samples	No. of Testing samples	Accuracy (in %)
4	No	20	50	78.9
4	Yes	20	50	90.4

Table 2: Results show the efficacy of the Partitioning algorithm in improving the efficiency of the recognition system (in terms of time) with marginal drop in accuracy. Variable range sizes used (4, 8, 16, 24 and 32). No. of training and testing samples used are 5 and 30, respectively, No. of classes used is 156.

Threshold angle (degree)	DTW used for distortion evaluation?	Encoding time per sample (sec)	Accuracy (in %)
0	Yes	31	90.44
10	Yes	22	86.43
30	Yes	12	85.04
50	Yes	10	83.55
70	Yes	8	81.60
90	Yes	6	80.87

Acknowledgment: The authors thank Technology Development for Indian Languages (TDIL), DIT, Government of India for funding this research, as part of the research consortium on Online handwriting recognition of Indian languages.

References

1. M. F. Barnsley, Fractals everywhere, New York: Academic, 1988.
2. T. Tan and H. Yan, Face recognition by fractal transformations, IEEE ICASSP, 6:3405-3408, 1999.
3. Mozaffari S., Faez K. and Faradji F, One Dimensional Fractal Coder for Online Signature Recognition, IEEE ICPR, 2:857- 860, 2006.