OCR in a Hierarchical Feature Space

Jaehwa Park,
Venu Govindaraju, *Senior Member*, *IEEE*, and
Sargur N. Srihari, *Fellow*, *IEEE*

Abstract—This paper describes a character recognition methodology (henceforth referred to as *Hierarchical OCR*) that achieves high speed and accuracy by using a multiresolution and hierarchical feature space. Features at different resolutions, from coarse to fine-grained, are implemented by means of a recursive classification scheme. Typically, recognizers have to balance the use of features at many resolutions (which yields a high accuracy), with the burden on computational resources in terms of storage space and processing time. We present in this paper, a method that adaptively determines the degree of resolution necessary in order to classify an input pattern. This leads to optimal use of computational resources. The *Hierarchical OCR* dynamically adapts to factors such as the quality of the input pattern, its intrinsic similarities and differences from patterns of other classes it is being compared against, and the processing time available. Furthermore, the finer resolution is accorded to only certain "zones" of the input pattern which are deemed important given the classes that are being discriminated. Experimental results support the methodology presented. When tested on standard NIST data sets, the *Hierarchical OCR* proves to be 300 times faster than a traditional K-nearest-neighbor classification method, and 10 times faster than a neural network method. The comparsion uses the same feature set for all methods. Recognition rate of about 96 percent is achieved by the *Hierarchical OCR*. This is at par with the other two traditional methods.

Index Terms—Pattern recognition, character/digit recognition, multiresolution, feature space, hierarchical classification, recursion.

1 Introduction

MOST character recognition techniques described in the literature use a "one model fits all" approach, i.e., a set of features and a classification method are developed and every test pattern is subjected to the same process [1], [2], [3], [4], irrespective of the constraints present in the problem domain.

In this paper, we present a dynamic character recognizer which uses a hierarchical feature space to achieve multiresolution while circumventing the disadvantages cited above. The recognizer begins with features extracted in a coarse resolution and focuses on smaller subimages of the character on each recursive pass, thus effectively working with a finer resolution of a subimage each time till the classification meets acceptance criteria. In other words, successive passes render the same features at a finer resolution.

The subimages are selected by an approach called *gaze planning*. It guides the successive passes through a series of subimages where it believes the features of interest are present. This is done in an adaptive and dynamic manner. Depending on the classes competing for the top spot, different subimages might be selected at any given stage. For example, if "3" and "5" are the top contenders at a particular recursive stage, the features from the upper zone of the test pattern holds greater discriminatory power and should be examined at a finer resolution.

By beginning with a global view and progressively increasing the resolution by subimages of the test pattern, we are able to accomplish an adaptive classification with optimal resource usage in terms of storage space and computational cycles of recursion. Further, the feature vector has more information from those subimages that are deemed to be more important than others at any given recursive cycle.

Experimental results are provided to support the validity of our methodology.

2 Hierarchical Classification

Let $\mathbf{x} = [x_0, x_1, \cdots, x_{N_x-1}]$ be a binary feature vector with dimension N_x and $C = \{c_0, c_1, \cdots, c_{N_c-1}\}$ be a set of N_c classes, then by Bayes' rule the a posteriori probability of c_i given $\mathbf{x}$ is given by

$$P(c_i|\mathbf{x}) = \frac{p(\mathbf{x}|c_i)P(c_i)}{P(\mathbf{x})} = \frac{p(\mathbf{x}|c_i)P(c_i)}{\sum_j p(\mathbf{x}|c_j)P(c_j)}, \tag{1}$$

where P(c_i) is the a priori probability of class c_i and $p(\mathbf{x}|c_i)$ is the class conditional probability. A vector code space $V = \{0, 1, 2, \cdots 2^{N_x} - 1\}$, can be defined where each point in V represents a different binary bit pattern of size N_x, thus mapping every binary feature vector $\mathbf{x}$ onto a unique point $v \in V$ in the vector code space. The probabilities $P(c_i|v)$, denoted as $P_i[v]$ can be stored in a look-up table of size

$$T(N_x, N_c) = |V| \times N_c = 2^{N_x} \times N_c \tag{2}$$

during training, which makes for a very efficient program. For example, $T(9, 10) = 5,120$. Assuming 4 bytes to represent floating-point numbers, the table would need 20,480 bytes. However, recognition accuracy with $N_x = 9$, $N_c = 10$ is poor for handwritten digit recognition; e.g., we obtained 54 percent accuracy (top choice) for the 10-class digit recognition problem, which is unacceptable in practice. The number of features and, hence, the feature space dimensionality needs to be increased to achieve accurate classification of handwritten characters, especially when the data is noisy. For example, a method described in [5] uses gradient, concavity, and structural features (known as GSC) which are features at three levels of granularity. GSC has good recognition accuracy ($\geq$ 97 percent). It requires $N_x = 512$ and $T(512, 10) = 2^{512} \times 10$, the table for which would be impossible to create or store.

There are many methods described in the literature for reducing the storage requirement by using a set of coefficients, such as the Rademacher-Walsh and Bahadur-Lazarsfeld coefficients [6]. In particular, Bailey and Srinath use a method of mapping the code space into 30 real coefficients to obtain satisfactory recognition performance for handwritten digits. [7].

2.1 Image Hierarchy

An alternative approach to the trade-off dilemma (between accuracy and storage) is to keep N_x small to begin with and focus on "important" subimages of the original image. Subimages are obtained at any stage by using a hierarchical division rule such as quad tree or quin tree which subdivide the image into four or five subimages, respectively (Fig. 1). If N_D denotes the maximum depth of the tree generated by subdivisions and the maximum number of branches per subdivision is denoted by N_B, then for a quad tree $N_B = 4$ and for a quin tree $N_B = 5$.

Let $I(d, l)$ represent the subimage of the lth node of a tree at depth d, where $d \in \{0 \ldots N_D - 1\}$ and $l \in \{0 \ldots N_B^d - 1\}$. Then the total number of nodes in the tree is

$$S(N_D, N_B) = \sum_{d=0}^{N_D-1} (N_B)^d. \tag{3}$$

• *The authors are with the Center of Excellence for Document Analysis and Recognition, the Department of Computer Science and Engineering, State University of New York at Buffalo, Amherst, NY 14260. E-mail: {jaehwap, govind, srihari}@cedar.buffalo.edu.*

Manuscript received 9 July 1998; accepted 29 Dec. 1999.
Recommended for acceptance by R. Plamondon.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107835.

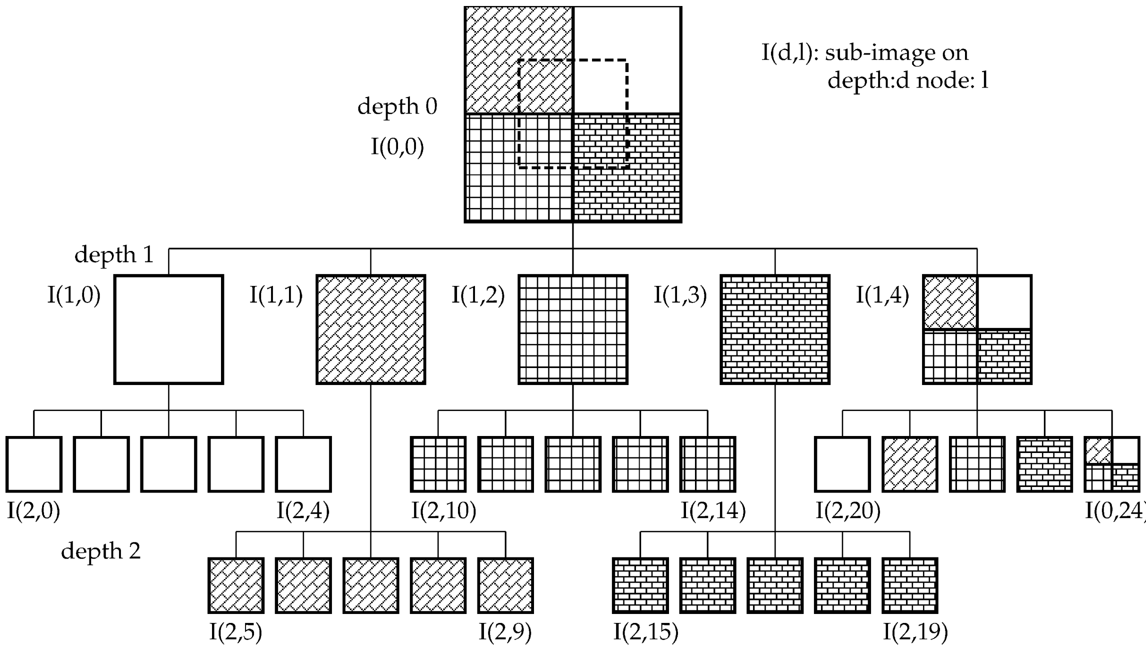

Fig. 1. Subimage division using Quin tree. There are nodes at depth 0,1, and 2 resulting in $S(2,5)$ = 31 subimages. In this case, Q_i^2 is the product of 25 probabilities.

Features are computed for each node that is visited. Let $v(d,l)$ represent the feature vector corresponding to subimage $I(d,l)$. Features extracted in earlier passes correspond to coarse global features while those in later passes provide local features. The actual feature extraction process and the type of the features remains the same, however, the features are now computed from a smaller portion of the image. This adds to some additional storage space increasing the probability table size to $T(N_x, N_c) \times S(N_D, N_B)$. For example, with $S(4,5) = 156$, and $T(9,10) = 5,120$, the probability table size would increase to 798,720 which is still practical.

Let $c_i(d,l)$ represent the subclass of c_i corresponding to $I(d,l)$. Then the a posteriori probability of class $c_i(d,l)$ given $I(d,l)$, is $P_i[v(d,l)]$, where $v(d,l) \in V$ represents the corresponding vector code. This value is obtained from (1), where $\mathbf{x}$ is the feature vector computed from the subimage, $I(d,l)$ and c_i is replaced by $c_i(d,l)$.

A confidence function Q_i^d for class c_i is defined as the product of the a posteriori probabilities corresponding to all subnodes belonging to layer d,

$$Q_i^d = \prod_{l=0}^{(N_B)^d - 1} P_i[(v(d,l)]. \tag{4}$$

The confidence Q_i of a class c_i considering all the layers of the tree is defined as the product of all the a posteriori probabilities of all its subclasses corresponding to all subnodes processed up to that time, i.e.,

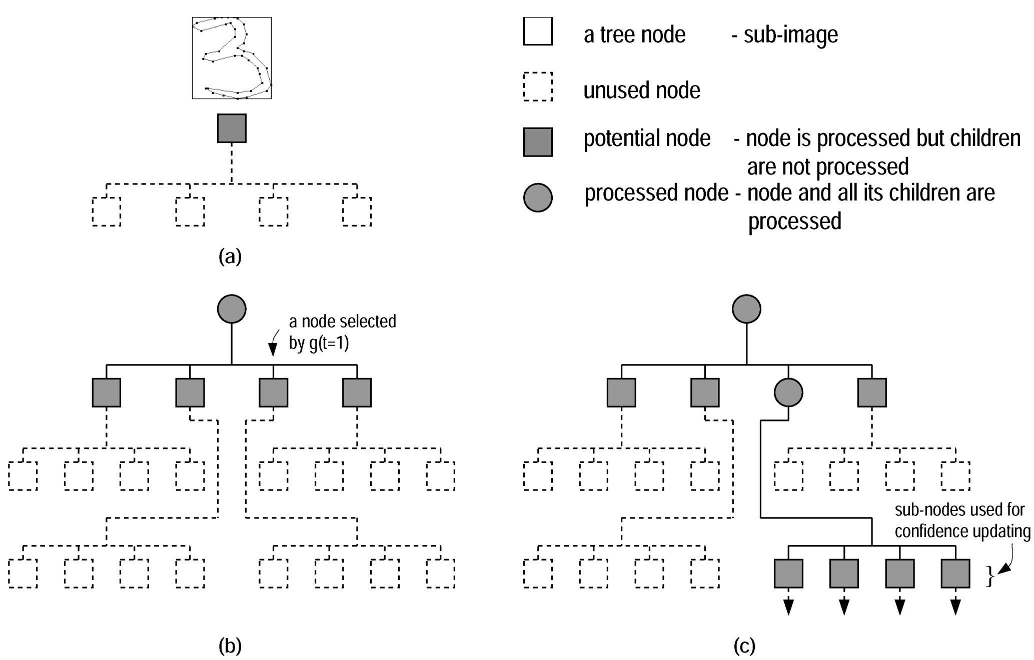

Fig. 2. Confidence update recursion using a quad-tree; (a) starting from root, (b) after one recursive pass, and (c) after two recursive passes.

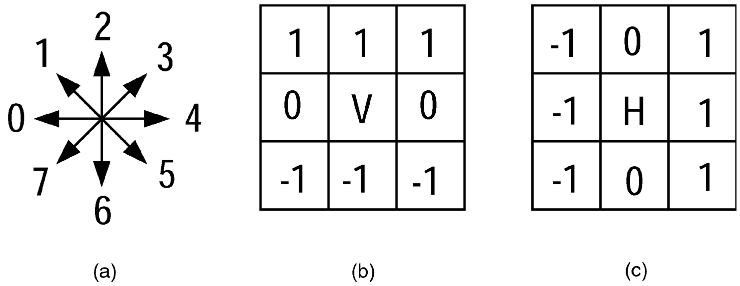

Fig. 3. Contour representation; (a) contour slope division, (b) vertical weight mask, and (c) horizontal weight mask.

$$Q_i = \prod_{d=0}^{N_D-1} Q_i^d = \prod_d \prod_l P_i[v(d,l)]. \quad (5)$$

If the classes and subclasses are statistically independent then Q_i represents the class a posteriori probability. This is clearly not the case (since the subimages are derived from the parent image). The value of Q_i lies in the interval [0,1].

Equation (5) requires that the probability of each class is computed as a product of $S(N_D, N_B)$ probabilities. In the next section, we discuss a method of reducing the number of product terms.

2.2 Gaze Planning

Gaze planning is a means of expanding only some of the nodes in the tree. A node is expanded only if its a posteriori probabilities do not satisfy a termination criterion e.g., the highest class probability is not sufficiently larger than the second highest class probability. We can fix the number of nodes whose probabilities are used to update confidences on each pass and we limit the selection of new nodes to a subset of *potential nodes* used in the previous pass but has unprocessed child nodes or subnodes (Fig. 2). The intermediate confidences can be computed recursively at pass t as follows: We use the notation $Q_i(t)$ to indicate the confidence computations when all subnodes are not used.

$$Q_i(t) = Q_i(t-1) \cdot {Q'}_i(t-1), \quad (6)$$

where ${Q'}_i(t)$ is the confidence generated from the subnodes of a node selected at time t, where

$$t \in \{0, 1, \ldots, \sum_{d=1}^{N_D-1} (N_B)^{d-1} - 1\}.$$

This process is illustrated for a quad tree in Fig. 2, where at depth 1 only one node is expanded.

Let the gaze function, $g(t)$, determine whether a node is to be examined in the next recursive pass. The gaze function $g(t)$ is designed to find the most informative node within the pool of potential nodes ready to be expanded. Let $sub(d,l)$ be a function which gives the indices of subnodes of the $(d+1)^{st}$ layer belonging to node (d,l) and $sub(d,l)|_{(d,l)=g(t)}$ gives the nodes marked (d_k, l_k) as the nodes selected for expansion. We then have

$${Q'}_i(t) = \prod_{(d_k,l_k) \in sub(d,l)|_{(d,l)=g(t)}} P_i[v(d_k, l_k)]. \quad (7)$$

Usually a classification result of a recognizer is accepted based on the confidence of the top choice and its separability from the second choice [5], [8]. Using the same criterion, normalized confidence $\kappa_i(t)$ of a class c_i at recursion time t is defined by the ratio of the current (intermediate) confidence and the maximum possible confidence of the class regardless of the specific feature vector taken from the look-up tables of the all corresponding processed and potential nodes.

$$\kappa_i(t) = \frac{Q_i(t)}{R_i(t)} \quad (8)$$

$R_i(t)$ is the product of the maximum possible probability of class c_i in any vector code through the selected node sequence of processed and potential nodes. The node sequence is the accumulated outputs of the gaze function g until pass $t-1$. Recursively,

$$R_i(t) = R_i(t-1) \cdot {R'}_i(t-1), \quad (9)$$

where

$${R'}_i(t) = \prod_{(d_k,l_k) \in sub(d,l)|_{(d,l)=g(t)}} max(P(c_i | v(d_k, l_k)))|_{\forall v(d_k,l_k)}. \quad (10)$$

A recursive equation of normalized confidence for class c_i is given as

$$\kappa_i(t) = \kappa_i(t-1) \cdot {\kappa'}_i(t-1), \quad (11)$$

where ${\kappa'}_i(t)$ is the normalized confidence of newly added branches of node (d,l),

$${\kappa'}_i(t) = \frac{{Q'}_i(t)}{{R'}_i(t)}. \quad (12)$$

From (7) and (11), the intermediate confidence and normalized confidence of each class is obtained at every pass. After ordering of classes using normalized confidences, a separability, $\sigma(t)$, of the top two choices is given by

$$\sigma(t) = \frac{Q_{top}(t)}{Q_{second}(t)}, \quad (13)$$

where class c_{top} has the highest confidence at time t.

For driving the gazing function, local separability which represents the support of current top ranked class at time t is extracted in each edge node which has unexamined subnodes. If class c_{top} is the top choice and class c_{second} is the second choice in

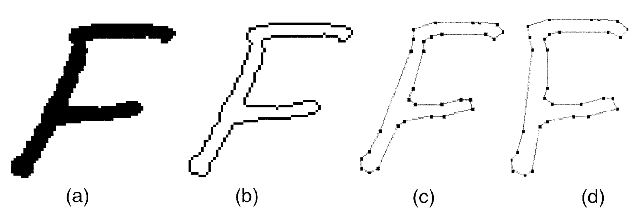

Fig. 4. Contour representation and preprocessing; (a) input image, (b) pixel-based contour representation, (c) piecewise linearized contour, and (d) piecewise linear contour representation after slant correction.

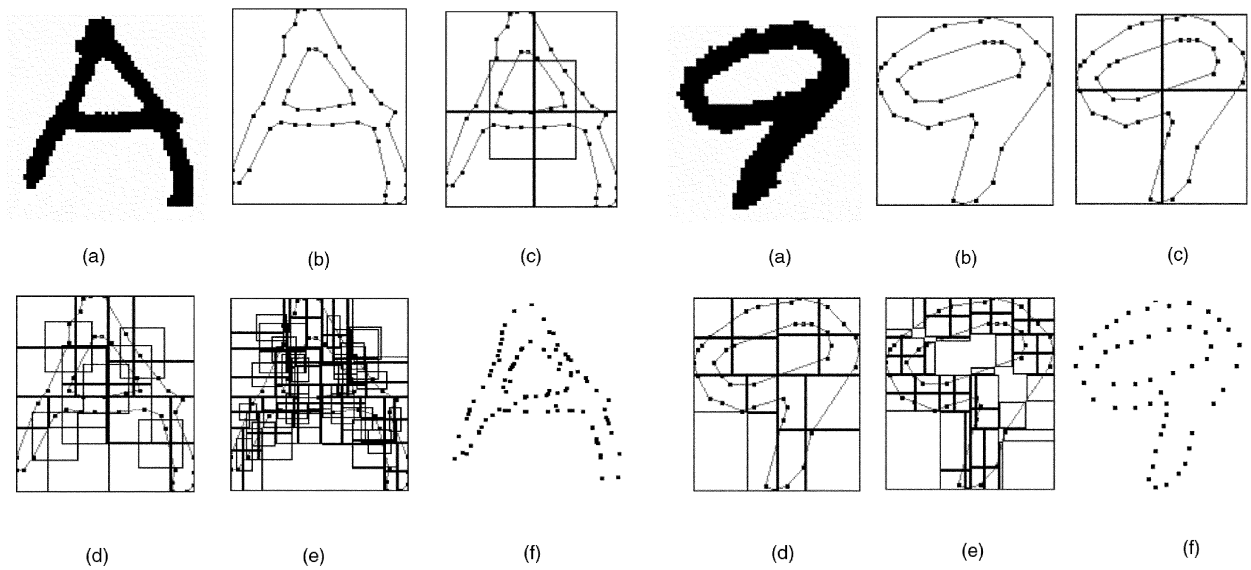

Fig. 5. Character image and subimages of quin tree (class A) and quad tree (class 9): (a) o; inal image, (b) piecewise linear contour, (c) layer 1, (d) layer 2, (e) layer 3, and (f) reconstruction of contour image from critical points of final layer.

intermediate confidence result of time t, local separability $\sigma^{(d,l)}(t)$ of a potential node (d,l) is given by

$$\sigma^{(d,l)}(t) = \frac{P_{top}[v(d,l)]}{P_{second}[v(d,l)]}. \tag{14}$$

If the current result does not meet acceptance threshold recursive computations continue. The next node to be expanded can be determined by comparing local separability of the remaining nodes using the gazing function.

3 Implementation

A binarized character image is represented by *piecewise linear contours* using chain code [9], [10] (Fig. 3a). A window of weights is convolved with the contour component and its neighbors to obtain the curvature at a contour point (Fig. 3b and Fig. 3c). The window size is determined by a function of the height of the bounding box of the character image and the density of black pixels within the bounding box.

At each contour point, horizontal and vertical slopes are found using the masks of Fig. 3b and Fig. 3c and binarized using a threshold (0.65 in our implementation). Contour points which have high curvature (derivative of the binarized slope) are identified as critical points. Contour segments between adjacent critical points are approximated to be linear. This representation adds to the overall processing speed by reducing the total data points and also by eliminating the need for smoothing and noise removal. Fig. 4b and Fig. 4c illustrate the procedure.

3.1 Slant Correction

Continuous vertical line components (spanning several critical points) whose length is greater than a threshold are identified. Slant angle is estimated by the average of the slant angles (from the horizontal) of all vertical line components, where the angles are weighted in proportion to their length in pixels. Using the estimated slant angle, the critical points are mapped for slant correction. Fig. 4d shows the translated points after slant correction.

3.2 Subimages

A variable size rectangular grid is used to define subimages for the quin (quad) tree. The bounding box of a character image is divided into four rectangular regions (Fig. 5c). The center of mass of the contour is first computed. A vertical and horizontal line through the center of mass determines the four regions. The quin tree structure is similar with an additional fifth subregion which is located in the central area formed by joining the centers of the other four subregions. Subsequent layers are successively constructed by the same method.

We have implemented quad and quin trees with a maximum depth of 4. Fig. 5c, Fig. 5d, and Fig. 5e show subregions for each layer. Fig. 5f show reconstructed contour image using centroid points of the finest subregions at the deepest (4th) level.

3.3 Features

Histogram of gradient and moment-based projections are used as features. Feature vector for each subimage is generated after binarizations of the feature measurements which are extracted from corresponding subregions. 8-bit binary feature vector in the quad tree and 9-bit feature vector in the quin tree are used.

Contour slope angles are quantized in pairs of $\pi/4$ slots that are opposite each other labeled 0 to 3 in Fig. 6a. The angle between the

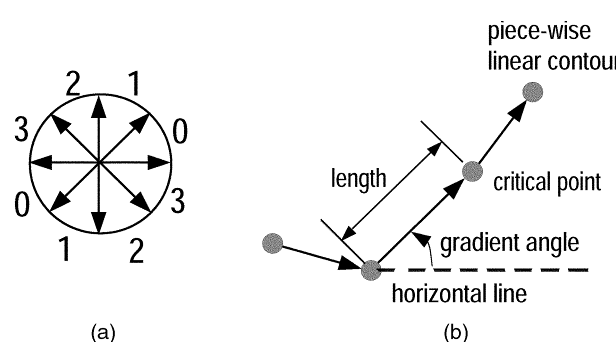

Fig. 6. Gradient feature generation; (a) gradient division and (b) gradient angle measure.

TABLE 1
Experiments are Conducted with Different Size of Training and Testing Data Sets

Data set	No. of classes	Training images	Testing images	Image source(s)	Training set		Testing set	
					top 1	top 2	top 1	top 2
1	10	25656	12242	Postal+NIST	99.5	99.9	96.2	98.8
2	10	159228	53301	NIST digit	98.6	99.7	98.2	99.5
3	26	34961	11481	NIST alpha upper-case	99.2	99.9	96.4	98.8
4	26	29059	9623	NIST alpha lower-case	98.7	99.8	90.4	97.1
5	52	64020	21104	NIST alpha full-class	93.4	98.7	78.2	94.7
6	52	64020	21104	NIST alpha folding	97.5	99.5	92.4	97.1

TABLE 2
Performance Comparison Between the Quad and Quin Trees

Experiment type	Set 1		Set 3		Set 6	
Tree type	Quad	Quin	Quad	Quin	Quad	Quin
Top 1 correct rate (%)	95.8	96.2	95.7	96.4	92.0	92.4
Top 2 correct rate (%)	98.7	98.8	98.6	98.8	97.0	97.1
Average recognition time ($msec/char$)	1.60	2.80	1.94	3.24	2.49	4.06
Lookup table size ($Kbytes$)	213	780	553	2028	1105	4056

horizontal line and the linearized contour piece between two adjacent critical points is assumed to be the slope angle of the contour segment and the length of a contour piece is taken as the Euclidean distance between adjacent critical points. (Fig. 6b).

The ratio of the total contour length corresponding to the contour segments whose slope angle falls in division i and the total contour perimeter in that subimage gives 4 gradient features.

Four (5 for quintree) moment features are obtained from the distribution of the contour within the subimages. The ratio of the length of the sum of contour segments which are present within a subimage to the total contour perimeter generate the moment features.

In our implementation, all the thresholds were set as 0.25 uniformly except the moment feature measurement of 5th subimages (0.5) for feature binarization.

4 EXPERIMENTS

Experiments were performed by using isolated digit and character images present in the NIST database 3 (SD 3) [11] and CEDAR database collected from handwritten postal addresses [12]. The images were divided into mutually exclusive training and testing sets for various experiments shown in Table 1. A 782 MIPS *SUN ULTRA 2* is the experimental platform.

From the training image set, probabilities $P(c)$ and $P(v(d,l)|c)$ in each subimage are obtained. Probabilities $P(c|v(d,l))$ in every subimage are calculated and stored in a look-up table.

The recursion of our recognition method can be terminated at any time by using various degrees of separability between top two choices as a terminal condition.

Table 2 shows a comparison using quad and quin tree methods using a maximum depth of 4. In the case of quad tree method, the subimages are 85 $(1+4+16+64)$, yielding a maximum of 680 dimensional binary feature vector when eight features are used. In the case of a quin tree method, the subimages are 156 $(1+5+25+125)$, yielding a maximum of 1,404 dimensional binary feature vector when nine features are used. Quin tree subdivision method shows better accuracy at the cost of higher storage and computational resources.

Table 3 shows a comparison of different classification methods on data set 3. To keep the comparison meaningful, the same features have been used by all three classification methods: 1) Hierarchical OCR, as described in this paper, 2) a neural network method, and 3) a nearest-neighbor method. The neural network has three fully connected layers; an input layers of 680 units, one hidden layer of 353 units, and an output layer of 26 units. The k-nearest-neighbor method described in [6] is implemented using a Hamming distance metric and six nearest neighbors. Storage requirements for Hierarchical OCR method refers to the size of the lookup table. In the case of the neural network, it refers

TABLE 3
Performance in Various Classification Methods Using Data Set 3

Classification Method	Proposed method	Neural Network	k-NN $k=6$
Top 1 correct rate (%)	95.7	96.4	95.7
Template size ($Kbytes$)	612	976	3,777
Recognition Time ($msec/char$)	1.45	11.5	384
Training Time ($hour$)	1	24	1

Proposed method operated on dynamic mode with $log(\sigma)=40$.

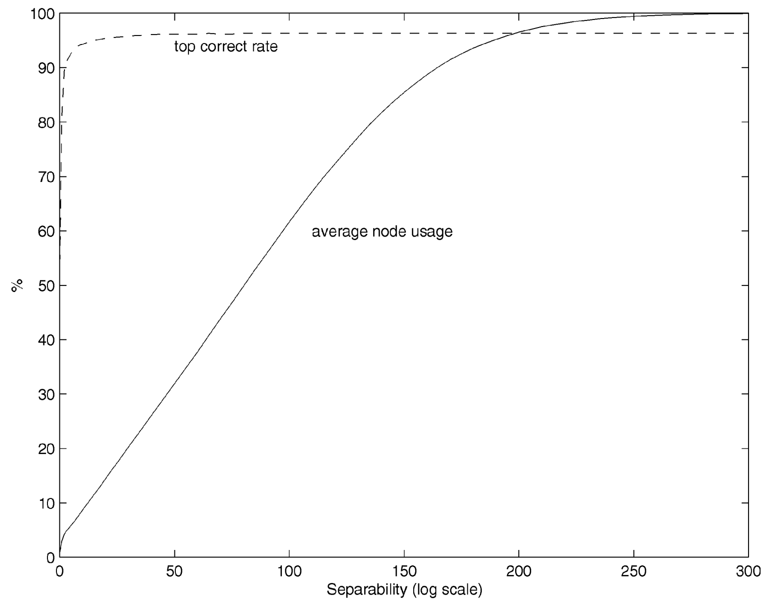

Fig. 7. Dynamic recognition performance.

to the total number of weights and in case of the nearest-neighbor classifier it refers to the prototypes in the training set. The relative merits of each of the three methods are highlighted in the table. The hierarchical OCR certainly has the speed advantage.

We have also studied the conditions for terminating recursion before it exhaustively goes through all the branches of the tree of depth 4. If the separability between the top two choices is acceptable at any stage, recursion can be terminated. The gazing function allows the sequencing of the nodes of the tree for processing. We use the criterion that the node next in sequence to be expanded is the one that allows the highest separability between the top two choices. Fig. 7 shows the use of additional nodes if a higher separability is demanded. The average number of nodes needed to achieve a separability $log(\sigma) = 5$ is one node. Fig. 8 shows how the recognition rates of all classes stabilize around $log(\sigma) = 30$.

5 DISCUSSION

Most character recognition techniques described in the literature use a "one model fits all" approach. We argue that such methods are passive. Most recognition methods that adhere to this paradigm have predetermined (hence, static) parameters such as:

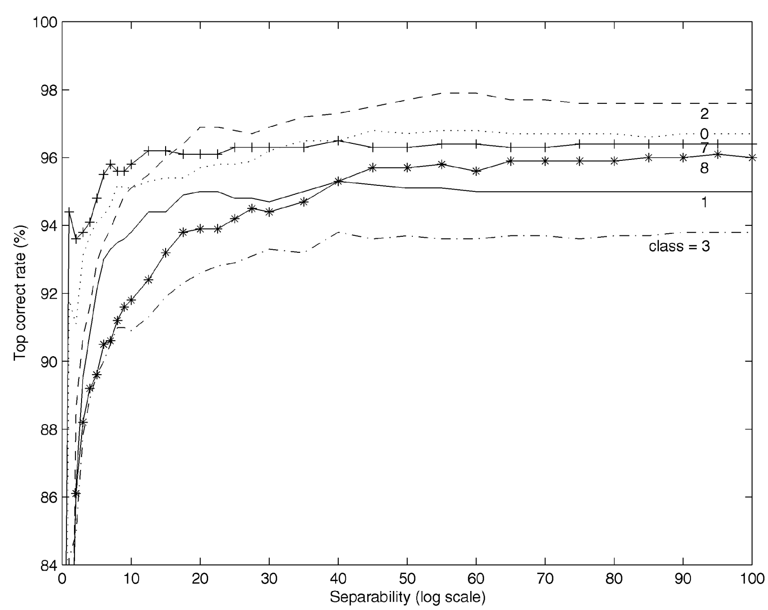

Fig. 8. Performance by class.

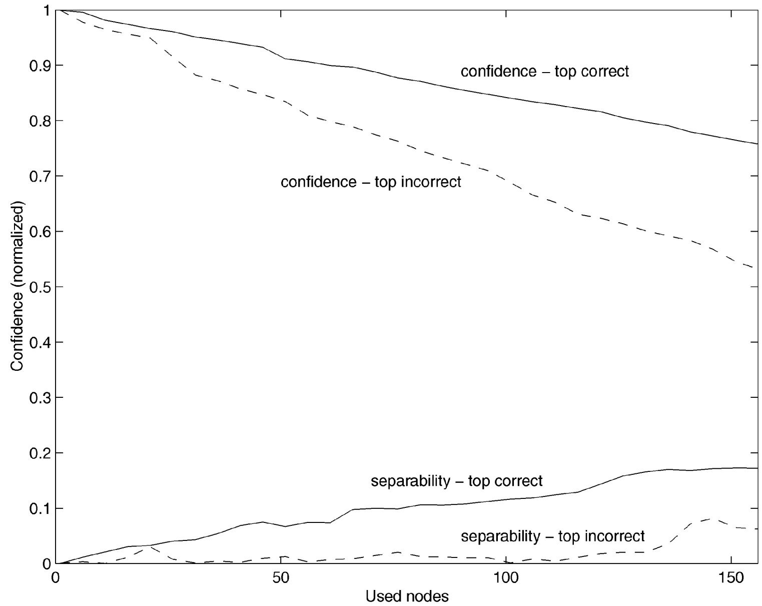

Fig. 9. Confidence and separability changes—in data set 1 with quin tree. Solid line shows successful cases and dashed line shows error cases.

1) dimension of feature vector, 2) the distance or confidence metric considered as threshold for recognition acceptance, 3) the set of classes contending to match with a test pattern (e.g., $\{0 \ldots 9\}$), and 4) equal computational resources available to every test pattern.

In very general terms, a large feature vector provides higher accuracy at the cost of computational resources. Multiresolution features have been recently used to capture features at different scales [13], [5]. However, approaches that mix features of different scales have several disadvantages as well. Correlations among different features is often unavoidable. A distance metric that captures features at all the scales and at the same time represented by a single scalar value is hard because the heterogeneous composition of the feature set makes it difficult to find an appropriate common distance metric. The computational complexity of a multiresolution recognizer such as the GSC described in [5] is known to be high because of the burden of additional features needed to accommodate the multiresolution aspect of the feature space.

We have defined the confidence of a class based on the assumption that features in subimages of an input pattern image are strongly correlated and the degree of correlation is uniquely characterized by the class. In other words, we expect the features to be highly correlated from the viewpoint of the true class and correlated to a lesser degree or not at all, when considered from the viewpoint of the "nontrue" classes. In Section 2.1 and (2), we keep updating the confidence values (Q_i) from successive recursive cycles. This would indeed lower the updated confidence of a class. However, based on the correlation assumption made, the confidences of the classes that are not true should decrease at a more rapid rate. Hence, the separability between the true class (top choice) and the nontrue class (second choice) will increase. This justifies the subsequent recursive cycles and using the product of confidences (5). The assumption of correlations is based on empirical observations. It is supported by the recognition accuracy of the method which is comparable to those described in [5], [14] that have trained and tested on similar, if not same-data sets.

Equations (13) and (14) do suggest that if an incorrect class is picked as the top choice in an early recursive cycle, the error is going to be propagated in subsequent recursive cycles. Fig. 9 illustrates that if a misclassification is made in an early recursive cycle, confidence value of the top choice (not the true class) rapidly drops and the separability becomes unstable and remains near zero even though the number of subimages used by new recursive increases. On the other hand, when the topic choice is the true class, the separability parameter holds steady.

6 Conclusion

We have presented a classification method using a hierarchical feature space which allows high recognition rates with a small set of features. Multiresolution classification is simulated by sampling smaller subimages successively in each recursion and fast processing is achieved by using vector code quantization of feature space and a look-up table.

Experiments are conducted using publicly available datasets. The recognition accuracy is at par with other methods described in the literature.

Acknowledgments

The authors would like to thank Dr. Ajay Shekhawat and Evelyn Kleinberg for valuable suggestions and comments.

References

[1] S. Mori, C.Y. Suen, and K. Yamamoto, "Historical Review of OCR Research and Development," *Proc. IEEE,* vol. 80, pp. 1,029-1,058, 1992.
[2] G. Nagy, "At the Frontiers of OCR," *Proc. IEEE,* vol. 80, pp. 1,093-1,100, 1992.
[3] C.Y. Suen, "Computer Recognition of Unconstrained Handwritten Numerals," *Proc. IEEE,* vol. 80, pp. 1,162-1,180, 1992.
[4] J. Mantas, "An Overview of Character Recognition Methodologies," *Pattern Recognition,* vol. 19, no. 6, pp. 425-430, 1986.
[5] J.T. Favata, G. Srikantan, and S.N. Srihari, "Handprinted Character/Digit Recognition Using a Multiple Feature/Resolution Philisophy," *Proc. Fourth Int'l Workshop Frontiers in Handwriting Recognition*, pp. 57-66, 1994.
[6] R.O. Duda and P.E. Hart, *Pattern Classifcation and Scene Analysis.* New York: John Wiley and Sons, 1979.
[7] R.R. Bailey and M. Srinath, "Orthogonal Moment Features for Use with Parametric and Non-Parametric Classifers," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 4, pp. 369-398, Apr. 1996.
[8] G. Srikantan, S.W. Lam, and S.N. Srihari, "Gradient-Based Contour Encoding for Character Recognition," *Pattern Recognition,* vol. 29, no. 7, pp. 1,147-1,160, 1996.
[9] H. Freeman, "Computer Processing of Line-Drawing Images," *Computing Surveys,* vol. 6, no. 1 pp. 57-97, 1974.

[10] C.-C. Lu and J.G. Dunham, "Highly Efficient Coding Schemes for Contour Lines Based on Chain Code Representations," *IEEE Trans. Comm.,* vol. 39, no. 10, pp. 1,511-1,514, 1991.
[11] M.D. Garris and R.A. Wilkinson, *NIST Special Database 3; Binary Images for Handwritten Segmented Characters (HWSC)*. New York: John Wiley and Sons, 1992.
[12] J.J. Hull, "A Database for Handwritten Text Recognition Research," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 5, pp. 550-554, May 1994.
[13] O.D. Trier, A.K. Jain, and R. Taxt, "Feature Extraction Methods for Character Recognition—A Survey," *Pattern Recognition*, vol. 29, no. 4, pp. 641-662, 1996.
[14] T.M. Ha and H. Bunke, "On-Line, Handwritten Numeral Recognition by Perturbation Method," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 5, pp. 535-539, May 1997.