# OCR in a Hierarchical Feature Space

Jaehwa Park, Venu Govindaraju and Sargur N. Srihari

Center for Document Analysis and Recognition

Department of Computer Science and Engineering

State University of New York at Buffalo

Amherst, NY 14260

Email: jaehwap@cedar.buffalo.edu

Phone: (716) 645 6164 x 144

Fax: (716) 645 6176

Will be published on IEEE PAMI 2000

### **OCR** in a Hierarchical Feature Space

#### 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% is achieved by the *Hierarchical OCR*. This is at par with the other two traditional methods.

**Keywords:** pattern recognition, character/digit recognition, multi-resolution, 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] - [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 multi-resolution while circumventing the disadvantages cited above. The recognizer begins with features extracted in a coarse resolution and focuses on smaller sub-images of the character on each recursive pass thus effectively working with a finer resolution of a sub-image each time till the classification meets acceptance criteria. In other words, successive passes render the same features at a finer resolution.

The sub-images are selected by an approach called *gaze planning*. It guides the successive passes through a series of sub-images 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 sub-images 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 sub-images 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

$$P(c_i|\mathbf{x}) = \frac{p(\mathbf{x}|c_i)P(c_i)}{P(\mathbf{x})} = \frac{p(\mathbf{x}|c_i)P(c_i)}{\sum_i p(\mathbf{x}|c_i)P(c_i)},$$
(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% 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%). 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" sub-images of the original image. Sub-images 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 sub-images respectively (Figure 1). If  $N_D$  denotes the maximum depth of the tree generated by sub-divisions and the maximum number of branches per sub-division 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 sub-image of the  $l^{th}$  node of a tree at depth d, where  $d \in \{0 \dots N_D-1\}$  and  $l \in \{0 \dots N_B^d-1\}$ . Then the total number of nodes in the tree is

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

Features are computed for each node that is visited. Let v(d,l) represent the feature vector corresponding to sub-image 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 sub-class 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 Equation 1, where  $\mathbf{x}$  is the feature vector computed from the sub-image, 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 sub-nodes belonging to layer d,

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

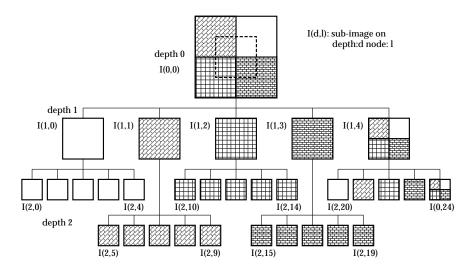


Figure 1: Sub-image division using Quin tree. There are nodes at depth 0,1, and 2 resulting in S(2,5) = 31 sub-images. In this case,  $Q_i^2$  is the product of 31 probabilities.

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 sub-classes corresponding to all sub-nodes processed up to that time, *i.e.*,

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

If the classes and sub-classes are statistically independent then  $Q_i$  represents the class *a posteriori* probability. This is clearly not the case (since the sub-images 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.

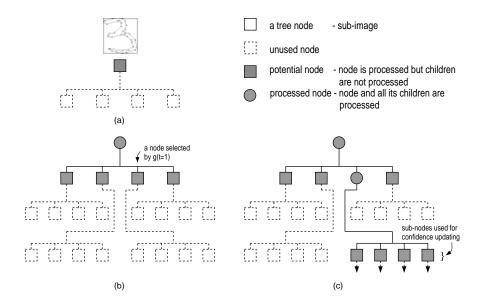


Figure 2: Confidence update recursion using a quad-tree; (a) starting from root, (b) after one recursive pass, and (c) after two recursive passes.

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 sub-nodes (Figure 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 sub-nodes are not used.

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

where  $Q_i'(t)$  is the confidence generated from the sub-nodes of a node selected at time t, where  $t \in \{0,1,\dots,\sum_{d=1}^{N_D-1}(N_B)^{d-1}-1\}$  This process is illustrated for a quad tree in Figure 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 sub-nodes 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

$$Q'_{i}(t) = \prod_{(d_{k}, l_{k}) \in sub(d, l)|_{(d, l) = g(t)}} P_{i}[v(d_{k}, l_{k})].$$
(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.

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

 $R_i(t)$  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,

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

where

$$R'_{i}(t)R' = \prod_{(d_{k}, l_{k}) \in sub(d, l)|_{(d, l) = g(t)}} max(P(c_{i}|v(d_{k}, l_{k})))|_{\forall v(d_{k}, l_{k})}$$
(10)

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

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

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

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

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

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

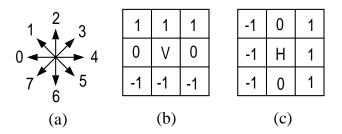


Figure 3: Contour representation; (a) contour slope division, (b) vertical weight mask, and (c) horizontal weight mask.

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 sub-nodes. If class  $c_{top}$  is the top choice and class  $c_{second}$  is the second choice in intermediate confidence result of time t, local separability  $\sigma^{(d,l)}(t)$  of a potential node (d,l) is given by

$$\sigma^{(d,l)}(t) = \frac{P_{top}[v(d,l)](t)}{P_{second}[v(d,l)](t)}$$
(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] (Figure 3a). A window of weights is convolved with the contour component and its neighbors to obtain the curvature at a contour point (Figure 3b, 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 Figure 3b, 3c and binarized using a threshold (0.65 in our implementation). Contour

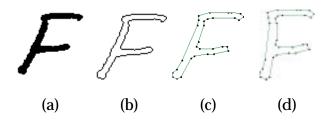


Figure 4: Contour representation and preprocessing; (a) input image, (b) pixel based contour representation, (c) piece-wise linearized contour, and (d) piece-wise linear contour representation after slant correction

•

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. Figures 4b and 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. Figure 4d shows the translated points after slant correction.

# 3.2 Sub-images

A variable size rectangular grid is used to define sub-images for the quin (quad) tree. The bounding box of a character image is divided into four rectangular regions (Figure 5c). the center of mass of the contour is first computed. A vertical and horizontal line through

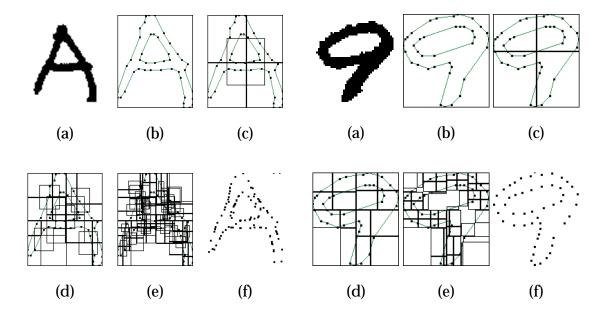


Figure 5: Character image and sub images of quin tree (class A) and quad tree (class 9): (a) o; inal image, (b) piecewise linear contour, (c) layer 1, (d) layer 2, (e) layer 3, and (f) reconstruction of contour image from critical points of final layer.

the center of mass determines the four regions. The quin tree structure is similar with an additional fifth sub-region which is located in the central area formed by joining the centers of the other four sub-regions. Subsequent layers are successively constructed by the same method.

We have implemented quad and quin trees with a maximum depth of 4. Figure 5c, 5d and 5e show sub-regions for each layer. Figure 5f show reconstructed contour image using critical points of the finest sub-regions at the deepest  $(4^{th})$  level.

#### 3.3 Features

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

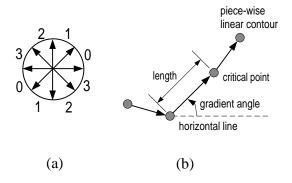


Figure 6: Gradient feature generation; (a) gradient division and (b) gradient angle measure.

Contour slope angles are quantized in pairs of  $\pi/4$  slots that are opposite each other labeled 0 to 3 in Figure 6a. The angle between the 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. (Figure 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 sub-image gives 4 gradient features.

4 (5 for quintree) moment features are obtained from the distribution of the contour within the sub-images. The ratio of the length of the sum of contour segments which are present within a sub-image 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  $5^{th}$  sub-images (0.5) for feature binarization.

# 4 Experiments

Experiments were performed 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

Data	No. of	Training	Testing	Image	Training set		Testing set	
set	classes	images	images	source(s)	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 1: Experiments are conducted with different size of training and testing data sets

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

Table 2: Performance comparison between the quad and quin trees

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 sub-image are obtained. Probabilities, P(c|v(d,l)) in every sub-image 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 sub-images are 85 (1+4+16+64) yielding a maximum of 680 dimensional binary feature vector when 8 features are used. In the

Classification Method	Proposed	Neural	k-NN
	method	Network	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

Table 3: Performance in various classification methods using data set 3. Proposed method operated on dynamic mode with  $log(\sigma) = 40$ .

case of a quin tree method, the sub-images are 156 (1+5+25+125) yielding a maximum of 1404 dimensional binary feature vector when 9 features are used. Quin tree sub-division 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 3 classification methods: (i) Hierarchical OCR as described in this paper, (ii) A neural network method, and (iii) A nearest neighbor method. The neural network is a fully connected. It has three 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 6 nearest neighbors. Storage requirements for Hierarchical OCR method refers to the size of the lookup table. In case of the neural network it refers 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

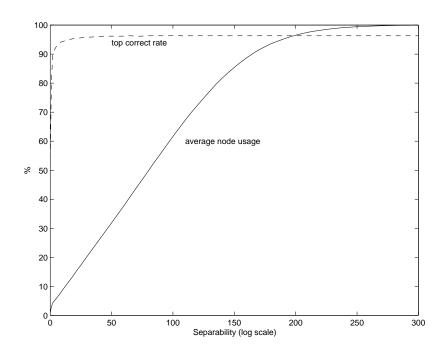


Figure 7: Dynamic recognition performance

the node next in sequence to be expanded is the one that allows the highest separability between the top two choices. Figure 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 1 node. Figure 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 pre-determined (hence static) parameters such as: (i) dimension of feature vector, (ii) the distance or confidence metric considered as threshold for recognition acceptance, (iii) the set of classes contending to match with a test pattern  $(e.g., \{0 \dots 9\})$  and (iv) equal computational resources available to every test pattern.

In very general terms, a large feature vector provides higher accuracy at the cost of

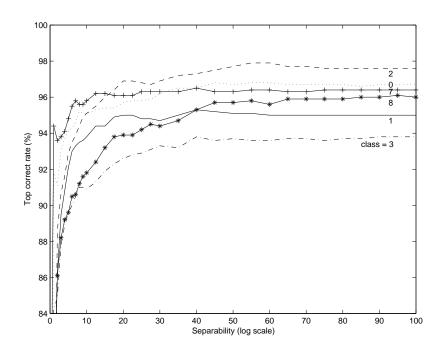


Figure 8: Performance by class

computational resources. Multi-resolution 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 multi-resolution recognizer such as the GSC described in [5] is known to be high because of the burden of additional features needed to accommodate the multi-resolution aspect of the feature space.

We have defined the confidence of a class based on the assumption that features in sub-images 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 "non-true" classes. In section 2.1 and

equation 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 non-true 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. Figure 9 illustrates that a if miss-classification 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 sub-images used by new recursive increases. On the other hand, when the topic choice is the true class, the separability parameter hold 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.

Multi-resolution classification is simulated by sampling smaller sub-images 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.

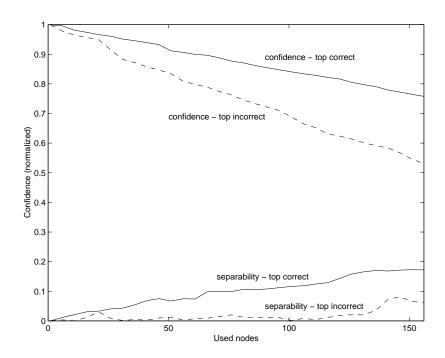


Figure 9: Confidence and separability changes - in data set 1 with quin tree. Solid line shows successful cases and dashed line shows error cases

.

# 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. *Proceedings of the IEEE*, 80:1029–1058, 1992.
- [2] G. Nagy. At the Frontiers of OCR. *Proceedings of the IEEE*, 80:1093–1100, 1992.
- [3] C. Y. Suen. Computer Recognition of Unconstrained Handwritten Numerals. *Proceedings of the IEEE*, 80:1162–1180, 1992.
- [4] J. Mantas. An Overview of Character Recognition Methodologies. *Pattern Recognition*, 19(6):425–430, 1986.
- [5] J. T. Favata, G. Srikantan and S. N. Srihari. Handprinted Character/Digit Recognition using a Multiple Feature/Resolution Philisophy. In *Proceedings of the Fourth International Workshop on Frontiers in Handwriting Recognition, Taiwan*, pages 57–66, 1994.
- [6] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, Inc, New York, 1979.
- [7] R. R. Bailey and M. Srinath. Orthogonal Moment Features for Use With Parametric and Non-Parametric Classifiers. *IEEE Transactions on Pattern Analysis and Machine Interlligence*, 18(4):369–398, 1996.
- [8] G. Srikantan S. W. Lam and S. N. Srihari. Gradient-Based Contour Encoding for Character Recognition. *Pattern Recognition*, 29(7):1147–1160, 1996.

- [9] H. Freeman. Computer Processing of Line-Drawing Images. *Computing Surveys*, 6(1):57–97, 1974.
- [10] C.-C. Lu and J. G. Dunham. Highly Efficient Coding Schemes for Contour Lines Based on Chain Code Representations. *IEEE Transactions on Communications*, 39(10):1511–1514, 1991.
- [11] M. D. Garris and R. A. Wilkinson. *NIST Special Database 3; Binary Images for Hand-written Segmented Characters (HWSC)*. John Wiley & Sons, Inc, New York, 1992.
- [12] J. J. Hull. A Database for Handwritten Text Recognition Research. *IEEE Transactions* on Pattern Analysis and Machine Interlligence, 16(5):550–554, 1994.
- [13] O. D. Trier, A. K. Jain and R. Taxt. Feature Extraction Methods for Character Recognition A survey. *Pattern Recognition*, 29(4):641–662, 1996.
- [14] T. M. Ha and H. Bunke. Off-Line, Handwritten Numeral Recognition by Perturbation Method. *IEEE Transactions on Pattern Analysis and Machine Interlligence*, 19(5):535–539, 1997.