

Handprinted Character/Digit Recognition using a Multiple Feature/Resolution Philosophy

J.T. Favata, G Srikantan, and S. N. Srihari

CEDAR

State University of New York at Buffalo, USA

Abstract

This paper outlines the philosophy, design and implementation of the Gradient, Structural, Concavity (GSC) recognition algorithm, which has been used successfully in several document reading applications at CEDAR. The GSC algorithm takes a quasi multi-resolution approach to feature generation. This philosophy coupled with the appropriate classification function results in a recognizer which has both high accuracy and good confidence behavior. This allows it to be used in higher level digit string and word recognition algorithms which search for digit/character boundaries. Tests of the GSC classifier on standard digit, character and non-character databases are reported.

1. Introduction

Many different approaches have been used by researchers to solve the problem of machine digit and character recognition [Suen92]. These approaches have included investigations of: feature sets [Srik93] [Man86], classifier algorithms, multiple combinations of classifiers [Ho93] and novel statistical methods [Klein93]. There can be much overlap between different methods and a precise taxonomy can prove difficult. It would be safe to say that the precise classification of an algorithm has much to do with the perspective of the investigator during the design of the algorithm.

Many different algorithms have been explored by the researchers at CEDAR [Lee93]. These algorithms have encompassed a wide range of feature and classifier types. Every algorithm has characteristics, such as high speed, high accuracy, good thresholding ability, and generalization, which are useful for specific applications. Examples of classifiers developed at CEDAR are listed in Table 1. This paper will outline the philosophical and practical details of one these classifiers: the Gradient, Structural, Concavity (GSC) classifier.

2. Philosophy

The approach used in designing the GSC was based on the observation that feature sets can be designed to extract certain types of information from the image. Feature detectors can be built to detect the local, intermediate and global features of an image. The basic unit of an image is the pixel and we are interested in both its location (x,y coordinate) and the relationship of the pixel to its neighbors at different ranges from locally and globally. This

can be expressed by saying we want to determine the relationship of each pixel to every other pixel at increasing distances. In a sense, we are taking a multi-resolution approach to feature generation. The GSC features approximate a multi-resolution approach by being generated at three ranges: local, intermediate and global.

The gradient features detect local features of the image and provide a great deal of information about stroke shape at a short distance. The structural features extend the gradient features to longer distances and give certain useful information about stroke trajectories. The concavity features are used to detect certain stroke relationships at long distances which can span across the image. In practice, there are computationally imposed limits to how a particular philosophy can be implemented. In the GSC algorithm, certain decisions were made in the exact detection and representation of the features to result in a practical algorithm. The exact implementation should not distract from the underlying philosophy. It should be emphasized that we are presenting one particular implementation of our philosophy and that others are possible. The total feature vector length is 512 bits. It is important to note that the feature vector is binary. The GSC feature vector is very compact, other algorithms may use a smaller number of multi-valued (or real) features but the effective number of bits to represent such feature vectors can actually be quite large.

3. Feature Description

The GSC algorithm was designed to work with binarized images, so it is presumed that the image has been thresholded using a suitable algorithm [Otsu79]. The image is slant normalized using a moment based algorithm to reduce the effects of skew. A bounding box is placed around the image and the features are computed (see below). The feature maps are sampled by placing a 4x4 grid on the maps (see Figure 1). The features themselves are computed independently of this sampling grid.

Gradient Features

The gradient features are computed by convolving two Sobel operators on the binary image. These operators approximate the x and y derivatives in the image. The vector addition of the operators' output is used to compute the gradient of the image. Since the gradient is a vector with magnitude and direction, only the direction is used in the computation of the feature vector. The direction of the gradient can range from 0 to 359 degrees. This range is split into 12 non-overlapping regions of 360/12 degrees. In each sampling region (4x4 grid), a histogram is taken of each gradient direction at each pixel which lies in the region. A threshold is applied to the histogram and the feature bit is set for each feature count that exceeds a threshold. This subset of the GSC features produces $12 \times 4 \times 4 = 192$ bits of the total feature vector.

grid size of 5x5. Other stroke features detected are four types of corners which consist of perpendicular co-occurrences of strokes. These features contribute $4 \times 4 \times 12 = 192$ bits to the total feature vector.

Concavity Features

These features, which are the coarsest of the GSC set, can be broken down into three subclasses of features. The total contribution of these features are $4 \times 4 \times 8 = 128$ bits.

Subclass A. Coarse Pixel Density Features

These features capture the general groupings of pixels in the image. They are computed by placing the 4x4 sampling grid on the image and counting the number of image pixels that fall into each grid. Thresholding converts these area counts into a single bit for each region. This feature contributes $4 \times 4 = 16$ bits of the feature vector.

Subclass B. Large Stroke Features

These features attempt to capture large horizontal and vertical strokes in the image. Run lengths of horizontal and vertical black pixels across the image are first computed. From this information, the presence of strokes are determined by testing for stroke lengths above a threshold. This feature contributes $4 \times 4 \times 2 = 32$ bits of the feature vector.

Subclass C. U/D/L/R/H Concavity Features

These features are computed by convolving the image with a star-like operator. This operator shoots rays in eight directions and determines what each ray hits. A ray can hit an image pixel or the edge of the image. A table is built for the termination status of the rays emitted from each white pixel of the image. A computationally efficient algorithm similar to run-length encoding is actually used to compute the star operator. The class of the each pixel is determined by applying rules to the termination status patterns of the pixel. Currently upward/downward, left/right pointing concavities are detected along with holes. The rules are relaxed to allow nearly enclosing holes (broken holes) to be detected as holes. This gives a bit more robustness to noisy images. These features can overlap, in that, in certain cases more than one feature can be detected at a pixel location. These features contribute $4 \times 4 \times 5 = 80$ bits.

4. Classification

The classification problem can be stated as finding functions which map feature vectors to classes. Ideally these functions should map all valid regions of the feature space to the class space. With high dimensional feature spaces, this can be a very difficult problem. There may never be enough training data to adequately estimate these functions in certain regions of

Structural Features

The structural features capture certain patterns embedded in the gradient map. These patterns are "microstrokes" of the image. Several 3×3 operators are passed over the

Table 1. CEDAR Classifiers

Name	Performance (digits)	Speed (digits/sec)
Image	96.02%	66.7
Binpoly	96.43%	25.0
Chaincode	98.33%	67.0
Gabor*	97.70%	6.7
Gradient	98.46%	5.3
Histogram*	97.47%	62.5
Morphology*	97.92%	0.5
GSC	98.87%	10.0

Notes: 1. (*) indicates classifiers which are no longer actively being developed
2. Performance and speed figures are approximate and used for relative comparison

gradient map to locate small strokes pointing up/down and diagonally. These strokes are combined into a larger features using a rule table. The largest span of the feature covers a

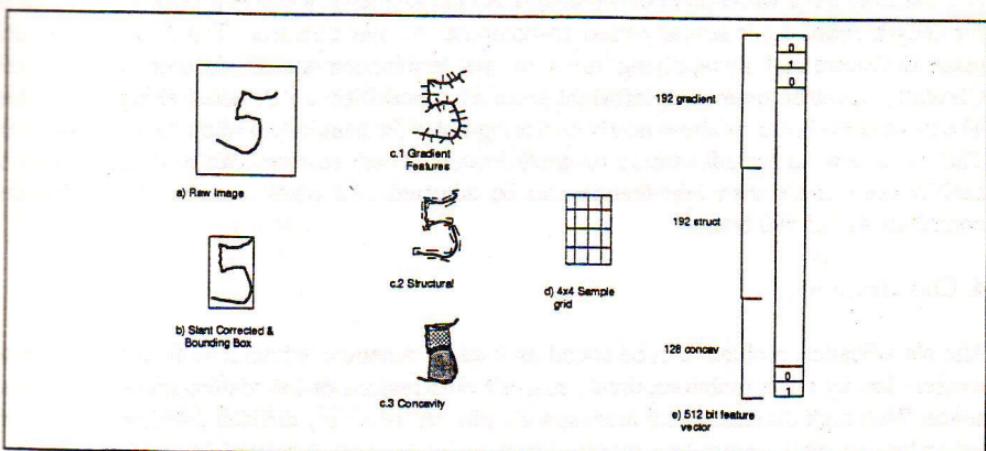


Figure 1. GSC features

the feature space. In addition, we can also assume that our features may not have enough resolving power to distinguish among certain cases of images or there may be an aliasing problem where two different image patterns may map into the (nearly) same feature vector. This can sometimes be due to practical implementation compromises involved in the algorithm design such as the need for speed or machine memory size limitations. Since we are dealing with handwritten images, it is also possible that certain cases of digits or characters may be so borderline or poorly written that even humans could misrecognize them.

In computing the classification functions, we would like to generate functions that accurately reflect the training set and generalize well to the testing sets. This means that the functions should accurately recognize members of the training set and smoothly rolloff as the feature vectors move away from the labeled vectors of the training set. That is, as the underlying image is smoothly transformed into another image of a different class, we would like a smooth transition of the classification function. This property is useful to prevent spurious responses in those regions of the feature space which are inadequately represented in the training set. In addition, we would like good behavior in invalid regions of the space. This last property is important if we use the recognizer to distinguish between valid and invalid images such as using the system to find valid digits from a segmented. This work has focused on several different classifiers each with various tradeoffs:

K-NN

The k-nearest neighbor (k-nn) approach attempts to compute a classification function by using the labeled training points as nodes or anchor points in the n ($n=512$) dimensional space. In a sense, this is the most detailed description of the space that is possible from the training samples. Rather than using a 1-nearest neighbor classifier, we chose a k-nn classifier to reduce the effect of mislabeled training data and to get a better estimate of the prototype density at any particular point in the space. By choosing an appropriate distance metric a smooth rolloff in response can be obtained as a feature vector is moved away from a cluster center. Since the feature vector is binary, the comparison between unknown and labeled vectors involves bit operations which can be done in parallel at the machine's word length. The main disadvantages of k-nn is the memory required to hold the training data and the speed in classification. A clustering technique has been developed to greatly reduce the number of comparisons necessary to implement this classifier and it is now comparable in speed to other classifiers.

Neural Nets

Classification functions based on feed-forward neural network architecture have been explored at CEDAR with some success. It is our experience to date that these functions do not perform well with the GSC feature set, especially with characters. The problems that occur are typically poor accuracy and ill-behaved response to invalid (non-character/digit)

test images. We are currently investigating the reasons for this unsatisfactory behavior. Some of the problems may have to do with the limited training data available for certain character classes. This is especially true for recognition of cursive characters in which there are no standard training and testing databases available yet. We are experimenting with fuzzy training techniques. This approach uses the confidences output from another recognizer algorithm as the target values of the neural net during training. Early tests have shown performance improvements in the neural net.

Polynomial Functions

This set of classification functions are computed as low order polynomial functions of the feature coefficients (bits) using a minimum least-squares error criterion. In general, these classifiers perform poorly compared to k-nn, however, they can be reasonably well behaved with respect to invalid images. Both a linear and quadratic function classifier have been used with success in a word recognition algorithm [Fav93].

5. Refinements

A number of refinements of both feature generation and classification have been tried with various results. The most successful improvement to the feature generation was to use a variable 4x4 sampling grid on the image. A horizontal and vertical histogram of the image is computed and sampling lines are placed on the equi-mass divisions of the histogram. This results in higher sampling of regions with the most mass. A significant improvement in performance with digit recognition has been obtained with this scheme. A number of different matching metrics [Tubb89] have been tried with the k-nn classifier. Each metric, or more appropriately, matching function, blends the number of features that co-occurred with the number of features that did not match or were not present.

6. Applications

The GSC classifier has been used successfully in several different applications. The first application is the segmentation of ZIP codes from digitized postal address images. Typically, a handwritten ZIP code contains 5 or 9 isolated digits which usually can be read by a digit recognizer. In some cases, however, two or more of the adjacent digits can be touching in unpredictable ways. One approach for accurate recognition is to force a segmentation of the digit string and recognize each (hopefully) isolated digit individually. The GSC classifier is a key element of this approach. Another successful application of the GSC algorithm is in word recognition. A handwritten word is a sequence of characters whose identities must be determined. This can be accomplished by segmenting the word and determining character boundaries and identities. For printed words, segmentation is usually easy, but for cursively written words it is much more difficult.

Experiments with the GSC classifier have shown that it excels in these applications because the confidences are reliable enough to use in detecting the best character and digit candidates after segmentation. Figure 2 shows the polynomial and GSC algorithms tested on a series of images containing noise, special marks and digit sequences. It can be seen that the GSC algorithm offers reliable confidences and that nearly every one of these images would be rejected by picking one threshold value.

Iterative Digit String Segmentation Algorithm

Given the image of a string of digits, the goal of the segmentation algorithm is to partition the image into regions, each containing an isolated digit. A recognition aided iterative method is used. Adjacent digits can be touching and some of the digits might be broken into more than one component (e.g. 5-hats). Therefore the number of digits in a digit string is not simply a count of the number of components in the field. Components which are classified as touching digits must be segmented appropriately. The module which performs the segmentation and subsequent recognition of the segmented digits does the following: Given a connected component with 2 to 4 digits, the module estimates the number of digits in the component, performs the appropriate segmentations and recognizes the individual digits. The module currently performs at a 71% correct rate.

The segmenter can be invoked in two different modes:

- (i) estimate the number of digits and
- (ii) force a given digit string length.

The number of digits is initially estimated from the aspect ratio of the digit string and successive estimates are obtained by a linear regression model. Digits that are recognized with high confidence are removed after each iteration. The effective contribution of the removed digits to the density of the digit string is recorded. This information is fed to a least squares linear model [Fen91]. By setting the density to zero (all digits are removed), the least squares equation can estimate the number of digits in the string. Connected digit components are split into required number of digits. The segmenter has a correct segmentation rate of 92.9% when the string length is specified, and is 87.5% correct when the number of digits has to be estimated.

General Handwritten Word Recognition

The GSC has been applied to general handwritten word recognition in which a word can contain any mixture of discrete, cursive or touching discrete characters. A recognition technique called the *hypothesis generation and reduction* algorithm (HGR) [Fav93] first segments the word image using a number of candidate segmentation points. The GSC classifier is then used to pinpoint the most likely character boundaries by extracting the

strokes from among pairs of segmentation points (see Figure 3). Later, a lexicon driven post-processor finds the most likely word using the GSC segment pair confidences and other available context.

	Polynomial	GSC		Polynomial	GSC
-	1 0.75	1 0.17	789	4 0.43	8 0.45
↔	1 0.85	0 0.68	552	9 0.45	1 0.31
↔	1 0.79	1 0.70	95	4 0.45	8 0.43
-	1 0.77	1 0.24	556	7 0.76	0 0.39
↓	1 0.88	0 0.43	♥	7 0.64	1 0.69
7	7 0.71	7 1.00	±	4 0.69	3 0.26
↗	7 0.64	7 1.00	Ø	4 0.73	0 0.76
↖	4 0.52	7 0.95	46%	4 0.50	4 0.24
↗	7 0.68	7 0.80	&	9 0.65	8 0.33
∂	9 0.77	0 0.34	78	7 0.45	9 0.28
Π	7 0.51	0 0.43	789	7 0.71	8 0.33
Ψ	1 0.55	4 0.65			

Figure 2 - GSC thresholding test - comparison of confidences returned by polynomial and GSC classifier on non-characters. Output is arranged as (input image, poly decision, poly conf, GSC decision, GSC conf).

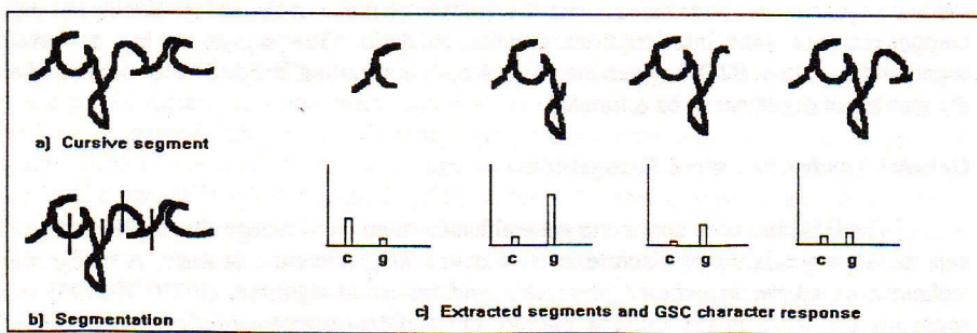


Figure 3 - Finding character boundaries

7. Experimental Results

The GSC classifier has been well characterized by extensive testing. Figure 4 compares several variations of this classifier on handwritten digits. The training set consisted of 24,000 digits taken from various databases, the test set consisted of 2700 digits. Figure 5 shows the GSC classifier performance trained separately on 60,000 upper and lower characters chosen from the NIST handprinted database. The test set consisted of 20,000 upper and lower case characters.

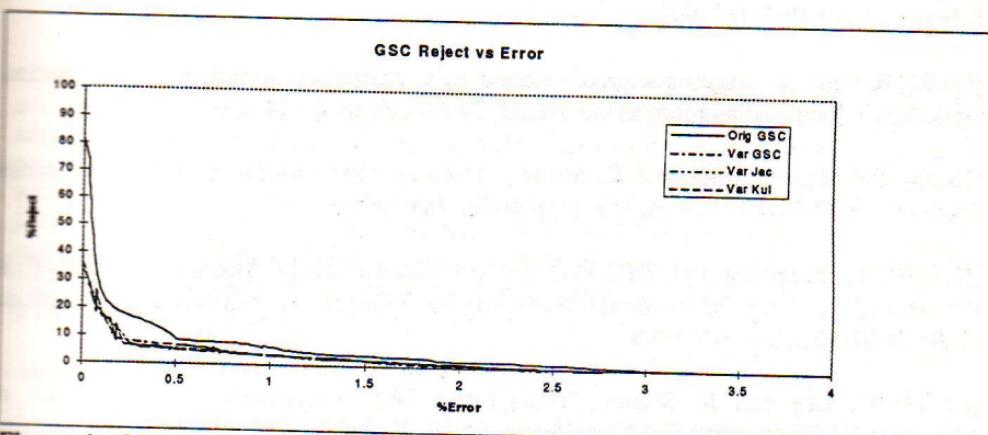


Figure 4 - GSC digit results for fixed and variable GSC features, with different metrics

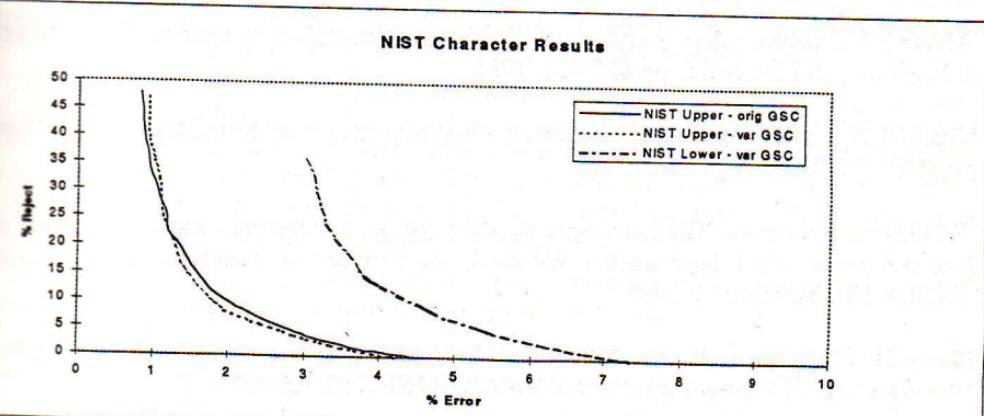


Figure 5 - NIST upper/lower character results

Acknowledgment

The authors would like to thank: Dr. Venu Govindaraju, Evie Kleinberg, Ajay Shekhawat, Phil Kilinskas and D. S. Lee for helping in various ways during the development of the GSC classifier.

8. References

- [Fav93] J. Favata, "Recognition of Cursive, Discrete, and Mixed Handwritten Words Using Character, Lexical and Spatial Constraints", Tech Report 93-32, Dept of Computer Science, SUNY Buffalo, 1993
- [Fen91] R. Fenrich, "Segmentation of automatically located handwritten words", Proc. Int. workshop in handwriting recognition, Bonas, FRANCE, pp. 33-44, 1991.
- [Ho93] T.K. Ho, J. Hull, and S. Srihari, "Decision Combination in Multiple Classifier Systems", *IEEE PAMI*, Vol 16, No. 1, pp 66-75, Jan 1994
- [Klein93] E. Kleinberg and T.K. Ho, "Pattern Recognition by Stochastic Modeling" in Proceedings of Third International Workshop on Frontiers in Handwriting Recognition (IWFHR III), Buffalo, NY 1993
- [Lee93] D. Lee and S. Srihari, "Handprinted Digit Recognition: A comparison of Algorithms" in Proceedings of Third International Workshop on Frontiers in Handwriting Recognition (IWFHR III), Buffalo, NY 1993
- [Man86] J. Mantas, "An Overview of Character Recognition Methodologies", *Pattern Recognition*, Vol 19, No. 6, pp 425-430, 1986
- [Otsu79] N. Otsu, "A threshold selection method from grey-level histograms", *IEEE Trans on SMC*, Vol 9, pp. 62-66, Jan. 1979
- [Srik93] G. Srikantan, "Gradient representation for handwritten character recognition" in Proceedings of Third International Workshop on Frontiers in Handwriting Recognition (IWFHR III), Buffalo, NY 1993
- [Suen92] C.Y. Suen et al, "Computer Recognition of Unconstrained Handwritten Numerals", *IEEE Proceedings*, Vol 80, No 7, pp 1162-1180, July 1992
- [Tubb89] J.D. Tubbs, "A Note on Binary Template Matching", *Pattern Recognition*, Vol 22, No. 4, pp. 359-365, 1989

Favata, Srikantan and Srihari, Proc. IWFHR 1994, pp. 57-66.