A Shape Analysis Model with Applications to a Character Recognition System *

Jairo Rocha Theo Pavlidis

Computer Science Department
SUNY at Stony Brook
Stony Brook NY 11794

Abstract

A method for the recognition of multifont printed characters is proposed, giving emphasis to the identification of structural descriptions of character shapes using prototypes. Noise and shape variations are modeled as series of transformations from groups of features in the data to features in each prototype. Thus, the method manages systematically the relative distortion between a candidate shape and its prototype, accomplishing robustness to noise with less than two prototypes per class, on the average. Our method uses a flexible matching between components and a flexible grouping of the individual components to be matched. A number of shape transformations are defined. Also, a measure of the amount of distortion that these transformations cause is given. The problem of classification of character shapes is defined as a problem of optimization among the possible transformations that map an input shape into prototypical shapes. Some tests with hand printed numerals confirmed the method's high robustness level.

1. Introduction

This paper presents a method for character recognition (OCR) which emphasizes explicit design as opposed to learning. Prototypes for each class are designed explicitly. Matching between data and prototypes relies on definitions of the distortions and noise encountered in OCR environments. Basically the method is a structural one [1] since character classes are defined in terms of graphs. However, it avoids two common pitfalls of structural methods: their sensitivity to noise [2] and the need to construct a large number of prototypes to account for many shape variations [3] . This is achieved by modeling distortion using a set of transformations from group of features in the data, to features in the prototypes.

Another departure from past OCR work is that the major part of the classifier is designed without resorting to "learning". Learning forms the cornerstone of statistical pattern recognition and it is often considered a must by researchers in the field. However, it is usually impractical to train a classifier on a complete set. This point was recently brought up by L. A. Pintsov [4] who found that the required "learning" set is enormous and well beyond the size of most current experimental sets.

The need for a large learning set can be seen by considering the example of training a classifier to discriminate between an "O" and a "C". In the case of an "O", a break in any part except the middle of the right side should not affect recognition. In the latter case, it could cause confusion with a "C". To learn this characteristic, the recognizer should be presented with "O"s containing all possible combinations of acceptable breaks. Including such distortions should increase the size of a training set by a factor of 10 if not more. Already the Bell Labs Neural Net is trained by producing distortions artificially according to a model rather than by using true samples (personal communication by Henry S. Baird). In contrast, our system uses prototypes designed by hand and models of noise and distortions. Therefore, it can transform combinations of distortions without previous training.

The following is an outline of the classifier: first, a page is scanned and stored electronically and each character is converted into a set of features which also form a graph. Such representations have been used by numerous authors [1] . The specific method producing the graphs used in this paper is described in [5] . Second, the main classifier finds, for each candidate graph C, a prototype graph P that matches C more closely than any other prototype graph. Finally, a second stage classifier, described also in [5] , is used to distinguish shapes that are similar under the flexible matching (such as "4" with "9"). In this final stage, detailed differences among particular characters are used, as in [6] .

One of the novelties of the method is to allow multiple to one matchings from features of C to features of P. This approach departs from the traditional computational versions of inexact matching where every effort is made to find one-to-one mappings. Xie and Suk [7] , for example, use relaxation to calculate matching probabilities among features. Lu et al. [8] can represent relations

* Research supported by U.S. Postal Service Contract No. 10423091C3770.

0-8186-2840-5/92 $03.00 © 1992 IEEE

among features and among subgraphs of features. However, no effort is made in any of these works, to group features that have been divided by spurious points. This is equivalent to the notion that all features carry "information", except those that seem "unimportant" at the low level, and are eliminated using thresholds. Yamada [9] allows a limited multiple-to-one matching. He proposes a dynamic programming matching method for contours of characters. The algorithm finds the best global correspondence of segments. In addition, not only one-to-one correspondence between mask segment and input but also one to two and two to one correspondences are permitted. The method is very effective; nonetheless it is computationally expensive because the number of segments in a contour is large; it also suffers from the usual problems of character merging and splitting that change the contours drastically.

Graph structures are treated as polygonal approximations of symbols represented on the plane. Their features are interpreted (and modified) separately by different prototypes. They can be grouped in a very dynamic way, which contrasts with the static representations of most previous works. This means that in order to recognize a character, the graph that represents its shape may have to be virtually transformed in the context of each prototype. Sometimes topological changes are needed on the graph in order to be identified as a prototype, like closing of open loops and insertion of features. Therefore, we study also subgraphs that cannot be continuously transformed into a prototype. Algorithms to test isomorphism of planar graphs, (e.g., [10]) are not useful for our problem.

One of the most puzzling problems in shape analysis is that of defining shape invariants. It is well known that geometrical and topological invariants are not good stable features; however, they are important for the shape recognition. We believe that topological and non-topological transformations should be used in shape analysis but always measured in the context of the shapes that are to be recognized. Therefore, not only intrinsic shape invariants but also *a priori* knowledge of the prototypical models and a measure of the importance of the transformations are used.

The rest of the paper is organized as follows: Section 2 gives a brief explanation of the feature extraction method; Section 3 introduces the definition of a multiple-to-one matching; Section 4 defines the transformations used; Section 5 explicitly states some assumptions; an algorithm to calculate multiple-to-one matchings is presented in Section 6; a distance between graphs is defined in Section 7; experimental results are in Section 8, and conclusions and further research are given in Section 9.

2. Feature description

Features are extracted from single characters directly from gray scale images using a method described in [5] . They include convex arcs and strokes, singular points and their relationships. A singularity is one of the following three places:

- a branch point: a point where more than two arcs or strokes are near each other;
- a concave vertex: a point where two convex arcs meet but the union of the arcs is not convex.
- a sharp corner: a point where the internal angle between two strokes is small;

The singular points are represented as links between the arcs strokes and convey the spatial relationship among the structural features.

The feature extractor reduces the dimension of the problem but does not make any implicit local decision about feature meaning, leaving that task totally to the classifier.

The final product of the feature extraction process is the structural description of a character shape that consists of the specification of its features and their spatial interrelations. A character shape is a graph $G(J,F)$, where edges F of the graph are features (strokes or arcs) of the character, and nodes J of the graph are junctions among strokes and arcs, and represent the singular points on the shape. The singular points are points with more than two adjacent features, ending points (points with only one adjacent feature), concave angles and sharp corners. Strokes have two attributes: angle and length (normalized to the maximum length in the graph); arcs have three attributes: the beginning and ending angles from the center in the counterclockwise direction, and the length. A joint associates a spatial relation type (up, down, left, right) to each adjacent feature.

3. Multiple-to-one matching

Structural descriptions are used to define prototype characters that store part of the knowledge-base of the recognition system. This system inputs candidate characters, calculates their structural description and attempts to identify the character with a prototype model.

The recognition method is based on the matching of a candidate character C with a prototype graph P. The method allows multiple-to-one matchings from features of C to features of P. Furthermore, each set of features in C that is mapped to a unique feature in P must define a path in C that is transformed into a feature in P.

Previous matching algorithms (i.e., [11] , [12] , [13] and [8]) assume that an input shape is identified better if features and joints of the graph are matched one to one to features and joints in some prototype graph. However, this assumption is not necessarily true.

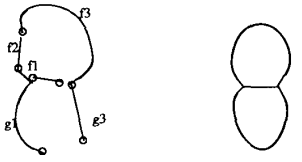

Figure 1. Two paths: f_1, f_2, f_3 and g_1, g_3, which is broken. They match to the arcs of the "8".

For example, a joint can be "artificially" added by the presence of a spurious concave angle, like between features f_2 and f_3 in Figure 1. In another example, a feature labeled as an arc in the prototype may need to be matched to two adjacent arcs in the input graph even though there is a sharp angle in the joint that connects the arcs. These are examples of continuous transformations among homeomorphic graphs [14] . Consequently, we study subgraphs of the input graph that are homeomorphic to a prototype. The subgraph homeomorphism problem is NP-complete even if the graphs involved are connected and planar [15] . When the prototype graph is fixed, there is a polynomial time solution to the problem of fixed subgraph homeomorphism for planar graphs [16] . However, this result is of little practical significance for us because the degree of the polynomial is too large to be useful, and because we also study other subgraphs that are not homeomorphic to a prototype.

We now define a multiple-to-one matching, which matches paths to features. Let $C=G(J_C, F_C)$ and $P=G(J_P, F_P)$ be candidate and prototype graphs, respectively. A *matching* is a pair (H, I) of partial functions, $H: F_C \to F_P$ and $I: J_C \to J_P$, such that they satisfy the following conditions:

1. for all $f_i \in F_C$, $i=1,...,n$, $g \in F_P$, and $g=(c,d)$,

   if $H(f_i)=g, i=1,...,n \Rightarrow$

   $f_i, i=1,...,n$, *form a path that starts in a* and *ends in b*

   and $I(a)=c$ and $I(b)=d$ *or* $I(a)=d$ and $I(b)=c$.

2. If two different paths of features of C are matched by H, then they do not cross, i.e., they do not share internal joints.

Intuitively, a matching of two graphs is a mapping of features and joints, in which the joints are matched consistently with the feature matching.

4. Graph transformations

Under the assumption that there is no observation noise and no alteration of the features or their relationships, exact matching of structural descriptions is enough to identify candidates with the prototypes. However, any real structural description of a character is randomly altered and therefore, inexact identification is necessary for real world systems.

Inexact matching can be formalized as a minimization of the possible transformations from perfect matching. As a result, the features are matched as well as possible and their relationships are preserved as possible through the matching. In this section, we define the transformations that are allowed, the amount of deformation of each transformations and the context in which they are applied.

A transformation eliminates a deformation in the input graph. A deformation is identified when there is a singularity (concave angle, sharp angle, gap, etc.) in the candidate, that it is not present in a particular prototype. Given an input graph, it is very likely that transformations allow it to match different prototypes; however, we expect that the candidate graph needs to be transformed less when matched to its correct prototype than to any other prototype. As a result, the amount of deformation imparted by the transformations should be accumulated in order to compare effects of different transformation sequences.

4.1. Transformations

The following are the transformations, with their respective measure:

Bypassing of a concave angle. Concave angles carry shape information (e.g., in "3"s and "8"s), so they are identified on the shapes. However, if they appear on arcs when they are not expected, they can be bypassed with an associated deformation cost of size of the indentation (distance d in Figure 2).

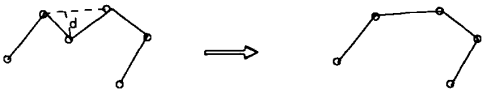

Figure 2. Concave angle of size d.

Straightening of strokes. If the candidate is expected to have a single stroke among two given joints, but instead it has several adjacent strokes, they are straighten. The associated deformation cost is the maximum distance from the original joints to the final stroke. See Figure 3.

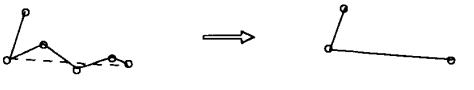

Figure 3. Straightening of a path of strokes.

Association of joints. When two joints are situated in different places but the prototype suggests that they should be associated to a single joint, they are associated

to only one joint that has as adjacent features the union of the original two nodes. The measure associated is the distance d shown in Figure 4.

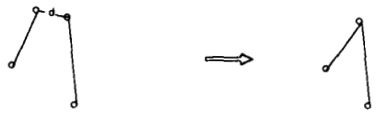

Figure 4. Association of two joints separated by a distance d.

Filling of gaps. A feature may be broken but its original joints may be identified. In this case, the parts connected to these joints are assumed to conform only one feature. The associated cost is the size of the gaps, as shown in Figure 5.

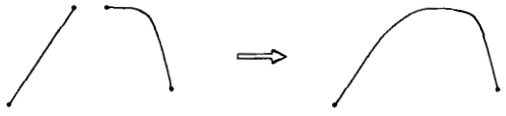

Figure 5. Filling the gap of a path.

Rewriting of strokes into arcs. Due to the shape representation technique used for skeletons, an arc is a list of segments, and vice versa. Therefore, if for any reason (sharp angle in a joint, joint of degree bigger than two) a list of strokes was not represented as an arc, then it is converted into one. An angle between two strokes is smoothed into an arc with an associated cost proportional to its cosine.

Figure 6. A path of strokes can be rewritten into an arc.

Insertion or deletion of features. If a feature appears in the candidate graph and it is not present in the prototype, it is considered noise or decoration, and it is deleted; the associated cost is equal to the length of the feature. In considering the prototype, features are required or optional. If a feature appears in the prototype but not in the candidate, the matching possibility is not considered any further unless the feature is not required. In this case, there is no cost associated for the missing optional feature.

Attribute transformations. A stroke is matched to another stroke if their direction is compatible (both directions are inside a cone of 60 degrees); the associated cost depends on the deviation of directions and lengths among them. An arc is matched to another arc with an associated cost that depends on the deviation of their lengths and their initial and final internal angles. An arc in the candidate can be matched into a stroke in the prototype, as follows: the arc is transformed into its secant (the stroke defined by the endings of the arc), and the associated cost is the matching cost of the secant with the prototype stroke plus the maximum distance from a point in the arc to its secant. An arc in the prototype is never matched to a single stroke in the candidate, since arcs and strokes have different meanings for character shapes.

4.2. Context of transformations

The context of a transformation is a set of joints and features in the candidate and prototype graphs that are used in the transformation. It consists of a feature in the prototype, with both adjacent joints, and a path of features in the candidate, including the joints.

In the context of a path and a feature in the prototype, the type of singularities in the path determines the kind of transformation that will be applied. Since we consider a transformation for each type of singularity, the transformations defined are sufficient to bypass any kind of combination of singularities. As a result, they can model any distortion than causes singularities.

5. Assumptions

Inexact matching is computationally expensive. However, we can add some geometrical constrains that will make the problem more tractable.

Figure 7. A prototype graph for "4", and a candidate graph.

We should not forget that the graphs we are matching represent bidimensional shapes *oriented* on the plane. Each feature of the graphs contains attributes that define relative orientation and location with respect to other features. For example, consider the prototype graph for the numeral 4 that has a joint with four adjacent features; the individual orientation of these features in real printed "4" varies considerably, but the order in which they appear around the joint does not change; in Figure 7, starting in feature f and counterclockwise, we find features g and h; this order is consistent with the order of the respective features in the prototype. We also notice that the deviation in the orientation of a stroke, with respect to the one in its prototype, cannot be excessive, since horizontal lines have different meanings that verti-

cal lines. We assume the following constrain:

Geometrical assumption. A matching should preserve the relative order of the orientation of the features around two matched joints, and the overall orientation of the features, since rotations near 90 degrees are very unlikely.

There is another assumption about paths in the candidate, that we make explicit here, and it is illustrated in Figure 8.

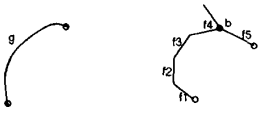

Figure 8. The path f_1, f_2, f_3, f_4 is extended with f_5.

Good-continuity assumption. Let $f_1, \ldots, f_n$ be a path in the candidate, that ends in a joint b, with degree greater than one, and let g be a feature of the prototype. If it is known that the path must be extended with a feature f_{n+1}, adjacent to b, to improve the matching with g, then it is possible to find optimally the next feature f_{n+1} adjacent to b that it is also being matched to g using only the attributes of the features adjacent to b and the attributes of g.

In other words, the orientation of a feature g is enough to decide locally which feature to choose to extend a path, when the last joint is a branch point.

6. Algorithm

The transformation and matching of a candidate graph into a prototype graph is combinatorial in nature, because the feature set needs to be partitioned into paths; an example is shown in Figure 1. If the paths that need to be transformed are known in advance, then the problem can be reduced to the polynomial one-to-one inexact matching. However, there are cases in which there is no simple way to decide which grouping of features into paths is better, unless they are considered explicitly.

In this section we will describe the basic routine that matches features adjacent to two joints, the use of different sequences of transformations to extend a matching, and the recognition of an optimal solution.

6.1. Matching of two joints

The reasons given for the geometrical assumption imply that given two joints to be matched (one in the prototype and in the candidate), the relative orientation of features around a joint is a very good guideline for the matching process. The algorithm in Table 1 calculates a best inexact matching between the features adjacent to two joints, given that two of those features are already matched.

Algorithm *Match_join*(a, b, f, g).
Input: Joints $a \in J_C$ and $b \in J_P$; Features $f \in F_C$ and $g \in F_P$;
Output: Matching of the features adjacent to a and b, such that f is matched to g, and cost of the matching.

Sort the features adjacent to a, counterclockwise after f.

Sort the features adjacent to b, counterclockwise after g.

Apply an *inexact string matching* algorithm among the previous two lists, using the function *cost*(f_1, g_1) to calculate the matching cost of two features, and the function *length*(f_1) to calculate the penalty for not matching a feature.

Table 1. Algorithm *Match_join*.

The function *cost* measures the difference between two features, according to their type (arc or stroke) and attributes. The function *length* calculates the relative length of a feature.

6.2. Alternatives for transformation

We just explained how two joints are matched. In this subsection, we consider the several options for selecting the next pair of joints to be matched. Assume we have a partial matching between C and P. The order in which the joints have been matched in the partial matching is directed by a depth-first search on the joints in the candidate graph. This means that enough information is saved with the partial matching to find the last feature and joint considered and a feature already matched but with one of its joints not matched yet.

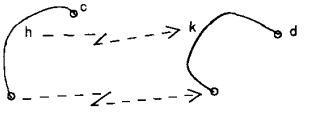

Figure 9. Extension rule: the next pair of joints to be matched is c and d.

To extend a partial matching of joints and features with a new pair of matched joints, the new pair is found by looking for a feature $h \in F_C$ already matched with k but with one of its adjacent joints, c, not visited yet, as shown in Figure 9. The corresponding non-visited joint in the prototype is uniquely determined, since is the joint d adjacent to k that is not matched to the other joint adjacent to h. The next step is to match the features adjacent to c and d, such that h is matched to k. This method to obtain the next pair of features and the next pair of joints to be matched will be called *the extension rule* . For example, in Figure 10, the extension rule finds that the next pair of joints to be matched is a_2 with b_2, adjacent to the features f_2 and g_2.

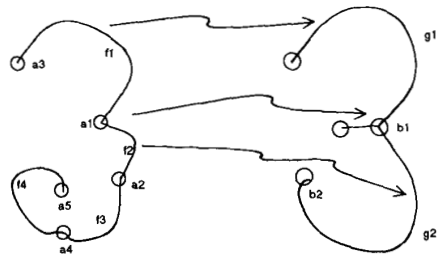

Figure 10. a_2, a_4 and a_5 are alternatives for matching b_2.

Then, the algorithm considers paths of different lengths, starting with the feature and joint suggested by the expansion rule. In Figure 10 again, the alternatives considered for matching feature g_2 in the prototype are the following paths: $\{f_2\}$, $\{f_2, f_3\}$ and $\{f_2, f_3, f_4\}$. The corresponding three possible matchings for the joint b_2 are a_2, a_4 and a_5. In this example, paths are extended with the only possible next feature, since there are no branch points among the intermediate joints. In general, the next feature to expand a path is found using a good-continuity criteria, when a branch point creates more than one alternative to continue the path. Notice that the extension is done in the context of a feature in the prototype, so there is knowledge of the direction that the path should follow.

The function *transform*(*path*, *feature*) finds the cost of transforming the list of features in the path by using the appropriate transformations. Therefore, each of the alternative path lengths has an associate cost. The maximum length of a path is bound by a predefined threshold that depends on the maximum size of a feature in the candidate.

6.3. Search for the optimal solution

The algorithm that we propose considers paths of different lengths in certain directions, starting on all possible joints. Although this strategy seems too expensive, the great variability of the graph characters makes difficult to introduce irrevocable decisions that permanently discard alternatives before they are proven fruitless. We now finish the explanation of the method to calculate an optimal many-to-one matching under the assumptions.

In our heuristic search, the states or objects are the partial matchings between C and P. The generation of new states from a state is performed by considering paths of different lengths, starting with the feature and the joint suggested by the extension rule and matching the ending of each path (and its adjacent features) with the corresponding joint (and its adjacent features) in the prototype. The heuristic function that decides which state is closer to a solution depends on the cost of the partial matching of the state. The search is initialized with the matching of all possible features from C to a predefined and required feature from P. A state is considered a solution if its corresponding matching cannot be expanded.

There are two important situations that can happen during the matching process. First, a joint d in the prototype, that is already matched to a joint in the candidate, is matched again to a different joint, as suggested by the extension rule; in this case, a transformation associates the joints matching d, in the context of the particular prototype. The second important case arises when two independent paths are matched to the same feature g in the prototype; some geometric constraints are checked and the two paths are transformed into only one path by filling the gap between them.

The key observation when matching two graphs is that there are many futile explorations. For example, when considering the mapping of a joint of a graph that represents the top of a "4" with the joint on the bottom of the prototype of "4", features will match poorly if they do match at all; in this case, the accumulated matching cost indicates that it is a poor option. Thus, the geometrical distribution of adjacent features is a good indication of how far a path should go.

We use a cheapest-first strategy that selects for processing the state with lowest partial matching cost. This guarantees that the first goal state chosen for extension is also the cheapest solution, because the search is *admissible* [17] . This strategy suppresses any information about the cost of the extension of a matching, so actually there is no heuristic search involved, and it guarantees to find the optimal solution under the assumptions.

A solution interprets paths of features in the candidate and matches both endings of each of the paths. A multiple-to-one matching can be seen as a high level discrimination of the singularities (joints) into "important" or "not important" according to whether they are path endings, or not.

In the next section, we will explain how the results of comparing an input graph with several prototypes are combined for classification.

6.4. Complexity analysis

To analyze the inexact matching algorithm we have to consider the complexity of algorithm *Match_join*, and the number of times that it is called by the search procedure. Assume that m, is the maximum between the number of features in the prototype and in the candidate. Let $\Delta(G)$ be the largest degree of a joint in a graph G. A dynamic programming string matching algorithm finds a matching of the adjacent features of two joints in at most

$\Delta(P)\Delta(C)$ steps. However, the search procedure can call this routine 2^m times, in the worse case. Therefore, the algorithm is exponential in the number of features in the graphs.

It is very important to notice that the number of features or joints in a feature graph is usually under 5. Therefore, the real time of execution is relatively small.

7. Distance between graphs

A distance measure between graphs representing character shapes is proposed in this section. We believe that different kind of measures for transformations can be associated to represent the overall amount of transformation of one shape with respect to another one. In addition, we think that humans know which type of transformation is more important than other, and therefore, we attempt to simulate this knowledge.

The effect of comparing transformation costs when matching a candidate to a prototype is the minimization of the transformations required for the identification with the prototype. However, to accomplish the classification process, we are also required to compare transformations across classes. The task of comparing two deformations with respect to only one prototype needs to be extended to the task of comparing two deformations with respect to different prototypes; situations like the following have to decided: what is it more costly: an extra stroke in a "1" or the smoothing of a sharp angle in a "2" ?

To tackle this problem, we need to extract directly from the matching the different kind of transformations we think will have different importance degrees between the prototypes. We divide the transformation costs in the following five types:

1. geometrical distortion (deviation of lengths and angles between strokes, and differences of lengths between arcs),
2. morphological distortion (deviations of initial and final angles between arcs, size of a bypassed concave angle, sharpness of a smoothed angle, and deviation of straightened arcs and lists of strokes from the final stroke),
3. distance between associated joints (size of gaps),
4. size of non-matched features in the candidate, and
5. size of non-matched features in the prototype.

Since each class P_i (prototype) can give a different level of importance to each cost type j, a weight constant $w_{i,j}$ is introduced. Let C be a fix candidate graph. The value of cost type j of C with respect to the prototype P_i is represented as $c_{i,j}$. The distance to the prototype P_i is defined as:

$$D(i) = \sum_{j=1}^{5} w_{i,j} c_{i,j}.$$

If N is the number of prototypes, the candidate graph is classified as belonging to the class of the prototype P_k if $D(k)$ is the minimum among the set $\{D(i), i=1, ..., N\}$. If another value $D(l)$ is close to the minimum, a disambiguation function is invocated that concentrates in discriminatory features for the prototypes P_k and P_l.

The weights ($w_{i,j}$) were set manually. The order of importance of the cost types for all the classes is the same as the order in which they were presented above, where the less important one is the geometrical cost and the most important one is the size of the non-matched features in the prototype. So far, their precise values has not been decisive, with the important exception of the weights used to define the frontier among character shapes (for example, between "0" and "6"). Sensitivity analysis showed that weights can be varied greatly without affecting significantly the recognition rate of numerals.

8. Results

The shape classifier is part of an address recognition system based on feature extraction from grey scale described in another paper [5] . The system recognized numerals using less than 2 prototypes per class on the average. This classifier technique has been applied successfully to over 16,000 numerals belonging to a USPS database of real printed addresses. The database is a mixture of typical mail plus mail rejected of the current postal OCR readers, so it is more challenging than an average sample (for example, see Figure 11). The recognition results are: 96.9% recognition rate, 0.5% rejection rate, 2.6% error rate. The classification time was 4 numerals per second on a SUN SPARCstation 2, without including the feature extraction time. No effort has been made to optimize the program for speed.

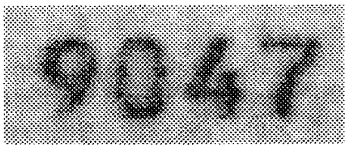

Figure 11. A low contrast input image.

The main cause of errors is broken characters. When features are disconnected (because of white stripes or in some dot matrix printed characters), the classifier may fail to connect the features in the appropriate places. A new module is being implemented to recognize disconnected parts of features and complete the corresponding paths according to the prototype. Other errors include shape confusions, transformation errors and bad feature extraction.

9. Conclusion

A structural method for character recognition was introduced. We defined a matching procedure between two graphs which represents the minimum morphological transformations that a graph will undergo so the matching is possible. The multiple-to-one matching defined allows paths in the candidate graph to be matched (transformed) to features in the prototypes. The cheapest matching detects globally the singular points of the skeleton that are important for the identification of the shape. In other words, the importance of the joints and features is not estimated locally.

The variability of actual character shapes is absorbed by the optimization process and it is modeled in terms of a sequence of transformations and an associated cost. Character shape definitions are simple: few prototypes are needed and the number of features in each of them is small. This method diverges from previous works which incorporate a wide number of forms into the shape definitions. Statistical classifiers (e.g., neuronal networks), that learn the shape differences, and structural classifiers with large number of prototypes are examples of systems that look for completeness by exhaustion.

It should be noted that our system has not been automaticly trained. About 1,000 samples of numerals were used to select the prototypes and were processed repeatedly to revise the transformation rules and their associated costs. However, no significant degradation occurred on the testing data (less than 0.4%), and the sensitivity analysis showed that the system is not tightly tuned. This demonstrates the system's robustness and the generality of the method. Also, some tests with handwritten numerals reveal promising results.

Future research includes the extension of the method to deal with handwritten character recognition, and to help in the segmentation process. The use of recognition for segmentation emphasizes the fact that the knowledge of the shapes that are to be recognized defines the association and interpretation of the input features. We believe that shape analysis is a subjective task in which the knowledge of the possible models guides the recognition of the input shapes.

References

1. S. Mori, C. Suen, and K. Yamamoto, "Historical Review of OCR Reseach and Development," *IEEE Proceedings*, vol. 80, pp. 1029-1058, July 1992.
2. M. Bokser, "Omnidocument Technologies," *IEEE Proceedings*, vol. 80, pp. 1066-1078, July 1992.
3. L. Lam and C. Suen, "Structural Classification and Relaxation Matching of Totally Unconstrained Hadwritten Zip-Code Numbers," *Pattern Recognition*, vol. 21, no. 1, pp. 19-31, 1988.
4. L. Pintsov, "One View of the Methodology in Handwriting Character Recognition," *SPIE/IS&T's Symposium on Electronic Imaging, Science and Technology*, p. 115, San Jose, California, February 9-14, 1992.
5. W. J. Sakoda, J. Zhou, and T. Pavlidis, "Feature Extraction Directly from Gray Scale with Applications to an Address Recognition System," *Postal Service, 5th. Advanced Technology Conference.*, Washington, D.C., Nov 30 - Dec 2, 1992. To appear.
6. S. Kahan, T. Pavlidis, and H. S. Baird, "On the recognition of printed characters of any font and size," *IEEE Trans. PAMI*, vol. 9, no. 2, pp. 274-288, 1987.
7. S. Xie and M. Suk, "On Machine Recognition of Handprinted Chinese Characters by Feature Relaxation," *Pattern Recognition*, vol. 21, no. 1, pp. 1-7, 1988.
8. S. Lu, Y. Ren, and C. Suen, "Hierarchical Attributed Graph Representation and Recognition of Handwritten Chinese Characters," *Pattern Recognition*, vol. 24, no. 7, pp. 617-632, 1991.
9. H. Yamada, "Contour DP Matching Method and its Application to Handprinted Chinese Character Recognition," *Proc. 7th Intern. Conf. on Pattern Recognition*, pp. 389-392, Montreal, Canada, July, 1984.
10. J. Hopcroft and J. Wong, "Linear Time Algorithm for Isomorphism of Planar Graphs," *Proc. 6th Ann. ACM Symp. on Theory of Computing*, pp. 172-184, New York.
11. A. Sanfeliu and K. Fu, "A Distance Measure Between Attributed Relational Graphs for Pattern Recognition," *IEEE trans. on SMC*, vol. 13, no. 3, pp. 353-362, May 1983.
12. D. Burr, "Elastic Matching of Line Drawings," *Proc. 5th Intern. Conf. on Pattern Recognition*, pp. 223-228, Miami Beach, Florida, Dec, 1980.
13. L. Shapiro and R. Haralick, "Structural Descriptions and Inexact Matching," *IEEE trans. on PAMI*, vol. 3, no. 5, pp. 504-519, Sep. 1981.
14. F. Harary, *Graph Theory*, p. 107, Addison-Wesley, 1972.
15. M. Garey and D. Johnson, *Computers and Intractability: a guide to the theory of NP-completeness*, 1979.
16. N. Robertson and P. D. Seymour, "Disjoint paths - a survey," *SIAM J. Algebraic Discrete Methods*, vol. 6, pp. 300-305, 1985.
17. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, p. 44, Addison-Wesley , Paris, France, 1984.