Character Recognition Without Segmentation

Jairo Rocha and Theo Pavlidis

Abstract—**A segmentation-free approach to OCR is presented as part of a knowledge-based word interpretation model. This new method is based on the recognition of subgraphs homeomorphic to previously defined prototypes of characters [16]. Gaps are identified as potential parts of characters by implementing a variant of the notion of relative neighborhood used in computational perception. In the system, each subgraph of strokes that matches a previously defined character prototype is recognized anywhere in the word even if it corresponds to a broken character or to a character touching another one. The characters are detected in the order defined by the matching quality. Each subgraph that is recognized is introduced as a node in a directed net that compiles different alternatives of interpretation of the features in the feature graph. A path in the net represents a consistent succession of characters in the word. The method allows the recognition of characters that overlap or that are underlined. A final search for the optimal path under certain criteria gives the best interpretation of the word features. The character recognizer uses a flexible matching between the features and a flexible grouping of the individual features to be matched. Broken characters are recognized by looking for gaps between features that may be interpreted as part of a character. Touching characters are recognized because the matching allows nonmatched adjacent strokes. The recognition results of this system for over 24,000 printed numeral characters belonging to a USPS database and on some hand-printed words confirmed the method's high robustness level.**

Index Terms—**Character recognition without segmentation, broken character recognition, touching character recognition, homeomorphic subgraph matching, relative neighborhood graph.**

I. Introduction

Character segmentation is a key requirement that determines the usability of conventional OCR systems. The classical paradigm for character recognition has three steps: segmentation, feature extraction, and classification. However, proper segmentation requires a priori knowledge of the patterns that form meaningful units, which implies recognition capability. To tackle this dilemma, we intend to use the information carried by prototypes of each character to decide whether a particular stroke belongs to two adjacent characters in a word.

According to Lecolinet and Crettez [11], two main classes of methods for recognition of words have been proposed: analytical and global. The first one attempts to segment words into letters to recognize them, while the other attempts to recognize the words globally. We will not discuss the global approach further because words made up of numerals (postal codes, for example) require full understanding of each symbol. There are basically two kinds of analytical methods:

1) Analytic methods with *external* segmentation, where segmentation precedes the recognition stage, as in [5].
2) Analytic methods with *internal* segmentation, where segmentation and recognition are performed at the same time, as in [20].

Lecolinet and Crettez pointed out what many others have said before: segmentation before recognition is paradoxical, because without understanding the symbols, there are no good criteria to avoid errors of segmentation. They also believe that basing segmentation on recognition has some drawbacks: The recognition of words with illegible characters is more difficult because the proper segmentation is not known for nonrecognized parts of the word.

Manuscript received July 2, 1994; revised Feb. 24, 1995.

J. Rocha is with the Departament de Matematiques I Informatica, Universitat de les Illes Belears, 07075 Palma de Mallaorca, Spain; e-mail jairo@ipc4.uib.es.

T. Pavlidis is with the Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400; e-mail: theo@sbcs.sunysb.edu.

IEEECS Log Number P95076.

Tsujimoto and Asada's segmentation method [20] considers several candidates for break positions, which are then confirmed by recognition. This means that not all break position candidates are used but only the ones that produce meaningful characters. The candidate break positions are organized in a decision tree. A path on this tree represents a list of possible breaks on the word. Their method, however, performs only rectilinear breaks between characters and depends on projections to determine the possible breaks. Casey and Nagy [2] built a recursive segmentation and recognition procedure. Their system divides words in blocks and tries to decompose all blocks wider than a certain adaptive threshold. Initially, a viewing window is set to the size of the block. If a classifier rejects the block, then the viewing window is narrowed from the right-hand side and the classifier is applied to the smaller window. Gradual narrowing is applied until a meaningful character is found in the window. The window is then reset to the residue of the original block. If the recognition of a residue fails, the system can backtrack to the previous decision. However, the system can segment only vertically separable characters by performing a vertical scan of the image. Besides, if any subpart of a block is illegible, partial recognition of the block is not possible.

Edelman et al. [3] proposed a method for cursive handwriting recognition that permits recognition of character instances that appear embedded in connected strings. The main difference between their method and ours is that they search for affine transformations of predefined stroke types, while we search for subgraphs homeomorphic to predefined prototypes. Two strokes can be homeomorphic without being affine and vice versa, so the methods are not comparable.

Statistical representation has become popular with the use of Hidden Markov Models (HMMs) [6]. Most of them divide the word into simple strokes totally ordered that correspond to parts of characters. By training, they can learn transition probabilities between characters and observation probabilities of strokes as part of characters. HMMs have been relatively successful on off-line handwriting recognition and differ from our approach in that we avoid training.

Most other internal methods, e.g., [1], are part of on-line systems, in which there is temporal information that is useful to split characters.

In this paper we present a framework for word interpretation in the presence of touching or broken characters, crossbars, and noise. This is possible because we are able to recognize a character even if strokes not belonging to the character are attached to it. An implementation of this internal method is given that recognizes touching and broken numeral and capital letters in the presence of noise and underlines. Since the algorithm does not guess the possible break positions but finds structures that match previously compiled models, segmentation becomes unnecessary. Only for computational efficiency some very conservative segmentation may be performed. The system can partially recognize a word if illegible characters are present.

The following is an outline of the word classifier. First, the image of a word is converted into a feature graph. Features are extracted directly from gray-scale images using the method described in [19]. Edges of the feature graph are skeletons of the input strokes. Graph nodes correspond to singularities (inflection points, branch points, sharp corners, and ending points) on the strokes.

The second part of the classifier is the identification of gaps between features and has been developed to tackle the recognition of broken characters. A gap is a region where two strokes are in proximity. Thus, at the end of this stage, a new kind of edges, called *bridges*, are introduced into the feature graph. Bridges represent the spatial relationships of otherwise disconnected components. This will be explained in more detail in Section II.

The third stage of the classifier is the recognition of meaningful subgraphs of the feature graph, using the representation and method described in [16] for segmented characters, that is reviewed in Section III. Thus this paper extends [16] to unsegmented words.

Each subgraph that matches a prototype well is introduced into a net that compiles different alternatives of interpretation of the word. The final stage organizes the net in such a way that a path from the beginning to the the end of the net represents a consistent succession of characters that are found in the word, when reading from left to right. Section IV describes the net and its construction. The cheapest path in the net globally finds the interpretation that best represents the input, according to the matching costs of the subgraphs, the size of the nonmatched strokes, and layout parameters.

Section V presents experimental results, and Section VI gives some conclusions.

II. Gaps on Broken Characters

Template-matching algorithms can recognize broken versions of character templates without major difficulty. This is true also for other classifiers that use as features the full image (pixels) of the character description [17]. The reason for this phenomenon is simple: The difference between a broken image and the ideal one is relatively small. For template-matching, if a gap is introduced in an image and it does not change the interpretation of the image, it increases the distance from the image to all templates so the recognition is not affected. Statistical classifiers can be trained to recognize the most common broken characters, and in this context a broken character is better seen as the result of a feature extractor anomaly.

In contrast, most structural approaches to classification are confused by broken characters, since a broken piece may drastically change the representation of the character structure. However, some classifiers (e.g., [13]) that represent characters as spatially related parts do not assume connectivity of the parts. Therefore, they can handle broken characters when the gaps appear between the parts of a predefined model. However, if a gap splits a part, these classifiers do not know whether or not to associate two strokes separated by that gap.

Our approach can be seen as an application of the same philosophy that allows recognition of broken characters by a template-matching technique: the representations of a character with and without a gap look alike modulo the graph matching. In other words, the representation does not decide if the gap corresponds to a missing part or to a real separation of strokes, but allows both possibilities to coexist until the matching with models decides which is the best interpretation of the gap.

We are not aware of any other classifier presented in the literature that has the property defined above. This property for gap representation is just an extension of our paradigm that all singularities on the image should appear on the input graph, but only in the context of the models is the interpretation of the singularities given.

Thus, gaps should be represented as character parts that may be missing or not from the ideal image. Gaps are now positive features with ambiguous interpretations, as are strings of connected strokes. This brings up two problems: first, how to identify all gaps, in other words, how to define a gap, and, second, how to consider only possible useful ones, since we want to increase the number of edges in the input graph as little as possible. This section addresses these two problems.

A. Identification of Gaps

The problem of defining a shape of a set of disjoint points or segments is one of the subjects of study of computational morphology. There are several computational works on how to define the shape of a set of points ([21], [22], [15], [10], [4]), but not as many on how to obtain a shape from a set of segments [23]. One of the reasons for this phenomenon is the extra number of visual attributes that segments have that make it more difficult to organize the input.

Perceptual organization refers to the process in which relevant groupings and structures are derived from an input image without prior knowledge of its content. What we present in this section is the use of some perceptual properties to identify visually important gaps on the low-level input graph, without using knowledge of the characters we are looking for. In other words, the final description that is the result of the gap identification is independent of the conceptually-driven interpretation of the whole description.

We implement a variant of the notion of relative neighborhood that has been successfully used in computational perception [21]. Toussaint defines the *relative neighborhood graph, RNG,* of a set of points on the plane. Two points in the set are relative neighbors if they are as close to each other as they are to any other point in the set. Formally, p and q are relative neighbors if there does not exist another point z of the set such that

$$d(p, z) < d(p, q) \text{ and } d(q, z) < d(p, q)$$

where $d(p, q)$ denotes the Euclidean distance between point p and q.

To put it still another way, two points are neighbors if there is no other point in a forbidden region between them defined by the *lune* of Fig. 1. Fig. 2 shows the *RNG* of a set of points.

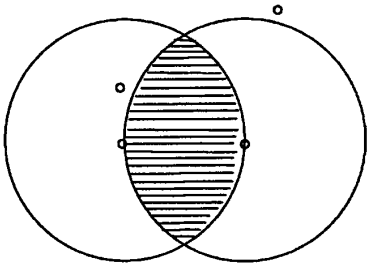

Fig. 1. The lune between two points to decide if they are neighbors.

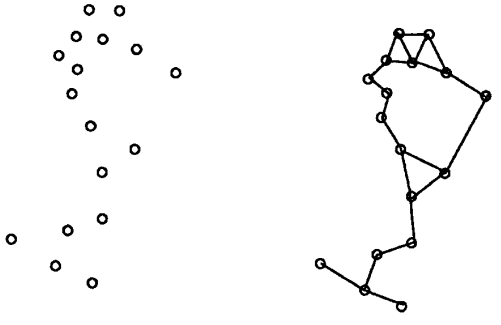

Fig. 2. The *RNG* of a set of points.

Our first approach to apply this notion of proximity is to consider as a point set all nodes of the input graph G. An optimal algorithm to find the *RNG* can be implemented in $O(n \log n)$ time [18], where n is the number of joints.

If two nodes are adjacent in the *RNG* but they are not connected through a stroke (segment) in G, then a marked segment, called *bridge*, is introduced between the nodes. The bridges identify the gaps.

This method has an important drawback: Bridges that are introduced may not be visually correct, or some bridges may be missing, because bridges are added only between joints and sometimes the closest place between two strokes is far from their endings. To solve the problem we add to the point set the closest points from each stroke of the graph G to each node of G. As a result, points in the middle of strokes or arcs may need to be calculated, and new nodes may need to be introduced subdividing edges, if bridges adjacent to them are found. The insertion of bridges and nodes to G creates a new graph that we call $\hat{G}$.

Figs. 3 and 4 show input that is recognized correctly after the bridges are added to the feature graphs.

Fig. 3. Input recognized correctly by the system after bridges are introduced.

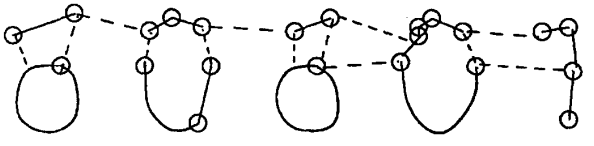

Fig. 4. The broken lines represent bridges on the graph.

The notion of identifying a gap by a bridge was first introduced in [16] for segmented characters. The same concept is now applied to the entire word graph. Bridges will connect adjacent connected components, expressing their relative closeness. In this way, pieces that make sense to be interpreted as only one character can be recognized. On the other hand, if bridges connect two characters, they will not be used in the interpretation because the characters will make sense without the use of bridges.

B. Efficiency Considerations

The size of the point set used depends on the number of features of G and the number of some other points that are internal to the edges; thus the point set may not be small. However, the time spent on computing the bridges is so small relative to the recognition time that it is not a major concern of this research.

On the other hand, we are interested in reducing the number of bridges added to G, because the recognition time is greatly affected by it. Two assumptions are made to reduce the number of bridges.

The first simplification is that only nodes of degree equal to one (endings) need to be connected to other strokes using bridges. Consequently, at least one of the nodes adjacent to a bridge is of degree one in G. In this way, bridges between "well-connected" parts of the characters are not introduced. This assumption has produced errors in the cases where the minimum distance between two strokes takes place far from joints of degree one. Therefore, it should be eliminated or relaxed in future implementations.

The second assumption is that we expect to add few bridges because the skeletons are highly connected. Dot matrix-printed characters are handled by the feature extractor, in the sense that sequences of small ridges and saddles are detected on the gray-scale surface and recognized as dot print [19]. In case of a high number of endings per unit, we assume that this is due to the presence of disconnected features from the background. Consequently, the disconnection should be preserved, so the big components can be distinguished as characters. Notice that it is difficult to recognize characters made of broken lines when the background is composed of small segments. Thus, when a lot of disconnected components are present in an image, only bridges of small size are added between strokes. Some experiments were done to determine a partially lineal function that relates the number of endings per unit with the size of the gap that accepts a bridge. Fig. 5 presents the function used. This adaptive introduction of bridges proved to be very useful in handling noisy images without affecting the other ones: The system is 18% faster, the segmentation is 1.4% better, and the overall recognition rate is 0.3% better than the system without the adaptive schema.

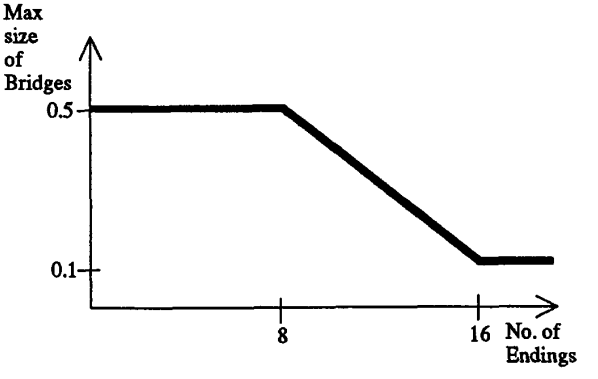

Fig. 5. Function that defines the maximum size of a bridge that is introduced.

C. Possible Improvements

It has been found experimentally that most of the gaps that are intended to be found are identified. However, there are certain problematic cases that will be discussed in this section. The implementation presented does not use full information of the input graph, since for the purpose of finding the bridges the input is reduced to a set of isolated points.

A more complete approach would imply the extension of the *RNG* definition from points to segments and use of the perceptual properties that groups of segments have.

Toriwaki and Yokoi [23] define the concept of *relative neighbor figures, RNF,* on the plane. Intuitively, two figures are *RNF* if two points that realize the minimum distance between the figures are relative neighbors according to the definition for points, in the sense that no point of the other figure is in the forbidden region between the points.

The type of input graph edge can be either arc or segment. Arcs are represented as convex polylines (list of segments). If we consider the set of all segments, including segments from arcs, as the set of figures, then bridges calculated using the *RNG* are not too different from the ones calculated using the *RNF* concept. The reason for this is that the minimum distance between two non-parallel segments is realized in at least one of their endings. The main difference lies in the fact that the *RNF* version for segments does not allow intersection between *lunes* with segments, while an *RNG* constraints the intersection of *lunes* with only some particular points of the segments.

Although we do not think that graphs calculated using the *RNF* concept would include better or fewer bridges, it is worth trying this more general version.

Fig. 6. Smooth connection is an important criterion for bridge generation. In both cases, the shortest bridge is not the best.

It seems more promising to use techniques of perceptual organization to complete strokes. In particular, it is well-known from the Gestalt school that elements that lie along a common line are easily grouped together [12]. Therefore, bridges should be connected as smoothly as possible to at least one of the strokes to which they are adjacent. This criterion would introduce better bridges on the skeletons showed in Fig. 6. Since the criterion of distance cannot be forgotten, the two of them should be combined. This difficult task is carried out in [9] by given weights of importance to smoothness, distance, and existence of other clues of connectivity.

III. Matching of Subgraphs

Given the feature graph for a word, our method finds subgraphs that are homeomorphic to some prototype. A distance function between graphs measures the amount of distortion between subgraph and prototype. This distance function represents the minimum geometrical transformations that a graph will undergo so that the matching is possible. Thus, it is used as a measure of the quality of the matching. The *cost* of matching two graphs is the distance between the graphs.

The global graph transformation that introduces bridges changes the topology of the original graph G. However, the following property is true: For a given prototype P, if there is a subgraph of G that is homeomorphic to P, then there is a subgraph of $\hat{G}$ that is homeomorphic to P. The reason for this is as follows: The subdivision of edges does not change the topology, and the new edges (bridges) do not reduce the number of subgraphs. $\hat{G}$ contains the same or more subgraphs than G, therefore, it is easier to find a meaningful interpretation.

The cost function and the algorithm that is used to find subgraphs are basically the same used for recognition of segmented words, described in more detail in [16]. In order to find subgraphs of the entire word graph that match prototypes, the recognition process is initiated on all strokes of the graph. The prototypes will guide the best recognition that contains the initial stroke. The cost of the subgraph matching does not include strokes that are not matched in the candidate, since they may belong to a different character. It does include a cost associated to the size of the bridges that are matched during the recognition because a bridge does not represent a stroke that is present in the input. Therefore, interpretations that use as few bridges as possible are preferred. Although the bridges are calculated without using knowledge of the shapes to be recognized, the decision of which bridges are actually used to fill gaps is done by the high-level matching procedure.

IV. Word Interpretations and Net

The main assumption of our method is that the characters of the word are laid out on one-dimensional relative positions. In particular, we assume that components are not arranged vertically (e.g., mathematical expressions are not considered), which is the case of text, postal addresses, etc. To find the two-dimensional relative positions of expressions, a more complex representation is needed. Okamoto and Miao [14] represent mathematical expressions as trees, although they are not concerned about segmentation.

We define an *interpretation* as a directed path of subgraphs matched to prototypes. Each node in the path corresponds to a subgraph, and the edges between the nodes represent an order from left to right of the subgraphs. There are two extra nodes in the path: the initial and the final. All strokes of the word that are not interpreted in any of the nodes of the path are represented on the edges of the path, as will be explained later. This includes the strokes of crossbars. Thus, crossbars are recognized using the features associated to edges of a path.

The quality of an interpretation is measured using the cost associated to each path and depends on the following costs: matching cost of the subgraphs, size of strokes shared by adjacent subgraphs, smoothness of the crossbars, size of strokes nonmatched, not including bridges, and layout quality, including consistency of vertical and horizontal sizes of all characters in the word and vertical alignment.

The previous costs define a fuzzy implementation of constraints, such as nonoverlapping of subgraphs and layout consistency. We cannot rigidly while enforce the constraints because they might not be present on real input.

Crossbars of a path can be identified as follows: Find a string of edges that uses as many strokes nonmatched in the path as possible and that is smooth. Notice that it does not matter if the string of edges uses strokes already matched to some character, because it is natural to expect some overlapping. Characters in the interpretation have been already recognized even in the presence of the crossbars, so there is no need for removing the lines. This is the main difference between our framework and other methods that remove interfering strokes without using domain knowledge, as in [7].

It is not expected that two adjacent subgraphs share edges, which is the reason why common strokes reduce the quality of the interpretation. This avoids cases such as a sample of a "2" with a vertical serif in the bottom to be interpreted as a "7" followed by a "4." However, common strokes can be calculated and, according to the context, be accepted; for example, horizontal serifs and long horizontal strokes are usually shared, so it is possible to divide them between the two subgraphs and recalculate the matching costs for each subgraph. Notice that because a path contains a full interpretation of the word, it is possible to make contextual decisions.

The layout quality specifications are part of the assumptions about the interpretation. They express the fact that printed numerals and capital letters are expected to have consistent height, width, and spacing. They can be calculated particularly for each path, so they do not depend on fixed expected ratios or sizes.

There is another consistency check that this representation allows: Subgraphs of the path that are matched to the same class should be similar, at the image level, if a single font is assumed per word. One way to implement this constraint is by template matching of the corresponding images. Notice that this is equivalent to making a second focused look at the original image after a first interpretation is attempted.

For implementation purposes, it is important to realize that different interpretations, i.e., paths, can share nodes. In other words, they can be organized so no duplication of information is necessary. Once meaningful subgraphs are recognized, the word recognition problem is reduced to the problem of organizing them from left to right and finding the best interpretation. We create a directed graph of interpretations that we call *net* for this purpose.

Each net node represents a subgraph that is recognized by the classifier. The fields associated with a node are: a label for the symbol that it represents, the list of edges of the subgraph, the matching cost, and the coordinates of the bounding box.

An edge between two nodes in the net represents a relative spatial order from left to right between the associated subgraphs. Fig. 8 shows an example of a net, where only some of the edges have been included; the transitive closure of the relation shown would be a more accurate representation, according to the definitions given below.

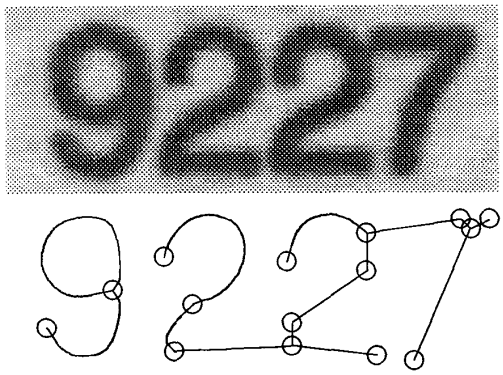

Fig. 7. The feature graph of a word, without including bridges.

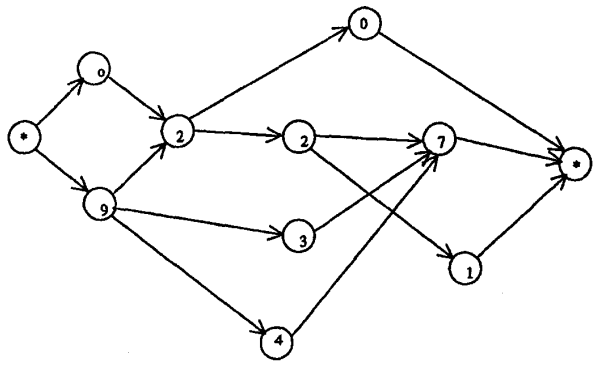

Fig. 8. Net for the graph on the previous figure.

Each time the classifier identifies a subgraph, a node is introduced into the net. The edges adjacent to the new node are decided according to a partial order defined among the subgraphs associated with the nodes. The partial order is defined as follows:

Given two subgraphs G_1 and G_2, $G_1 < G_2$ if

1) G_1 does not share many edges with G_2; formally,

$$size(G_1 \cap G_2) < size(G_1 - G_2),$$

$$size(G_1 \cap G_2) < size(G_2 - G_1)$$

where $size(G)$ represents the sum of the lengths of edges in G, and $G_1 - G_2$ is the graph G_1 in which all edges that appear in G_2 are eliminated;

2) The bounding boxes of the subgraphs can be situated one to the left of the other one. Formally, let x_1 and X_1 be the left and right sides of the bounding box of G_1, and similarly for G_2. The following condition must be true:

$$X_1 < x_2 \text{ or}$$

$$x_1 \leq x_2 \text{ and } X_1 < X_2 \text{ or}$$

$$x_1 < x_2 \text{ and } X_1 \leq X_2.$$

Also, any node is to the right of the initial node and to the left of the final node.

The fields associated with an edge that connects two nodes are:

1) list and size of strokes nonmatched that are before the node on the right and are not before the node on the left;
2) cost for differences in vertical layout of associated graphs;
3) amount of horizontal overlapping of the bounding boxes of the nodes.

The cost of an edge is the sum of the three values mentioned above and represents the cost of assuming that the node on the left precedes immediately the one on the right.

If there is a character that is illegible, the cost of not matching its strokes appears on edges between characters before and after the illegible one.

Once the net is built, it compiles costs for meaningful subgraphs, nonmatched strokes, and relative layout of the subgraphs. Therefore, a cheapest path in the net finds the sequence of symbols that best interprets the input, leaves as few strokes as possible without interpretation, and defines a uniform layout.

There are several important points to notice in this implementation of interpretations. The layout cost of the edges of the net is just an approximation of a more global one calculated on the path and not just locally between two nodes. Also, in a given path, a consideration of the strokes not matched in the nodes would recognize crossbars. But this is also a global operation on the path and, therefore, it has been replaced by the recognition of hyphens between two characters. These two simplifications are made so a cheapest path search algorithm can be used. Future research includes the improvement of the search strategy on the net so a global layout is enforced and the crossbars are found more accurately.

One case worth noticing is the one of underlined words. Typically, the underline, like the one in Fig. 9, is incomparable with the rest of the subgraphs; in other words, the underline and some of the character subgraphs are never in the same path because these are neither on the left nor on the right of the underline; therefore the best path in the net of Fig. 10 leaves without interpretation the underline, which is better than leaving some characters unmatched.

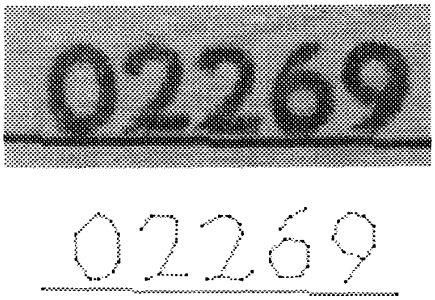

Fig. 9. The underline is incompatible with the other subgraphs.

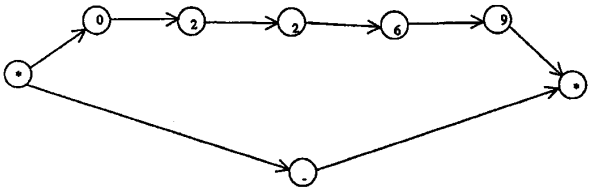

Fig. 10. In the best path, the underline of Fig. 9 is left unmatched because the alternative path leaves all character strokes unmatched.

The main objective of the implementation is to prove the viability of this framework for word interpretation and not to present an efficient way of doing it. Therefore, very simple heuristics have been implemented to bound the number of considered subgraphs: The worst-case complexity is exponential in the number of edges that fits inside a square centered on the first joint matched and of side length equal to 1.5 of the height of the word. This number is usually small. The cheapest path algorithm is polynomial in the number of subgraphs considered, but this is exponential in the worst case.

Currently being investigated is the use of lexical context to find a meaningful interpretation. The issues of concern are the combination of net costs and probabilities that come from n-grams statistic and the simultaneous search of paths in the net and alternatives in a dictionary. In the first case, it is possible to associate probabilities to costs, but training would be necessary. In the second case, complex data structures would have to be considered as in [8].

V. Results

This classifier technique was applied to 5,282 words (over 24,000 numerals) belonging to a USPS database of real printed addresses. The recognition results are: 96.5% of the words were correctly segmented, and 98.1% of the characters were correctly recognized. The classification time was three 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.

The cause of the segmentation errors are (in parentheses, percentage of the total number of wrongly segmented words):

1) Background noise (53%): In this case the number of features is very high, and most of the time the classifier does not have enough time to reach a solution (maximum time = 10 min). Most of these words can be recognized correctly if the background texture is labeled as belonging to other kinds of features. The classifier may avoid processing them unless they are needed to improve the recognition.
2) Extra or missing hyphens (24%): Some noisy strokes are interpreted as hyphens and vice versa. Criteria to discriminate hyphens include change in the pitch between characters, relative width with respect to the other characters, vertical position, and absolute size. Other criteria to be integrated should determine if the stroke of the candidate hyphen agrees in width and intensity with the strokes of recognized characters.
3) Shape confusion (15%): When characters are touching, the recognition is entirely based on the interpretation of the connected features. Therefore, it is possible that two connected characters can be interpreted as three characters. A better cost function for recognition and a context-dependent evaluation of costs for touching characters in the net will solve most of these cases.
4) Other errors (8%): Truth errors, administrative errors.

The causes of recognition errors of individual characters are (in parentheses, percentage of the total character recognition errors due to specific cause) [16]:

- Shape confusion (24%): some characters are confused with similar ones, such as "1"s with "7";
- Noise and background interference (20%);
- Feature extraction error (17%): missing features or spurious strokes that confuse the matching;
- Matching cost error (14%): although the matching of a character with the prototypes is correct, the associated cost of the matchings is not, because for some classes, the matching cost should be calculated using particular constraints (e.g., symmetry) on the distance function;
- Edge path errors (10%): when the assumptions for path generation (i.e., good continuity) are not satisfied, the intended meaning of the characters is never found;
- Gaps not used (9%): gaps should tend to preserve smoothness of the connected strokes; and
- Truth errors (6%).

The results are very encouraging because of the difficult cases that the system can solve and the high recognition rate by itself.

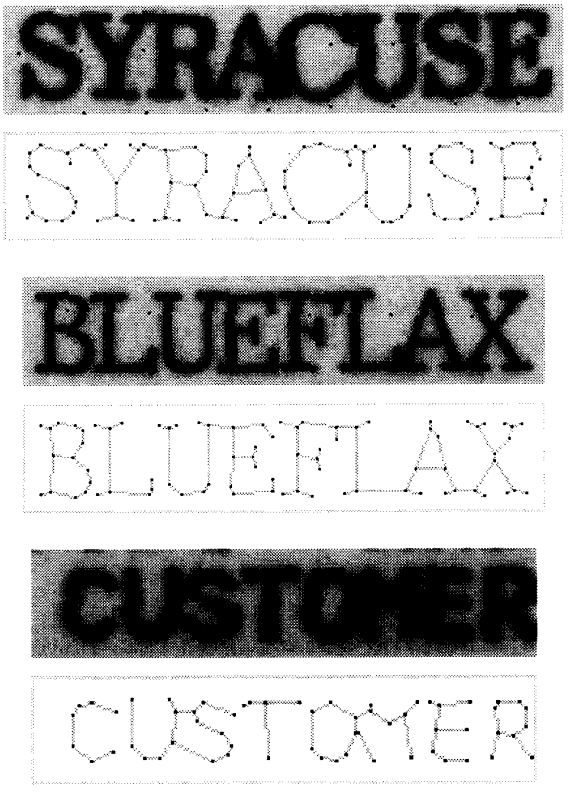

Fig. 11. Top: proper segmentation with large overlap of A and C. Middle: touching characters with seams in the image. Bottom: correct segmentation of overlapped characters.

When the prototypes for capitals are used (42 prototypes for 26 classes), the system can recognize all the characters in such text as shown in Fig. 11. The system was applied to 12,195 capital words (over 50,000 capital characters) belonging also to a USPS database. The recognition results are: 90.9% of the words were correctly segmented, and 94.13% of the characters were correctly recognized. The

causes of the errors are basically the same as described above, but the distribution of the errors varies: Most of the errors are shape confusions due to the greater number of classes. The system needs to include particular constraints that define relations among features of each class, as, for example, similar length, similar orientation, and symmetry among features in prototypes, etc. Ongoing research includes the definition of these constraints that are enforced after the subgraphs are identified, improving the discriminatory power of the classifier.

VI. Conclusions

We have proposed a system for segmentation-free recognition specially suited for the processing of touching and broken characters. This is accomplished by recognizing subgraphs even if they have gaps or strokes from other adjacent characters.

Broken characters have been a challenge in the past because the gaps between features cannot be filled correctly without knowing the symbol where they reside. Similarly, touching characters make it difficult to extract the exact set of features needed for identification. Our structural method allows the prototypes to guide the interpretation of strokes and gaps. This approach is fundamentally more complex than the ones that use simple typographic clues, but it is able to solve cases that are more difficult.

A global consideration of several meaningful alternatives in the net yields the word interpretation. The net can easily manage additional information about absolute and relative occurrence of symbols in a language of interest.

Future research includes studying search strategies in the net that yield better use of global layout and precise interpretation of crossbars. From the efficiency point of view, there are several improvements that are worth investigating. The present implementation is divided between calculating all possible meaningful subgraphs and then deciding for a consistent interpretation of them. These two processes should be combined so that meaningful subgraphs are searched in places that produce meaningful interpretations, thus bounding the number of possible subgraphs.

The framework for word interpretation presented in this paper can be used with any character classifier that is able to recognize subgraphs as meaningful units. In the case presented here, further research is needed on the calculation of the matching quality. For this purpose, the development of particular constraints that enforce certain relative relations between features of one class is currently being implemented.

Acknowledgments

The authors wish to express their gratitude to Professor Angelo Marcelli for his useful comments and to Dr. William Sakoda for his help on the error analysis.

This research was supported by U.S. Postal Service Contract No. 10423091C3770 and National Science Foundation Grant IRI 92-209057.

References

[1] M. Berthod and S. Ahyan, "On-line cursive script recognition: A structural approach," *Proc. Fifth Int'l Conf. Pattern Recognition*, pp. 723-725, 1980.

[2] R.G. Casey and G. Nagy, "Recursive segmentation and classification of composite character patterns," *Proc. Sixth Int'l Conf. Pattern Recognition*, pp. 1,023-1,026, Munich, 1982.

[3] S. Edelman, T. Flash, and S. Ullman, "Reading cursive handwriting by alignment of letter prototypes," *Int'l J. Computer Vision*, vol. 5, no. 3, pp. 303-331, 1990.

[4] E. Edelsbruner, D.G. Kirkpatrick, and R. Siegel, "On the shape of a set of points in the plane," *IEEE Trans. Information Theory*, vol. 29, no. 4, pp. 551-559, 1983.

[5] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation methods for character recognition: From segmentation to document structure analysis," *IEEE Proc.*, pp. 1,079-1,092, July 1992.

[6] M. Gilloux, "Hidden Markov models in handwriting recognition," S. Impedovo, ed., *Fundamentals of Handwriting Recognition*, vol. 124 of *NATO ASI Series F*, pp. 264-288. Springer-Verlag, 1994.

[7] V. Govindaraju and S.N. Srihari, "Separating handwritten text from interfering strokes," personal communication, 1992.

[8] J. Hull, S. Srihari, and R. Choudhari, "An integrated algorithm for text recognition: Comparison with a cascaded algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, no. 4, pp. 384-395, 1983.

[9] J. Hu, B. Sakoda, and T. Pavlidis, "Interactive road finding for aerial images," *IEEE Workshop Applications of Computer Vision*, pp. 56-63, Palm Springs, Calif., Nov. 1992.

[10] D.B. Kirkpatrick and J.D. Radke, "A framework for computational geometry," G.T. Toussaint, ed., *Computational Geometry*, pp. 217-248. North Holland, 1985.

[11] E. Lecolinet and J. Crettez, "A grapheme-based segmentation technique for cursive script recognition," *First Int'l Conf. Document Analysis and Recognition*, pp. 740-748, Saint-Marlo, France, Sept. 1991.

[12] D. Lowe, *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.

[13] K. Ohmori, "On-line handwritten kanji character recognition using hypothesis generation in the space of hierarchical knowledge," *Proc. Third Int'l Workshop Frontiers in Handwriting Recognition*, pp. 242-251, Buffalo, N.Y., May 1993.

[14] M. Okamoto and B. Miao, "Recognition of mathematical expressions by using the layout structures of symbols," *First Int'l Conf. Document Analysis and Recognition*, pp. 242-250, Saint-Marlo, France, Sept. 1991.

[15] J.D. Radke, "On the shape of a set of points," G.T. Toussaint, ed., *Computational Morphology*, pp. 105-136. North-Holland, 1988.

[16] J. Rocha and T. Pavlidis, "A shape analysis model with applications to a character recognition system, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 4, pp. 393-404, Apr. 1994.

[17] J. Schümann, N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberländer, "Document analysis—from pixels to contents," *IEEE Proc.*, vol. 80, pp. 1,101-1,119, July 1992.

[18] K.L. Supowit, "The relative neighborhood graph with an application to minimum spanning trees," *J. ACM*, vol. 30, pp. 428-448, 1983.

[19] W.J. Sakoda, J. Zhou, and T. Pavlidis, "Feature extraction directly from gray scale with applications to an address recognition system," *SPIE Symp. Electronic Imaging and Technology*, pp. 21-29, San Jose, Calif., Feb. 1993.

[20] S. Tsujimoto and H. Asada, "Resolving ambiguity in segmenting touching characters," *First Int'l Conf. Document Analysis and Recognition*, pp. 701-709, Saint-Marlo, France, Sept. 1991.

[21] G.T. Toussaint, "The relative neighborhood graph of a finite planar set," *Pattern Recognition*, vol. 12, pp. 261-268, 1980.

[22] G.T. Toussaint, "A graph-theoretical primal sketch," G.T. Toussaint, ed., *Computational Morphology*, pp. 229-260. North-Holland, 1988.

[23] J.I. Toriwaki and S. Yokoi, "Voronoi and related neighbors on digitized two dimensional space with application to texture analysis," G.T. Toussaint, ed., *Computational Morphology*, pp. 207-228, North-Holland, 1988.