Recognition of Handwritten Characters by Voronoi Representations

Niranjan Mayya
Andrew F. Laine

UF-CIS-TR-94-18

Department of Computer & Information Sciences
University of Florida, Gainesville, Fl 32611

Submitted to IEEE PAMI

Submitted for publication to the
IEEE Transactions on Pattern Analysis and Machine Intelligence
as a Regular Paper.

Recognition of Handwritten Characters by Voronoi Representations

Niranjan Mayya and Andrew F. Laine[1]
Department of Computer & Information Sciences
University of Florida,
Gainesville, FL 32611

Abstract

We present a new skeletonization algorithm well suited for the problem of handprinted character recognition. Our approach employs a novel algorithm for computing the Voronoi diagram of a polygon with holes. We show that Voronoi skeletons can serve as efficient shape descriptors because they preserve connectivity and Euclidean metrics. Compared to traditional skeletonization techniques, we suggest that shape representations based on Voronoi skeletons may increase the reliability of production quality character recognition systems.

A feasibility study is described in which more than 10,000 handprinted characters were recognized with an error rate of 2.34% by a neural network trained using Voronoi skeletons of character shapes from a class of 52 distinct alphanumeric patterns and graphical symbols. These results show that feature vectors extracted from Voronoi skeletons provide for high reliability in handprinted character recognition at a reduced cost of representation.

KEYWORDS: Character Recognition, Voronoi Diagrams, Skeletons, Medial Axis Transform.

[1] *Please address all correspondence to:*
Dr. Andrew F. Laine
Computer and Information Sciences Department
CSE Building, Room 301
P.O¿ Box 116120
University of Florida
Gainesville, FL 32611-2024

E-mail: laine@cis.ufl.edu
Phone/Fax: (904) 392-1239

1 Introduction

Handwritten character recognition is an important subproblem within the broader field of Optical Character Recognition (OCR). While many researchers have successfully provided partial solutions to this difficult problem, the problem of character recognition in general and handwritten character recognition in particular, remains unsolved. Specifically, more reliable methods need to be found before human performance in recognizing characters can be matched. A survey article by Mori et. al [20] documents the state-of-the-art in OCR, and papers by Suen et. al [30, 31] describe more recent advances made in handwriting recognition.

The steps involved in most character recognition systems can be broadly classified into preprocessing, feature extraction and classification. Of these steps, methods of feature extraction have received the largest amount of research and development. There are two parts to this step: (a) defining distinctive features of characters, (b) extracting features once they have been defined. Part (a) of this problem remains open. That existing character recognition systems have yet to match human performance, can in part be ascribed to the intractable nature of defining "distinctive characteristics" for character patterns. In [31], a broad taxonomy of feature extraction approaches are discussed. These include *global analysis* (techniques such as template matching, measurement densities of points, moments and mathematical transforms) and *structural analysis* (methods that extract loops, endpoints, junctions and arcs from the contour or skeleton of a character). The latter class is concerned with capturing the essential shape of each character. In the next subsection, we shall look at skeletons and skeletonization techniques in greater detail.

1.1 Skeletons (Symmetric Axis Transform)

The skeleton of a planar object, also known as the Symmetric Axis Transform (SAT) or the Medial Axis Transform (MAT) is a well known tool in shape modeling. Pavlidis [26] provides a formal definition: "Let R be a plane set, B its boundary, and P a point in R. A nearest neighbor of P on B is a point M in B such that there is no other point in B whose distance from P is less than the distance PM. If P has more than one nearest neighbor, then P is said to be a *skeletal point* of R.

The union of all skeletal points is called the skeleton or medial axis of R." An analogous definition is given by Blum using the well-known prairie fire analogy [3]; if one applies fire to all the sides of P, and lets the fire propagate at constant speed, then the skeleton is the locus of points where (fire) wave fronts meet.

The skeleton of a planar object relates the internal structure of the object to significant boundary features. It is a compact descriptor for the "natural" shape of an object, that well describes its global *topological and geometric* properties. Historically, skeletons have been previously used in higher level computer vision tasks such as object/character recognition, as they provide a more explicit, compact and stable representation, compared to an original intensity map of an image. In the context of character recognition, skeletons have an added advantage in that they provide a thin line representation similar to human handwriting. This allows for a more intuitive design of recognition algorithms [31]. Another important feature exhibited by skeletons is that their compact thin line representation is invariant to minor distortions, which allows for some measure of invariance to differences in individual writing style. Finally, skeletons make possible simple extraction of critical points such as end points and junctions which are cornerstone features for syntactic recognition.

However, skeletons have some disadvantages. The first among these, is that the thinning process may discard some amount of useful detail, that would be retained in other representations, such as contours. A good example of this is a filled in hole or concavity. This is an important problem in the context of character recognition, and an easy solution is not immediately apparent. Contours and skeletons can complement each other as feature descriptors for character recognition. A particularly good discussion of the relative merits and demerits of using skeletons versus contours in character recognition is given in [30] and [31], where a system is built that combines the powers of both representations. Another important disadvantage of skeletons is that some skeletonization algorithms introduce distortions of their own (sometimes known as noise spurs.). In later sections, we look in greater detail at how researchers across the field have handled this problem.

In the next sub section we describe previous skeletonization algorithms and introduce our approach to the skeletonization problem.

1.2 Algorithms for Skeletonization

Most existing implementations for computing skeletons use discrete space concepts that only approximate Blum's definition. In particular, preserving important properties such as *connectivity* and *Euclidean metrics* have been difficult to achieve in the discrete world. Previous algorithms (and implementations) that have been developed can be classified [13, 23] into three distinct groups: (1) Topological thinning, (2) Medial Axis extraction from a distance map, and (3) Analytic computation of a skeleton based on an approximation of the object contour.

Topological Thinning. A large class of these algorithms examine the topological relevance of object pixels rather than the metric properties of a shape. Typically, object pixels are repetitively tested and subsequently deleted, whenever their removal does not alter the topology of a thinned shape. Most character recognition systems that utilize skeletons, employ thinning techniques to compute them. The advantage of this approach is that connected skeletons can be ensured by relatively fast algorithms. However a significant disadvantage is that their discrete domain gives rise to *non-Euclidean metrics.* Different thinning algorithms applied on the same image can result in skeletons that may vary. These problems are in part due to different pixel "removal conditions" that are defined in terms of local configurations.

Medial Axis extraction from a distance map. The second method of skeleton extraction requires the computation of a distance map; determine for each point inside an object, the distance of the closest point from its boundary. Depending on the metric used for distance, a wide array of possible distance maps can be obtained. Unfortunately, the easiest and simplest algorithms are those based on non-Euclidean metrics, which lead to skeletons that are not accurate in terms of the fire front paradigm described earlier. However, algorithms that compute correct Euclidean distance maps do exist [4]. Skeletonization algorithms using quasi-Euclidean and Euclidean maps follow ridges in the distance map to construct a skeleton. *However, the problem with these methods are that connectivity is not guaranteed [13].*

Analytic computation of symmetric axes. The third method involves computation of symmetric axes by a direct analytical method based on polygonal approximation of a shape. Early work using this approach was by Montanari [21] who solved a system of linear equations to compute loci of equidistant points.

The Voronoi diagram [28] is a useful geometric structure which contains complete planar proximity information for a set of points [1]. This allows for simplified computation of a distance map directly from the Voronoi diagram. Furthermore, the Voronoi diagram of boundary line segments of a polygon is closely associated to its medial axis. In fact, the *medial axis* is exactly contained in the set of Voronoi edges of the polygon, and can be obtained simply by deleting the two Voronoi edges incident with each concave vertex [12]. Thus construction of the Voronoi diagram is clearly a technique for the skeletonization of polygonal shapes.

1.3 Skeletons Derived from Voronoi Diagrams

Using the Voronoi diagram to compute the skeleton of a polygonal shape is attractive because it results in skeletons which are connected while retaining Euclidean metrics. Furthermore, we obtain an *exact* medial axis, compared to an approximation. Thus we may reconstruct exactly an original polygon from its skeleton, (invertibility or one-to-one mapping). Finally, algorithms to compute the Voronoi diagram (and hence the skeleton) are much faster than methods that compute a distance map.

However, there are some disadvantages of using Voronoi diagrams to derive skeletons. We suggest that any method that utilizes Voronoi diagrams of polygons to compute skeletons must somehow overcome the disadvantages listed below, before it can be of practical value.

i) Natural shapes are non-polygonal. Thus, accurate polygonal approximations of such shapes are required in order to compute skeletons without loss of accuracy.

ii) The skeleton of a many sided polygon (of very short sides) will have a large number of redundant edges because of the Voronoi edges at these vertices. This results in increased complexity of the skeleton, without the addition of any shape information.

iii) Finally, robust and practical algorithms (affording ease of implementation) for Voronoi diagram construction of polygons are uncommon. Most existing algorithms make assumptions about cocircularity of no more than three points, and colinearity of no more than two. These constraints are difficult to satisfy in most practical applications.

In the next subsection we describe the basis of our approach to the skeletonization problem and show how we overcome the disadvantages of Voronoi diagrams described in this section.

1.4 Voronoi Skeletons for Character Recognition

We now describe a skeletonization algorithm that is well suited for the character recognition problem. The basis of our approach is the application of a algorithm for computing the Voronoi diagram of a polygon [16]. Our skeletonization algorithm retains the advantages of Voronoi diagrams described in the previous section, (connectivity, Euclidean metrics and high accuracy). This approach is thus a marked improvement over traditional skeletonization methods previously used in character recognition studies. Furthermore we overcome the disadvantages of Voronoi skeletons, identified in the previous section:

i) Character shapes are not extremely complex; hence accurate polygonization is realistic.

ii) We use a pruning approach to delete redundant skeleton edges that is guaranteed to retain connectivity. The pruning step is a simple consequence of the Voronoi diagram algorithm and does not require postprocessing. This also ensures that our method is stable with respect to being invariant to perturbations along a boundary. This allows for a certain degree of invariance to differences in handwriting style.

iii) Our Voronoi diagram algorithm is simple to implement. Assumptions about points being in general position are unnecessary. Unlike most geometric algorithms, special cases such as cocircularity of greater than three points or colinearity of more than two points are handled elegantly [16]. In addition, our algorithm features a robust numerical scheme to compute non-linear parabolic edges that avoids having to solve equations of degree greater than two.

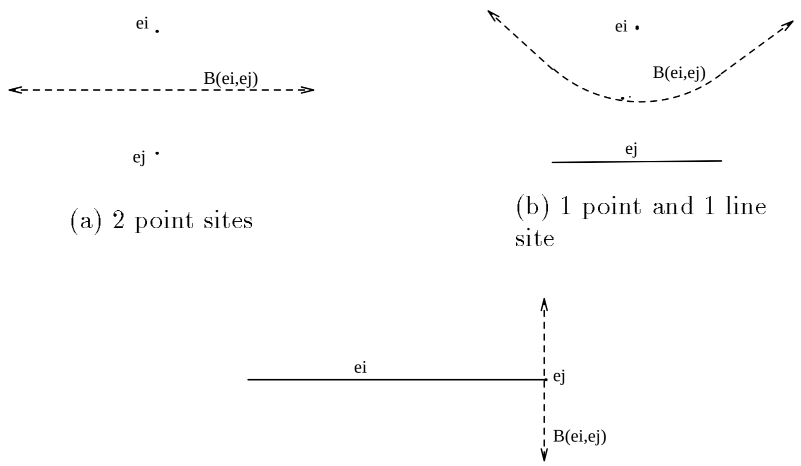
We present experimental results, in which over ten thousand handprinted characters were recognized with a 2.34% error rate by a neural net classifier [7] trained with Voronoi skeletons of character shapes, from a class of 52 distinct alphanumeric and graphical patterns. These results represent an improvement over earlier work by one of the authors of this paper [8, 9].

In the remaining sections we present the details of our approach. The rest of this paper is organized as follows. Section 2 contains a brief description of our Voronoi diagram algorithm. In Section 3 we explain how the Voronoi skeletons of character shapes are derived from their Voronoi diagrams. Finally, we discuss performance and recognition results in Section 4 and summarize our study in Section 5.

2 The Voronoi Diagram

In this section, we provide a brief description of our Voronoi diagram algorithm. For greater details of the algorithm and the proofs the reader is directed to [16].

The Voronoi diagram of a set of sites in two dimensions [28] is the partition of the plane into regions; each region i containing the set of points in the plane closest to the site i. In the most common case which has been exhaustively researched [28] in the past decade, the sites under consideration are points in the plane. In this case, edges of the diagram are straight line segments that are perpendicular bisectors of pairs of sites. Optimal algorithms as well as robust implementations have been devised for this version of the problem. The notion of a site has been generalized to include a collection of two-dimensional objects such as line segments and circular arcs. In the case of line segments, the edges of the ensuing Voronoi diagram are not just straight-line segments, but also arcs of parabola, since they separate the loci of proximity of object pairs of the types $(point, point)$, $(line, line)$, $(point, line)$; the last of these pairs give rise to arcs of parabola (See Figure 1). The Voronoi diagram of the interior of a polygon is of interest to researchers across different fields. In this case the set of sites include the edges and vertices of the polygon. It has been used [25] to generate meshes for multiply-connected polygonal domains and has been long known to provide the medial axis of polygonal shapes [12] that serve as a basis for efficient shape



(a) 2 point sites

(b) 1 point and 1 line site

(c) the point site is an end point of the line site

Figure 1: Bisectors of e_i and e_j

representation schemes [17]. While algorithms that compute the Voronoi diagram of the interior of a polygon exist [24, 19], easy to implement algorithms that are robust and make no assumptions about the position of the sites in a plane have not been proposed thus far. We believe that our algorithm fills this void. (See [16] for a discussion of previous work on Voronoi diagrams.)

2.1 Preliminaries

In this section, we will introduce some definitions and notation. Definitions 1 through 8 pertain to the definition of polygonal domains and Voronoi diagrams and are taken from [24].

Definition 1 *A closed line segment* $[a, b]$ *is the union of two endpoints* a *and* b *and the open line segment* (a, b).

Definition 2 *A multiply-connected polygonal domain* P *is the closure of a nonempty, bounded, connected, open (in the relative topology) subset of* $\mathcal{R}^2$ *whose boundary is the union of a finite number of closed line segments.*

The boundary of P, denoted by δP, consists of one or more disjoint subsets. The outer boundary of the polygonal domain is denoted by δP_0 and contains P. The inner boundaries represent the holes of the polygonal region and are denoted by δP_i, $1 \leq i \leq H$, where H denotes the number of holes.

Definition 3 *The vertices of δP are the points of intersection of the closed line segments which constitute δP. The edges of δP are the open line segments obtained by deleting the endpoints of the closed line segments which constitute δP*

Definition 4 *The projection $p(q, [a, b])$ of a point q onto a closed segment $[a, b]$ is the intersection of the line through a and b and the line perpendicular to $[a, b]$ and passing through q.*

Definition 5 *The bisector $B(e_i, e_j)$ of two sites e_i, and e_j, is the locus of points equidistant from e_i and e_j.*

Definition 6 *The half-plane $h(e_i, e_j)$ is the set of points closer to site e_i than to site e_j. Its complement, $\bar{h}(e_i, e_j)$, is the set of points not closer to site e_i than to e_j.*

The nature of the bisector is determined by the nature of the sites (point, line). In particular, when both e_i and e_j are point sites, the bisector is a straight line (the perpendicular bisector of e_i and e_j. When one of the sites is a point, and the other is an open line segment the bisector is in general a parabolic arc. For the special case where the point is one of the endpoints of the open line segment, the bisector is a straight line passing through the point and perpendicular to the open line segment. See Figure 1.

Definition 7 *Given a set of sites S, and a site e_i, $e_i \notin S$, the Voronoi region of e_i with respect to S, denoted by $V(e_i, S)$ is the set of all points closer to e_i than to any site in S.*

Lemma 1 $V(e_i, S) = \cap_{e_j \in S} h(e_i, e_j)$

Proof: Corollary 2, Lemma 1 of [24] ■

Definition 8 *The Voronoi diagram, VOD(S), of a set of elements, $S = e_i$ is given by* $\cup_{e_i \in S}\ V(e_i, S - e_i)$

2.2 Primitives used in the Algorithm

A site e_i can be either a point or an open line segment. Each such site is associated with the following information.

1. **A contact point** x_{p_i}. For a point site, the contact point is the site itself; for a line site, the contact point is one of the endpoints.

2. **A distance function** $d_i(x_0)$; returning the square of the distance of a point x_0 from site e_i. For a line site, the distance of a point to the site is defined as the distance from the point x_0 to its projection on the line site e_i. Given this definition, we can define the distance of a point to any site (either a point or a line) as follows, Given a line site, let $\vec{n_l}$ be its unit vector. If I is the identity matrix, and T denotes the transpose of a matrix, we define a 2×2 matrix M as

$$\begin{aligned} M &= I && \text{for a point site,} \\ &= I - \vec{n_l}\,\vec{n_l}^T && \text{for a line site} \end{aligned}$$

   Now we define the distance function as,

$$d_i(x_0) = (x_0 - x_{p_i})^T \; M \; (x_0 - x_{p_i})$$

3. **A gradient vector** $g(x_0)$; which is defined by.

$$g_i(x_0) = 2(x_0 - x_{p_i})^T \; M$$

2.3 Algorithm

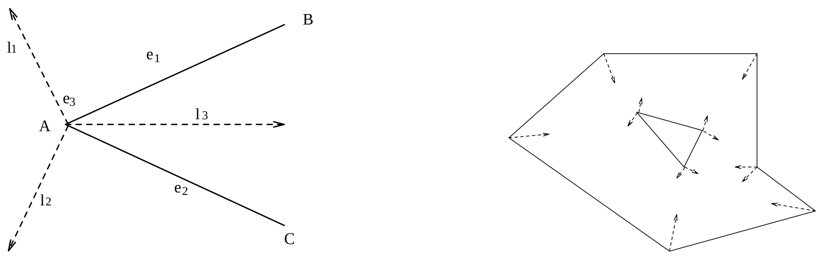
2.3.1 Overview

The Voronoi diagram gives us a complete description of the function $t(x)$, that returns the distance of the point x to the closest site in the set S. In particular,

- A Voronoi region (face) is characterized by a single site e_i; the function is given by $t_i(x)$.

- A Voronoi edge is characterized by two sites, e_i and e_j; the edge comprises the set of points where $t_i(x) = t_j(x)$.
- A Voronoi vertex is characterized by 3 or more sites, i, j, k,...m; the vertex satisfies the locus $t_i(x) = t_j(x) = t_k(x) = ...t_m(x)$.

Given an initial Voronoi vertex and the initial direction of the Voronoi edge emanating out of that vertex, we follow the path traced by this edge to determine the vertex at the other end. Every other site is examined to find the closest site that determines the new vertex. The new vertex is equidistant from three or more sites, every pair of which gives rise to a possible new Voronoi edge. The new Voronoi edges are added to an *unexamined Edge List*. The program terminates when all the edges have been traced.



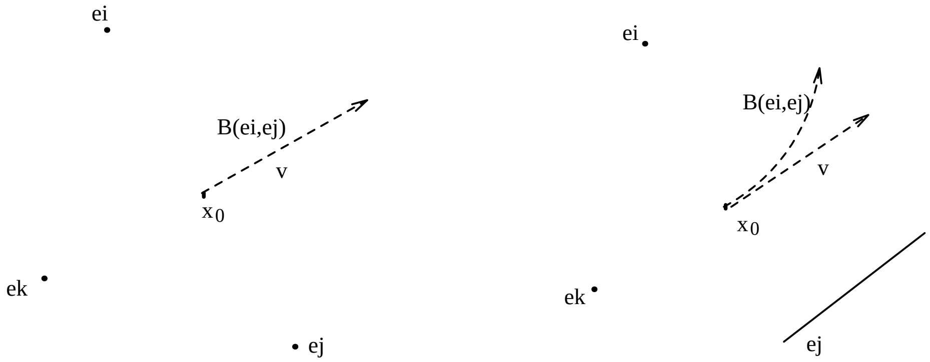
(a) Voronoi edges of 2 sides of a polygon

(b) Initialization for a polygon with holes

Figure 2: Initialization

2.3.2 Initialization

Given a pair of successive segments of any polygonal region, this gives rise to the following 3 Voronoi edges. See Figure 2a. BA and AC are 2 successive sides of a polygon. There are 3 sites corresponding to these 2 sides; two line sites e_1 and e_2 corresponding to the open line segments BA and AC, and the point site e_3. The 3 Voronoi edges corresponding to these sites are shown by the dashed lines. l_1 is the bisector of e_2 and e_3, l_2 that of e_1 and e_3, and l_3 the e_1, e_2 bisector. If we are considering the Voronoi diagram of the interior of the polygonal region alone, we are only concerned with those Voronoi edges inside the region. These are easily obtained by considering only the (line,line) bisectors (the l_3 type edges) of every convex pair of successive polygon segments. for the outer boundary of the polygon, and the (point,line) edges (the l_1 and l_2 type edges) for every concave pair of successive polygon segments. For the inner boundaries (the holes), we perform the reverse. Figure 2b shows the complete initialization for a simple case. The vertices of the polygon are also Voronoi vertices. Each of the edges determined in this step are added to the unexamined Edge List.



(a) $B(e_i, e_j)$ is along the direction of $\vec{v}$

(b) $\vec{v}$ is tangent to $B(e_i, e_j)$ at x_0

Figure 3: Initial Direction for a Voronoi edge

2.3.3 Curve Tracing

The unexamined Edge List holds edges which have been only partly determined. Specifically, an unexamined Voronoi edge E is a bisector of two sites e_i and e_j, containing the starting point x_0, and the initial direction $\vec{v}$ along which we must traverse to determine the other points along the edge. The initial direction of the bisector is determined upto the linear order and is given by $\vec{v}$ $= (g_i(x_0) - g_j(x_0))^{\perp}$, where $a^{\perp}$ denotes a unit vector perpendicular to the vector a. As has been noted earlier, if the Voronoi edge is a $(point, point)$ or $(line, line)$ bisector, it will be a straight line, in which case there is only a single terminating Voronoi vertex to be determined. If we have a $(point, line)$ bisector, the Voronoi edge is a parabolic curve; this curve has to be traced and intermediate points along it computed to fully determine the Voronoi edge.

In the linear case, the bisector $B(e_i, e_j)$ of sites e_i and e_j (see Figure 3), lies along the direction

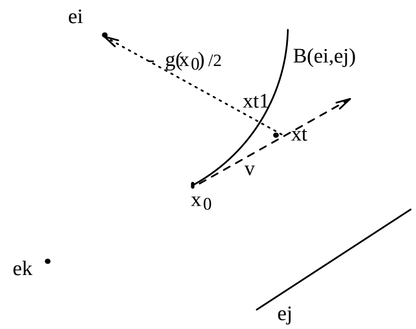


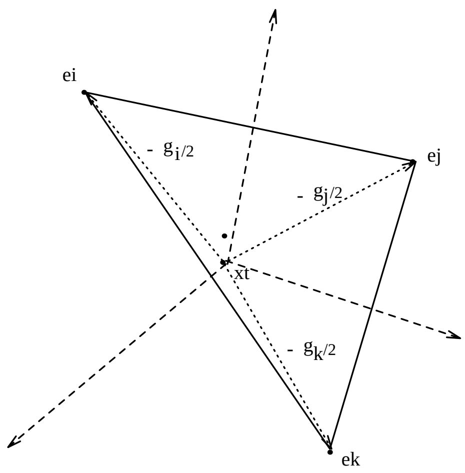
Figure 4: Finding an intermediate point on the curve

of $\vec{v}$. In this case, the bisector is fully determined by computing the Voronoi vertex x_t at the other end. Finding the terminating vertex on the bisector involves determining the site e_l, $(l \notin i, j, k)$, that is closest to the sites e_i and e_j. The algorithm examines every site e_l $(l \notin i, j, k)$, and computes the point x_{tl} that is the center of the circle passing through e_i, e_j and e_l. Each such point x_{tl} will lie on the bisector $B(e_i, e_j)$ along the direction $\vec{v}$; the point closest to the starting vertex x_0 is the new Voronoi vertex x_t.

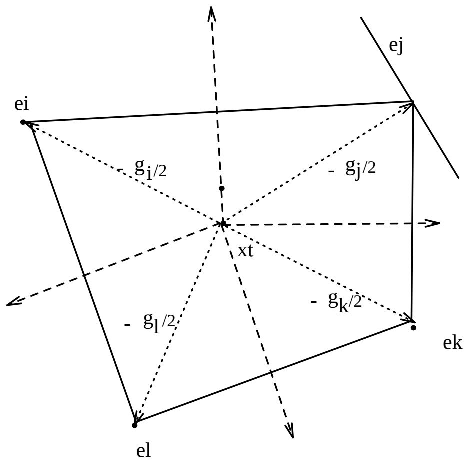
The situation in the non-linear case is slightly more complicated. In this case, the bisector $B(e_i, e_j)$ is a parabolic curve, and the initial direction $\vec{v}$ is a tangent to the bisector at the starting point x_0, (Figure 3b) and is the linear approximation of the bisector.

In order to determine the Voronoi vertex at the other end of a parabolic curve, an iterative technique is employed. First, a similar procedure as in the linear case is followed to determine a point x_t on the linear approximation of the bisector. Since x_t does not lie on the bisector itself, we need to move back to a point on the curve itself. It can be seen that we will intersect the bisector $B(e_i, e_j)$ if we move along the contact vector $-g(x_t)$ towards the point site. The point of intersection x_{t1} is an intermediate point along $B(e_i, e_j)$. See Figure 4.

It is easy to prove [16] that this procedure is guaranteed not to overshoot the Voronoi vertex; namely, the open interval of the bisector $B(e_i, e_j)$ between the points x_0 and x_{t1} contains no Voronoi vertex. Having determined an intermediate point on the bisector, the procedure is repeated using a new value for $\vec{v}$ that is given by $\vec{v} = (g_i(x_{t1}) - g_j(x_{t1}))^{\perp}$. The procedure terminates when we arrive at a point x_{tn} which corresponds to a site e_n, such that x_{tn} is equidistant from e_i, e_j and e_n. x_{tn} is the new Voronoi vertex.



(a) 3 co-circular point sites



(b) 4 co-circular sites with 1 line site

Figure 5: Determining edges from the convex hull of points of contact

We note that the rate of convergence of the iteration is the same as that of Newton's method, since we are approximating an arc by a straight line.

2.3.4 Adding New Edges

Having computed a new Voronoi vertex x_t, we must first perform a check to determine if this vertex has been found before. If this vertex has been computed earlier, it means that there must exist in the unexamined Edge List, an unexamined edge with starting point x_t. This edge must be deleted from the Edge List, since it has now been fully determined. If the new vertex has not been computed before, we proceed to determine all the edges coming out of this vertex.

We first need to determine all the other sites equidistant from this vertex, and then determine the new Voronoi edges that emanate out of the new vertex.

The first step is easy to perform; all sites e_l, which satisfy

$$d_l(x_t) = d_i(x_t) = d_j(x_t)$$

lie on the circle of radius $\sqrt{d_l(x_t)}$, centered at the new Voronoi vertex x_t.

The second step is to determine all the edges coming out of this vertex. In general, if we have k

sites equidistant from a vertex, there can be $\binom{k}{2}$ pairs of possible edges but all of these will not be Voronoi edges. From the vector $g(x_t)$, for each site e_i, we can determine the point of contact of $-g_i/2$ with e_i. If we denote the points of contact as y_i, the convex hull of all the y_i determines which pairs of sites correspond to Voronoi edge. Every convex hull edge gives a pair of sites that correspond to a Voronoi edge. Thus if we have k equidistant sites from the vertex x_t, there are exactly k Voronoi edges. See Figure 5.

One of these k edges is the edge which was just traced. The other $k-1$ edges are appended to the unexamined Edge List. The head of the Edge List is then examined and the curve tracing procedure is repeated until the Edge List is empty.

Due to the incremental nature of the edge tracing algorithm, we do not need to make any assumptions about points in general position. Hence co-circularity of more than 3 sites can be handled easily.

2.3.5 Time Complexity

Let the number of sites in the input be n_p, the number of Voronoi edges be n_e and the number of Voronoi vertices be n_v.

The initialization step is of the order of the number of sites in the input, since a single pass over the input suffices to create the initial edges and append them to the unexamined Edge List. This step takes $O(n_p)$ time.

Each edge that is added to the diagram in the edge tracing step requires us to examine each site to determine the closest site. Also, we need to search the list of current Voronoi vertices to check for the existence of a new vertex. Hence, we need $O(n_p + n_v)$ time for each new edge found. The total time required for the algorithm is therefore $O(n_p n_e + n_v n_e)$. By Euler's formula the number of Voronoi edges is $2n_p - 3$ and the number of Voronoi vertices $n_p - 2$. Hence the time complexity is $O(n_p^2)$.

3 Skeletonization using Voronoi Diagrams

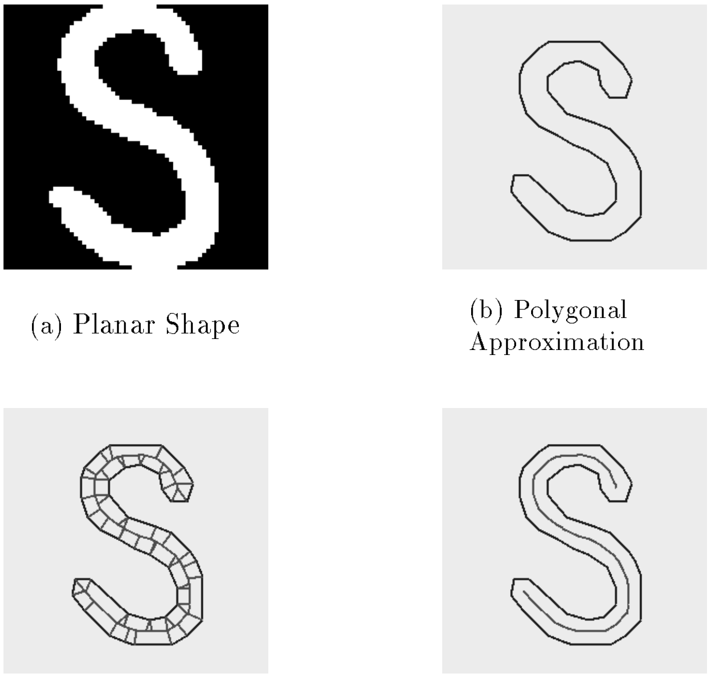
In this section, we describe our technique for obtaining skeletons of character patterns that are derived from Voronoi diagrams.

3.1 Preprocessing: Computing a Polygonal Approximation

In order to compute the Voronoi diagram of a shape, we first segment the shape's boundary and derive a polygonal approximation of its bounding contour. In our implementation, we used the zero crossings obtained from an image filtered with a Laplacian of a Gaussian [14], to obtain a bounding contour. The advantage of this technique is that it guarantees *closed contours*, which ensures that a contour tracing algorithm such as chain coding will converge quickly. After computing a chain coded bounding contour, critical points (those that reflect significant change in curvature) are retained; these serve as the vertices of polygonal approximation for each (planar) shape. The input shapes were approximated to ensure that significant changes in curvature are retained, while minor distortions that could result in noise spurs in the skeleton representation are smoothed over. This limited the occurrence of redundant edges in each representation.

3.2 Computing the Voronoi Diagram and Extracting Skeletons

Given the vertices for a polygonal approximation as determined above, we computed the Voronoi diagram using the algorithm described earlier in Section 2. The medial axis [12] of a polygonal shape can be derived from its Voronoi diagram by deleting those edges that arise from concave vertices of the polygon (see Section 2.) The data structure created to store Voronoi edges contained the information required to identify these edges. Hence it was easy to identify (and delete) edges that emerged from concave vertices of a polygon. However, resultant medial axes are characterized by a very large number of edges, as every vertex of the polygonal approximation (both concave and convex) gave rise to a Voronoi edge. Many of these Voronoi edges were redundant; i.e. they did not provide any additional structural information about the planar shape. Hence we designed a pruning technique to delete redundant edges.



(a) Planar Shape

(b) Polygonal Approximation

(c) Voronoi Diagram

(d) Pruned Skeleton

Figure 6: Voronoi Skeletonization of Planar Shapes.

3.3 The Pruning Operator

As shown in Figure 6c, the vertices of the initial polygonal approximation of a shape lead to a large number of edges that do not contribute to overall shape information. In a Voronoi diagram every Voronoi edge is a bisector of two sites on the boundary of a polygon; in particular the Voronoi edges at the vertices of a polygon are bisectors of adjacent sites. If the sides of a polygon are numbered in counterclockwise order along the boundary, we can define the *adjacency* of a Voronoi edge as follows. Let E be a Voronoi edge that is a bisector of two sites numbered i and j. Then $Adjacency(E) = |i - j|$. We observe that Voronoi edges that lie deep inside an object have higher adjacency than those on the perimeter. Furthermore segments that describe "global" topological (symmetry) relations are bisectors of high adjacency. This fact gives us a means of filtering out unimportant segments. We simply discard edges that are of adjacency lower than some preset threshold. Since the adjacency information is implicitly contained in our edge data structure (see Section 2), no additional post-processing is required to obtain this reduced efficient set of edges.

We note that the authors of [23] use a similar (but not identical) technique for pruning the edges of Voronoi diagrams. The problem in their case was more critical, since they started out with a set of raster crack end-points along the boundary of an object and hence had an extremely large number of Voronoi edges to process.

Figure 6 shows the result of the three step procedure described above on a planar shape of the character 'S'. Figure 7 presents additional examples of skeletons derived using our skeletonization technique.

3.4 Preserving Connectivity

As described in Section 1, one of the advantages of skeletons derived from Voronoi diagrams is that we are guaranteed connected skeletons. However, what is the effect of the pruning step on connectivity? Furthermore, how does one determine an optimal threshold for deleting the maximum number of redundant skeletal edges without losing connectivity? Unfortunately, it is not possible to determine a threshold value a priori and apply it to a large class of objects. However, by constraining the threshold value to 1 (retaining only those edges that are of adjacency greater than 1), we can prove that the resultant skeleton remains connected.

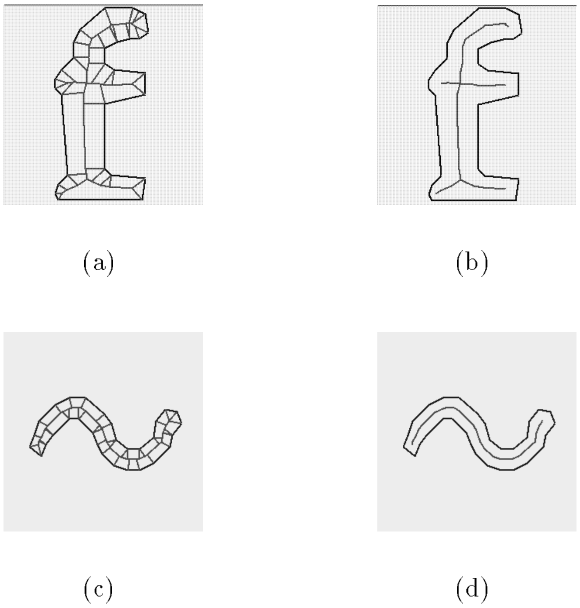


Figure 7: (a),(c) Sample Voronoi Diagrams, (b), (d) Corresponding pruned skeletons

Lemma 2 *Pruning Voronoi edges of adjacency 1, is guaranteed to preserve connectivity.*

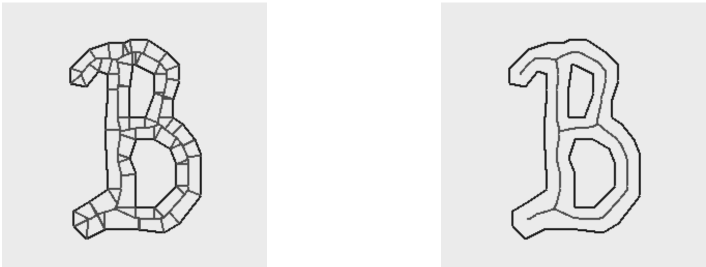
Proof: The Voronoi diagram of a polygon is a Planar Straight Line Graph [24], which by definition, is connected. For the removal of an edge to result in a disconnected graph, the edge must be a "bridge"; namely the edge must be part of every path between any two vertices of the graph. It is easy to see that no edge of adjacency 1 can be a bridge, since (by definition of adjacency) every edge of adjacency 1 terminates at a vertex of a polygon. Hence removal of all edges of adjacency 1 is guaranteed to preserve connectivity of the resulting skeleton. ■

3.5 Skeletonizing Shapes with Holes

One of the attractive features of our method is that it handles shapes with and without holes in a uniform manner. Figure 8 shows an example of our approach applied on a holed object. No additional initialization is required. However, when carrying out the pruning step, care must be taken to define the adjacency information correctly, in order that the last numbered site on the outer boundary and the first numbered site on the next hole are not considered adjacent. Shapes with multiple holes are handled similarly.

(a) Planar Shape

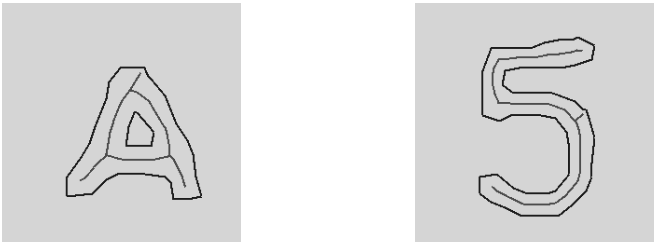
(b) Polygonal Approximation



(c) Voronoi Diagram

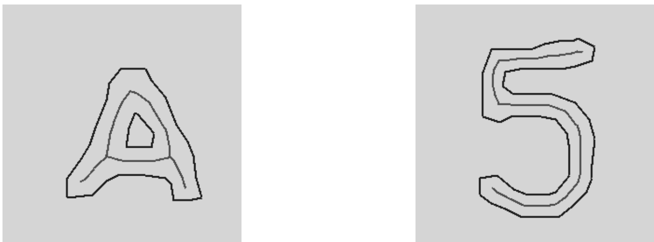
(d) Pruned Skeleton

Figure 8: Skeletons of Character shapes with holes



(a)

(b)



(c)

(d)

Figure 9: (a,b): Very small edges cause noise spurs. (c,d): Smoothing contour eliminates noise spurs.

	Time Complexity	Actual Running Time
Segmentation	$O(N \log N)$	2.26 sec [1]
Polygonization	$O(B)$	0.10 sec
Voronoi Skeleton	$O(B^2)$	0.81 sec

Table 1: Performance Evaluation: Running times reported per 64x64 character image.

3.6 Performance Evaluation

In this section, we briefly look at the time complexity of the entire skeletonization process. In the discussion that follows, let N denote the size of a square image, ($N = M * M$), where M is the length of each row and column. Let B represent the average size of the bounding contour of each planar shape. In Section 2, we saw that the time complexity of the Voronoi diagram algorithm is $O(B^2)$, where B is the number of sites on the bounding contour. Existing skeletonization algorithms have time complexities that are a function of N alone, and hence valid comparisons are difficult. In practice, we have observed that in spite of the quadratic complexity, the algorithm is efficient in practice, since the number of boundary points B, is generally an order of magnitude less than N. For completeness, however, the preprocessing steps must also be taken into account. The table below summarizes both the theoretical time complexity of each stage, as well as the actual execution time taken on a Sparc 10 computer to compute the skeleton of each character shape. The timing measurements are reported for each character of size 64x64, averaged over a total of over 1,350 distinct character samples. Note that the Voronoi construction processing time accounts for only 25% of the total time required for skeletonization.

In the next section we show that Voronoi skeletons computed by the procedure described above, provide a more efficient representation compared to existing skeletonization algorithms.

3.7 Comparison of Techniques

Our approach exploits the fact that the medial-axis of a polygonal shape is implicitly contained in its Voronoi diagram. This fact immediately ensures (a) *that the computed skeleton lies in* $\mathcal{R}^2$, *(Euclidean metrics) and* (b) *connectivity is guaranteed. We compute the exact medial axis as opposed to a discrete approximation.* In the following paragraphs we show that these properties provide a significant advantage over traditional skeletonization techniques.

Skeletons computed by thinning algorithms are constrained by 4 or 8 connectivity of a discrete grid in which the object shapes are embedded [5]. The typical drawback of such approaches is the loss of Euclidean metrics. Furthermore thinning algorithms also have to deal with redundant edges as a result of noise artifacts on the boundary of object shapes.

An alternative approach to skeletonization involves *ridge following* techniques based on distance maps computed from object shapes. The quality of the resulting skeletons is critically dependent on the metric used to derive the distance map. Non-Euclidean metrics (city-block distance, chessboard distance among others), lead to simple skeletonization algorithms, but can result in inaccuracy of upto 40% with respect to Euclidean distances [13]. While methods to compute Euclidean [4] or quasi-Euclidean [22] distance maps exist, these methods do not guarantee connectivity. Gaps occur due to the discrete domain on which pattern shapes are embedded, and are usually filled in by postprocessing steps.

More recently, new skeletonization algorithms [23], [13] have appeared in the literature that are marked improvements over the traditional techniques described above. In the sequel, we briefly describe these methods, and compare them with our approach.

In [23], the authors compute the Voronoi diagram of *the set of points along the boundary of an object shape.* Two points arise with respect to the general scope of this technique: (a) A very large number of points are required to correctly approximate object shapes; however this may not be efficient. Further the large number of points lead to an extremely large number of redundant Voronoi edges, necessitating the use of complex pruning techniques. (b) A more critical drawback of this approach is that it is not easily applicable to objects with holes. In particular, it is not possible to decide a priori, the number of sampled points on an object, necessary to ensure that the Voronoi edges between the inner and outer boundaries will be computed.

Our approach on the other hand, overcomes both these deficiencies. The number of redundant edges are small. Thus a simple pruning step, that involves no postprocessing suffices to eliminate almost all spurious edges. In addition, by controlling the polygonal approximation process to discard edges of very small length on a contour, we can eliminate the occurrence of redundant edges that might not be deleted by the pruning step. Figures 8a and 8b, show examples where the polygonal approximations of the character shapes contained very small length edges that resulted in redundant edges that were not deleted by the pruning step. By ensuring that the polygonal approximation will not contain very small length edges these spurs are eliminated in Figures 9c and d.

Furthermore, polygons with and without holes are handled in a uniform manner. Polygons are defined in terms of line segments, and are therefore well defined at every point $p(x, y) \in \mathcal{R}^2$ along the boundary. This overcomes the drawbacks of discretizing the bounding contour as in [23].

The authors of [13] employ the *snake model* (an active contour model), to compute skeletons. In their technique, initial control points are defined for the snake at curvature extrema of a shape's bounding contour. The *dynamic grassfire* is then simulated by snake propagation. This technique maintains correct Euclidean techniques by computing the Euclidean distance map, while the nature of the contour model ensures that connectivity is maintained. However, one limitation of this approach is the problem of computing the correct curvature in the discrete domain. Furthermore, a very ragged boundary implies a large number of curvature extrema leading to redundant skeleton edges; in this case other criterion need to be applied to reduce the number of initial control points. In addition, a special case arises when the bounding contour includes a circular arc whose center (a) lies within the object and (b) is an end-point of a skeleton branch. In such a case, additional control points need to be defined to ensure accurate skeleton computation. *In contrast, our technique offers a elegant and uniform approach to skeletonization that is independent of the shapes topology.*

An implicit advantage offered by our method, is that *a graph representation of the skeleton is immediately available.* Recall that the Voronoi diagram of a polygon is a planar straight line graph, and that by Lemma 2, the pruning step maintains connectivity. Skeletonization methods that compute the skeleton as a pixel map (as compared to an edge map representation provided by our method), require an intermediate vectorization step before a graph representation may be obtained. In the context of character recognition systems, our technique is amenable to the simple extraction of structural features such as end-points and junctions by a simple traversal of the graph comprising a skeleton.

In summary, we have shown that Voronoi skeletons are powerful shape descriptors which overcome several significant disadvantages of existing techniques. In the next section we present results of our character recognition system based on this novel representation.

4 Experimental Results

In this section, we describe experimental results for our method exercised on a large database of handprinted characters.

4.1 The Database of Sample Patterns

In order to both evaluate and compare the performance of our feature extraction and representation methods, we exercised a large database of handprinted characters previously studied [8]. The database consisted of 52 distinct patterns containing over 10,000 samples, collected from 17 different writers. Table 2 summarizes the distribution of patterns in our database.

The characters included were well defined. To avoid ambiguity, authors were asked to slash the numeral '0', to print '1' without serifs and to place horizontal bars on the letter 'I'. In addition, only upper-case alphabets were considered, and the numeral '4' was required to be closed. Sample characters are shown in Figure 10.

Figure 10: Samples patterns for '0', 'O', '1', 'I', '2', 'Z' and '4'.

Sample patterns were collected on forms and were digitized at 300dpi/8bit resolution. Each sample was then normalized in size to 64 x 64 before processing [8].

Patterns	Number of Distinct Patterns	Number of Samples		
		Training	Testing	Total
0-9	10	1,360	680	2,040
Graphics	16	2,176	1,088	3,264
A-Z	26	3,536	1,768	5,304
All Above	52	7,072	3,536	10,608

Table 2: Distribution for database of sample patterns.

4.2 A Neural Network Classifier

The neural network used in our recognition system required that every input pattern be of constant size. Recall that, the output of our Voronoi skeleton algorithm was defined by a set of connected edges. Since the number of edges will vary for distinct instances of characters, we needed to ensure that each input was represented by a uniform size. The edge set representing the skeleton of each input pattern was rendered onto a binary image of size 16 by 16. The binary skeletons consisting of 256 input nodes were then used to drive the neural network with distinct training and testing samples from the 52 alphanumeric patterns in our database.

4.3 Neural Network Topology

We employed a conjugate gradient method to train a neural network [7] containing at most three hidden nodes and 52 output nodes. The number of hidden nodes was determined experimentally by varying the number of hidden nodes and identifying the best results. The input patterns were preclassified according to their Euler number (the number of connected components minus the number of holes.) This topological sorting step was performed primarily to reduce training time. We found that training three smaller neural networks (for shapes with 0, 1 and 2 holes respectively), took considerably less time than training a single more complex network. We note that classifying characters according to the topology is justified on account of the constrained nature of our database and the quality of our characters (no gaps.) When dealing with databases of poorer quality, it may not be reasonable [27, 2] to attempt such preclassification without additional preprocessing.

Classification Cases	Number of Errors	Error Rate	Classification Rate
0-9	7	1.02%	98.98 %
A-Z	25	1.41%	98.59%
A-Z, Graphics, 0-9	83	2.34%	97.66%

Table 3: Performance evaluation for three classification cases: [0-9], [A-Z] and [A-Z, Graphics, 0-9] for Voronoi skeleton representation.

We trained the network with three subsets of the database: (a) Numeric characters 0-9, (b) Alphabetic characters A-Z and (c) the entire database of 52 patterns (A-Z, 0-9, and 16 graphic characters).

Table 3 summarizes our existing experimental results. As shown, results for recognizing numerals and characters were very reliable; correct recognition rates of 98.98% and 98.59% were observed respectively.

4.4 Comparison of Performance

In the context of handwritten character recognition, it is difficult to exactly compare distinct approaches. There are a large number of variables that can bias results. These include the database size, the method of partitioning training and testing data sets, the kinds of errors reported (substitution and rejection errors), the quality of characters in the database, the nature of the character set (constrained or unconstrained) and so on. However, it is certainly meaningful to compare methods that have been tested under similar (if not identical) testing conditions. In [8], an earlier study by Laine et al, a neural net classifier was trained on rectangular wavelet representations of character shapes. The database used in our present study was identical to the one used in [8], as was the type of classifier used. A comparison of our results is provided in Table 4 and shows that Voronoi skeletons provided more reliable classification. The percentage of correct

Performance Evaluation				
	Voronoi Skeletons		Rectangular Wavelets	
Classification Cases	Number of Errors	Error Rate	Number of Errors	Error Rate
0-9	7	1.02%	11	1.61%
A-Z	25	1.41%	26	1.47%
A-Z, Graphics, 0-9	83	2.34%	93	2.60%

Table 4: Performance evaluation for three classification cases: [0-9], [A-Z] and [A-Z, Graphics, 0-9] for Voronoi skeletons and rectangular wavelet representations.

classification is comparable to those obtained by some of the leading practitioners in this area. In [18], the authors extract structural features from the skeleton and uses 11 specialized modules to come to a decision on the identity of patterns. The authors of [10] also use primitives derived from skeletons for recognition. Both of the last two methods were tested on the same database (US ZIP code database of CENPARMI (Concordia University); the training and testing sets consisted of 4000 and 2000 numerals respectively. Correct classification rates of 86.05% [18] and 93.1% [10] were observed. In [31], a multi-expert system is described which combines the results of four techniques. These include the last two methods described above, a structural technique that extracts features from the contours of characters, and a statistical approach. The combined system (tested on the same database) reported a correct classification rate of 93.05%, but had a zero substitution rate to achieve 100% reliability. Srihari [29], presents several different approaches ranging from stroke based recognizers to structural contour based chain code classifiers. The classifiers were based on different methodologies: statistical, structural, and syntactic. Each approach was trained on a set of 18,468 digits and tested on a set of 2,711 digits with correct classification results ranging from 83.6% to 96.4%. The data was derived from handwritten addresses obtained from the US Postal Service. Le Cun et al. [11] achieved good results with a back propagation neural network using size-normalized images as direct input. With a training and testing set size of 7291 and 2007 handwritten digits respectively, the system achieved a correct classification result of 92%.

5 Conclusions and Summary

In this paper we have presented a new skeletonization algorithm derived from Voronoi diagrams. Our approach relies on a new method for constructing Voronoi diagrams of polygons that is robust, easy to implement and attractive for practical applications. We have shown that Voronoi skeletons are connected, accurate (characterized by Euclidean metrics), and are superior to skeletons obtained by existing thinning algorithms. Redundant edges were deleted in a pruning step, that was guaranteed to preserve connectivity of each skeleton. The technique implicitly provides a graph representation that enables the simple extraction of end points and junctions. Feature vectors consisting of Voronoi skeletons of character shapes were shown to be well suited for character recognition. Correct classification rates of 97.66%, 98.56% and 98.98% for the entire database, characters 'A'-'Z', and numeric characters '0'-'9' respectively, were obtained, when a neural network was trained with Voronoi skeleton representations of the character shapes. These results suggest that Voronoi skeletons can provide efficient shape descriptors for character recognition.

Our future research efforts will be directed towards building a better and more robust character recognition system. Specifically, we shall work with larger databases (such as the NIST database), and test our representation using unconstrained characters. For character recognition systems to be practically viable, it is essential that error rates be split into substitution errors and rejection errors [27]. Modifying our existing recognition system to incorporate such distinctions is another area that will receive our attention in the near future. Finally, our skeletonization algorithm makes it easy to extract structural information such as end-points and junctions that are used in many syntactic recognition schemes. Since the skeleton is output in the form of a planar straight line graph, a single graph traversal is sufficient to determine end points and junctions. We shall investigate whether syntactic character recognition schemes based on Voronoi skeletons would provide efficient representations of more complex patterns.

References

[1] F. Aurenhammer [1991], "Voronoi diagrams - a survey of a fundumental geometric data

structure," *ACM Computing Surveys*, Vol. 23, 345–405.

[2] H. S. Baird [1993], "Recognition technology frontiers", *Pattern Recognition Letters* Vol 14, 327–334.

[3] H. Blum, [1967] "A transformation for extracting new descriptors of shape," *Models for the Perception of Speech and Visual Form* (W. Wathen-Dunn, ed.), Cambridge MA: MIT Press.

[4] [1980] P. E. Danielsson, "Euclidean distance mapping," *Comput. Graphics Image Processing*, Vol 14, 227-248.

[5] [1981] E. R. Davies and A. P. N. Plummer, "Thinning algorithms: a critique and a new methodology," *Pattern Recognition,* Vol. 14, 53–63.

[6] R. C. Gonzalez and R. E. Woods[1992], "Digital Image Processing", Addison-Wesley.

[7] B. Kalman [1990], "Super linear learning in back propagation neural nets," Dept. of Computer Science Technical Report WUCS-90-21, Washinton University, St. Louis.

[8] A. Laine, S. Schuler, and V. Girish [1993], "Wavelet representations for recognizing complex annotations," *Machine Vision and Applications* Vol 6, 110–123.

[9] A. Laine, W. E. Ball and Arun Kumar [1991], "A Multi-Scale Approach for Recognizing Complex Annotations in Engineering Drawings," *Proceedings of the IEEE CVPR*, Lahaina, Hawaii.

[10] Lam, L., S. W. Lee and C. Y. Suen [1988], "Structural classification and relaxation matching of totally unconstrained handwritten zip=code numbers", *Pattern Recognition* Vol 21(1), 19–31.

[11] Le Cun Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, and H. S. Baird [1990], "Constrained neural network for unconstrained handwritten digit recognition", *Proc. International Workshop on Frontiers in Handwriting Recognition*, Concordia University, Montreal, April 1990, 145-154.

[12] D. T. Lee [1982], "Medial Axis Transformation of a Planar Shape," *IEEE Trans. Patt. Anal. Machine Intell.* PAMI-4, No. 4, 363–369.

[13] F. Leymarie and M. D. Levine [1992], "Simulating the grass-fire transform using an active contour model," *IEEE Trans. Patt. Anal. Machine Intell.* PAMI-14 No 1, 56–75.

[14] D. C. Marr and E. Hildreth,[1980] "Theory of edge detection," *Proc. Royal Soc. London.,* B 207, 187–217.

[15] N. Mayya and V. T. Rajan [1994] "Voronoi Diagrams of Polygons: A Framework for Shape Representation," *Proceedings of IEEE CVPR 1994*, Seattle, Washington (to appear).

[16] N. Mayya and V. T. Rajan [1993] "Voronoi Diagrams of Polygons: A Framework for Shape Representation," IBM Research Report RC 191282, 11/23/93.

[17] N. Mayya and V. T. Rajan [1993] "An Efficient Shape Representation Scheme using Voronoi Skeletons", IBM Research Report RC 19161, 09/14/93.

[18] Nadal C. and C. Y. Suen [1988], "Recognition of totally unconstrained handwritten digit by decomposition and vectorisation," Technical Report, Concordia University, Montreal.

[19] Siavash N. Meshkat & Constantine M. Sakkas [1987], "Voronoi Diagram for Multiply Connected Polygonal Domains II: Implementation and Application," *IBM J. Res. Develop.* 31, No. 3, 373–381.

[20] S. Mori, C. Y. Suen and K. Yamamoto [1992], "Historical Review of OCR Research and Development," *Proceedings of the IEEE*, Vol 80, No. 7. 1029–1058.

[21] U. Montanari [1969], "Continuous Skeletons from Digitized Images," *JACM*, 16, No. 4, 534-549.

[22] U. Montanari [1968], "A method for obtaining skeletons using a quasi-Euclidean distance," *J. Assoc. Comput. Machinery*, Vol. 15, 60–624.

[23] R. Ogniewicz and M. Ilg [1992], “Voronoi Skeletons: Theory and Applications”, *Proc. ICPR*, pp 63–69.

[24] Vijay Srinivasan & Lee R. Nackman [1987], ”Voronoi diagram for multiply connected polygonal domains I: Algorithm,” *IBM J. Res. Develop.*, 31, No 3, 361–372.

[25] V. Srinivasan, L.R. Nackman, J. Tang and S.N. Meshkat [1992], “Automatic Mesh Generation using the Symmetric Axis Transformation of Polygonal Domains,” *Proceedings of the IEEE*, Vol. 80, No. 9, 1485–1501.

[26] Theo Pavlidis [1982], “Algorithms for Graphics & Image Processing”, Computer Science Press.

[27] Theo Pavlidis [1993], “Recognition of printed text under realistic conditions”, *Pattern Recognition Letters,* Vol. 14 317–326.

[28] F. P. Preparata and M. I. Shamos [1985], *Computational Geometry-An Introduction*, Springer Verlag, New York.

[29] Sargur N. Srihari [1993], “Recognition of handwritten and machine-printed text for postal address interpretation”, *Pattern Recognition Letters*, Vol 14, 291–302.

[30] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet and L. Lam [1993], “Building a new generation of handwriting recognition systems,” *Pattern Recognition Letters* Vol 14, 303–315.

[31] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet and L. Lam [1992], “Computer Recognition of Unconstrained Handwritten Numerals,” *Proceedings of the IEEE* VOl 80, No 7, 1162–1180.