FEATURE ANALYSIS USING LINE SWEEP THINNING ALGORITHM

Fu Chang[1], Ya−Ching Lu[1], and Theo Pavlidis[2]
[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
[2] Dept. of Computer Science, SUNY, Stony Brook, NY 11794−4400, U.S.A.

ABSTRACT

In this article, we propose a new thinning algorithm based on line sweep operation. A line sweep is a process where the plane figure is divided into parallel slabs by lines passing through certain "events" and items are then processed according to an order of the slabs. Assuming that the contour of the figure to be thinned have been approximated by polygons, the "events" are then the vertices of the polygons and the line sweep algorithm looks for pairs of edges that lie within each slab.

The pairing of edges are useful for detecting both regular and intersection regions. The regular regions can be found at the site where pairings between edges exist. Intersection regions, on the other hand, are where such relations would cease to exist, due to the fact that pair relations between edges of wide distance were canceled. A salient feature of our approach is to find simultaneously the set of regular regions that attach to the same intersection region. Such a set is thus called an intersection set.

The output of our algorithm consists of skeletons as well as intersection sets. Both of them can be used as features for subsequent character recognition. Moreover, the line sweep thinning algorithm is efficient in computation as compared with a pixel−based thinning algorithm which outputs skeletons only.

1. INTRODUCTION

A common practice of optical character recognition (OCR) is to proceed with a process which extracts features out of character images. To cope with the high degree of deviations of characters in either machine−printed or handwritten form, the extracted features are expected to meet the following three fundamental requirements. (1) They agree with the intuition of human beings. (ii) Their representations are tractable for the purpose of subsequent computations in character recognition. (iii) Their types are stable relative to possible deviations of characters.

Strokes have long been suggested as a type of feature for OCR purpose, for the obvious reason that they form the units of characters. From the image point of view, however, strokes are inferred entities. Moreover, the task of the inference is made more difficult by the fact that many strokes do not exist in isolation, but intersect with or merge into other strokes. Examples of stroke intersection occur in the English letters, such as 'T,' 'X,' etc., where two strokes either touch or cross each other. Stroke merging can be seen in the case of two interconnected letters 'tt' in which two horizontal strokes merge into one.

Pixel−based thinning algorithms (see [1] for survey), which are based on gradually removing pixels from characters, do not solve the above problems well. In spite of large time consumption on a sequential machine, many of the algorithms result in skeletons which are ambiguous in the intersection regions.

An alternate approach has been direct vectorization of the planar regions composing the shape. The earliest such attempt was based on linking black intervals on adjacent scan lines thus forming stacks of vertical or nearly vertical runs that are then mapped into vectors. This simple approach has trouble handling nearly horizontal strokes. A more sophisticated approach along these lines is described in [2]. That paper includes references to earlier work going as far back as 1958. While this method handles successfully cases that earlier approach could not, it still suffers from some of the same problems of pixel based techniques, although it is much faster in computation.

The recent efforts in coping with the above difficulties have developed along two directions. In one of them, an attempt is to divide the plane figure of character into areas that are non−overlapping parts of strokes and those that are overlapping parts. This direction has been taken by Mahmoud, Haiba and Green [3], Li and Suen [4], and also Nishida, Suzuki and Mori [5]. The commonalty among these approaches is to seek a way to identify the compact shapes in a figure before thinning them into lines. Nishida, Suzuki and Mori, in particular, considered the region to be thinned as consisting of two contour arcs that can be matched to each other.

They named this type of region as regular, as opposed to singular region that is not to be thinned.

The other direction is taken by Chouinard and Plamondon [6]. Their approach is to use a line−following scheme to move along pairs of pixels that lie on the character contour. Each pair of pixels is supposed to determine a point on the thinned line to be produced. Rather than bypassing thinning operation to the intersection regions, their scheme uses a window to detect the shape and type of each region to make more accurate thinning.

The current paper develops a method to identify the regular regions, in the sense of Nishida, Suzuki and Mori, and it also analyses the type of intersection regions, studied by Chouinard and Plamondon. Our method is based on a general concept of *line sweep* (also called *plane sweep* [7], pp. 10−11). One major advantage of using a general method is the ease of implementation. In addition, it results in faster execution because of the efficiency of such a procedure.

The proposed method assumes that the contour of the shape to be analyzed has been approximated by polygons. It then looks for pairs of edges of the polygons that are nearly parallel and are also close to each other according to certain measure. The pairing of edges is useful for detecting the existence of both regular and intersection regions and also for determining their shape. For example, regular regions can be found at the site where pairing between edges exist. Such a pairing relation ceases to exist at intersection regions. A salient feature of our approach is to find simultaneously the set of regular regions that attach to the same intersection region. Such a set is called an *intersection set*. We base this approach on the observation that, although an intersection set is structurally more complex than a stroke, the former is nevertheless more basic and also more stable than the latter from the image viewpoint.

When all the intersection sets have been determined, it is possible to obtain a skeleton throughout the character figure. The outcome of our algorithm is not only a set of thinned lines, but also the identification of the pattern that is associated with each intersection set. It is felt that the pattern identity of intersection sets can sometimes be more useful for OCR purpose.

For one thing, in the case of connected letters 'tt,' it is more important to detect two crosses within the connected figures than merely constructing the thinned lines. While the description of the algorithm is rather complicated, the time of its execution is quite short as we show in Section 6.

This article is organized as follows. The next section describes the line sweep procedure which pairs polygon edges. Section 3 shows how paths are formed out of edges by means of their pairing relations. Section 4 consists of the description of the procedure for finding intersection sets and the application to feature analysis. Section 5 describes thinning process and Section 6 is about experimental results. Finally, a conclusion is made at Section 7.

2. LINE SWEEP ALGORITHM

Line sweep is a process where the plane figure is divided into parallel slabs divided by lines passing through certain "events" and items are then processed according to an order of the slabs. In our case, the "events" are vertices of the polygon. It is easy to construct a linked list for each slab in which the edges are sorted according to certain order. In the example of Figure 1, the dashed lines define the slabs which are labeled by capital letters. In each slab, the edges can be ordered from left to right. For example, edge (1,2) can be placed ahead of (8,1) in the linked list for slab A, and (1,2) ahead of (7,8) in the list for slab B, etc.

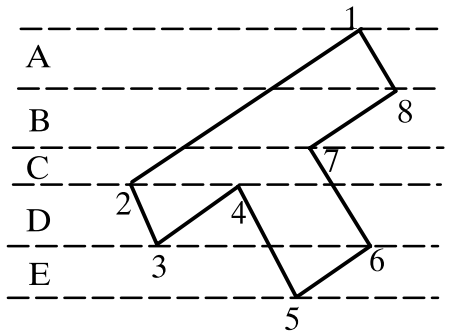

Figure 1. The plan is divided into parallel slabs.

Usually the number of edges per slab is much smaller than the total number of edges, so that algorithms of complexity $O(N^2)$ are converted into algorithms of complexity $O(N\log_2 N)$.

In our case, rather than considering all possible pairs to find which ones are nearly parallel, we sort them first and then apply the line sweep algorithm.

The algorithm may miss edges that are nearly parallel to the slab direction, so it is applied twice to find all pairs of edges that are mutually visible either in the horizontal or in the vertical direction. Two matrices M_H and M_V are used to register the mutually visibility of edges in the horizontal and vertical direction, respectively. They are defined as follows. Let S be a structure associated with an edge with two members {x, j}, where j is the label of S and x is the coordinate of the intersection of S with a horizontal sweep line (one per slab) measured from an arbitrary start. Let S[1], S[2], . . . , S[n] be a list of the structures sorted according to x. If we apply a line sweep algorithm in the horizontal direction that avoids singularities (for example [8]), n will always be even. Then we set

$$M_H[S[i].j][S[i+1].j] = M_H[S[i+1].j][S[i].j] = 1,$$

for $i = 1, 3, 5, \ldots, m-1$, and zero otherwise.

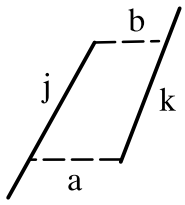

Figure 2. The region between j and k is the area enclosed by j, k and the two horizontal lines. The distance between j and k is min(a,b).

In Figure 2, edge j and k are mutually visible in the horizontal direction, i.e., $M_H[j][k] = M_H[k][j] = 1$. The area enclosed by j, k and the two (dashed) horizontal lines is called the *region* between j and k, a useful notion in later context. To avoid pairing edges which are widely apart, we would always reset $M_H[j][k]$ and $M_H[k][j]$ to zero, if the distance between j and k, defined in Figure 2, is greater than a threshold value Th_H. The value we use in our experiments for Th_H is 1/6 width of the box that encloses a given character.

The matrix M_V can be similarly defined, and is also subject to a similar distance restriction.

When j and k are two edges with $M_H[j][k]=1$ ($M_V[j][k]=1$), we say that j is a *H−counterpart* (*V−counterpart*) of k and vice versa. Moreover, we say that j is a *left* (*upper*) or *right* (*lower*) edge, according to whether j lies at an odd or even position of the linked list associated with a slab determined by horizontal (vertical) line sweep. In the example of Figure 1, (1,2), (2,3), and (4,5) are left edges and (3,4), (6,7), (7,8) and (8,1) are right edges.

3. CONSTRUCTION OF PATHS

Intuitively, a stroke can be conceived as a pair of arcs with maximal matching. To make this concept rigorous, we use two linked lists, each representing one of the paired arcs.

We start with constructing paired arcs with the help of the relationship represented by M_H and M_V. The resulting entities are called H− and V−paths. We then merge H− and V−paths into extended paths. Details of construction are given as follows.

3.1 Construction of H− and V−Paths

We start with the construction of H−paths. In this case, we use a linked list A to collect left edges and another linked list B to collect right edges. The operation of adding (i.e, inserting or appending) edges to A or B is as follows.

First, we say that two edges are vertex−connected, if they share a common vertex. On the other hand, edge m and n are said to be H−connected, if there exists a horizontal line l between m and n. Finally, two edges are said to be connected, if they are either vertex−connected or H−connected.

Now, starting with a pair of empty linked list A and B, we can add first a left edge j to A and then a right edge k to B, if k is a H−counterpart of j. At the next step, we add a left edge m to A, if m satisfies the following condition.

Condition C: m is connected with an outstanding (i.e., the first or last) element of A, and m has a H−counterpart n such that n is either itself an outstanding element of B or n is connected with an outstanding element of B. In case m is H−connected with an outstanding ele-

ment p of A, we shall add both m and m′ to A, where m′ is the horizontal line lying between m and p (Figure 3).

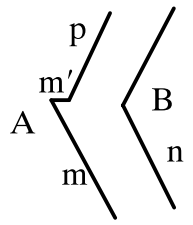

Figure 3. Edge m is H−connected with an element p of A, and m′ is a horizontal edge lying between m and p.

We keep on adding left edges to A, as long as they satisfy Condition C one after the other. Then, we go on adding right edges to B, so long as each one of them satisfies a similar condition. The operation is switched between A and B alternatively, until no edge can be possibly added to either of them.

Having constructed a pair of linked lists, we go on to construct another pair of linked list by adding an unused edge to each of them and proceeding as in the above. When we have constructed all possible pairs of linked lists, we want to further determine the *endpoints* for each linked list. For this purpose, we assume the following order convention for edges and vertices.

Convention: The points on a polygon and the edges of a polygon are ordered in such a way that when one moves along the polygon from one point to another point, or from an edge to another edge, the interior of the polygon always lies on one's left−hand side (for example, [10]). When point (edge) Γ comes ahead of point Δ (edge) in such an order, we use "$\Gamma \rightarrow \Delta$" to represent their relation.

We say that π is a point of A, if π lies on an edge contained in A. Also, we say that e is an edge of $P=\langle A,B\rangle$, if e is contained in either A or B.

Now for a given pair of linked lists $\langle A,B\rangle$, we first determine the *endpoints* for A as follows.

Let α and β be the first and last vertex of A, where the order of vertices are specified as in the Convention. The endpoints of A will be derived from α and β in the following way (Figure 4a).

Suppose that α lies on edge j_α of A. If j_α does not appear in any other paired linked list, then α is taken as an endpoint of A. Otherwise, we choose a point π_α as follows. Let $\Omega_\alpha = \{\xi: \xi$ is on j_α and ξ lies on a horizontal line that passes through a vertex of B$\}$. Then π_α is taken as the point in Ω_α such that $|y(\alpha) - y(\pi_\alpha)| = \min\{|y(\alpha) - y(\xi)| : \xi \text{ is in } \Omega_\alpha\}$, where $y(\cdot)$ is the y−coordinate of a given point. We derive the other endpoint of A from β in a similar way. The endpoints for B can be determined similarly.

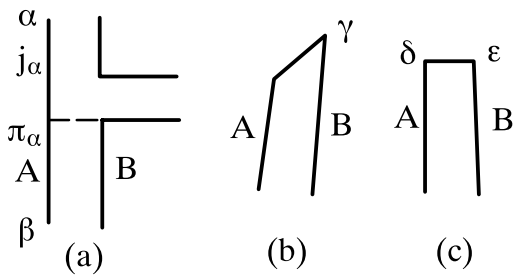

Figure 4. (a) Creation of a new vertex π_α and a new edge (π_α, β). (b) Vertex γ is an endpoint. (c) Vertices δ and ε are also endpoints.

Remark: In Figure 4, not all edges on the character contour are shown. But it is assumed that if an edge is not shown in the figure, it is not part of the path discussed in the text. The same remark also applies to all the forthcoming figures.

In what follows, we shall refer to the endpoints of paths also as vertices. The line segments between two consecutive vertices are still called edges. One can conceive the original polygon edges as now divided into several pieces at the newly added vertices, and each of the pieces now being taken as an edge. Thus, back in the example of Figure 4a, point π_α is now taken as a vertex and (π_α, β) is taken as an edge.

We now summarize the procedure for constructing paired linked lists as follows.

1) Starting with two empty linked lists A and B, add any left edge to A.

2) Insert or append right edges to B, if they satisfy Condition C.

3) Insert or append left edges to A, if they satisfy a similar condition.

4) Iteratively apply 2) and 3) until no more edge can be inserted or appended to A or B.

5) Construct all other pairs of linked lists, following steps 1 to 4.

6) Determine endpoints for each linked list thus constructed.

If A and B are two linked lists constructed in the above procedure, we shall refer to the pair P=⟨A,B⟩ as a *H−path*. We shall refer to A and B as *flanks* of P. It can be easily shown that a flank is always a continuous curve. A *V−path* can be similarly defined as a pair of linked lists Q=⟨C,D⟩, where C consists of upper edges and D consists of lower edges. Linked lists C and D are also referred to as flanks of Q.

Suppose that ⟨A,B⟩ is a H− (V−) path. If A and B share an endpoint γ (Figure 4b), then γ is called a *terminal*. If, on the other hand, an endpoint δ of A is connected with an endpoint ε of B by way of a horizontal (vertical) line (Figure 4c), then δ and ε are also called terminals.

Two flanks are said to be *connected*, if one of the following conditions holds. (i) They share a common endpoint ε. (ii) They share a common edge. (iii) Their endpoints are connected by edges which do *not* have any H− or V−counterpart (in short, counterpart).

3.2 Construction of Extended Paths

We now construct *extended paths* out of H− and V−paths in the following way.

A H−path R can be *merged* with a V−path S to form an extended path T, provided each flank of R is connected with a flank of S, and conversely.

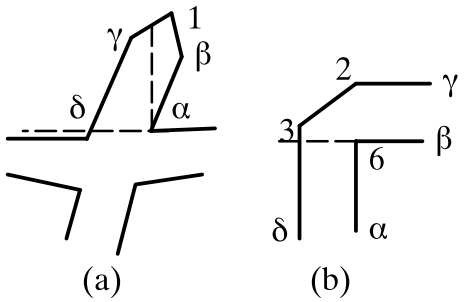

Figure 5. (a) Two paths with a terminal are merged into an extended path. (b) Two paths without any terminal are merged into an extended path.

In Figure 5a, the H−path R=⟨{(1,γ),(γ,δ)},{(α,β),(β,1)}⟩ can be merged with the V−path S=⟨{(β,1),(1,γ)},{(α,β)}⟩. In Figure 5b, the H−path R=⟨{(3,δ)},{(α,6)}⟩ can be merged with the V−path S=⟨{(γ,2)}, {(6,β)}⟩. Note that, in Figure 5b, the flank {(γ,2)} of S does not share any endpoint or edge with the flank {(3,δ)} of R. Instead, their endpoints are connected by edge (2,3), which has no counterpart.

When two paths R and S are merged into an extended path T, their flanks A and B are determined as follows. First, we consider the case in which R and S have a terminal (Figure 5a). Let $\Lambda = \{\varepsilon: \varepsilon$ is an endpoint of R or S$\}$, α be the first element of Λ, and δ the last element of Λ, where the order is given as in the Convention. Also, let $\Pi = \{\tau: \tau$ is a terminal of R or S$\}$, β be the first element of Π, and γ be the last element of Π. Then, let A be the linked list which is composed of all edges lying between γ and δ, and B be the linked list which is composed of all edges lying between α and β. Next, we consider the case in which R and S do not have any terminal (Figure 5b). Then, the extended path T has two separate connected sets. Let α and β be the first and last endpoint of one set, and γ and δ be the first and last endpoint of the other set. Then, again, let A be the linked list which is composed of all edges that lie between γ and δ, and B be the linked list which is composed of all edges that lie between α and β. In either case, the extended path T is represented as $\langle$A,B$\rangle$.

We can further merge an extended path T with another H− or V−path U to generate an even larger extended path, provided the same conditions for merging holds between T and U. When the new extended path is generated, its flanks are also determined in the same way as described in the above. The merging is continued, until there is no more H− or V− path that can be merged with the extended path lastly formed.

Having completed the construction of an extended path, we mark those H− and V−paths comprised in it. We then continue to construct other extended paths out of the unmarked H− or V−paths, until we exhaust all the possible ones. Note that the procedure for constructing extended paths is independent of the order of construction.

For a H− or V−path R to be merged with another path S, every flank of R has to be connected with a flank of S, *and* also conversely. In Figure 6, path R=$\langle\{(4,5)\}, \{(6,7)\}\rangle$ can *not* be merged with S=$\langle\{(1,2)\},\{(3,4),(4,5),(5,6),(6,7),(7,8)\}\rangle$, since both flanks of R are connected with the right flank of S but the left flank of S is not connected with any flank of R. In this example, however, all the edges of R are comprised in a flank of S. When this is the case, we say that R *embeds* in S.

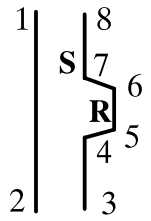

Figure 6. The V−path R embeds in the extended path, which is identical to the H−path S in this example.

From now on, we shall only deal with extended paths. When no confusion can be possibly generated, we shall simply refer to them as *paths*. For the time being, we shall focus on the paths which do not embed in any other paths. Those which embed in some other paths will be treated as special case.

4. FEATURE ANALYSIS

The paths constructed thus far do not exactly match with our intuitive notion of strokes. In Figure 7, for example, we see that path P=⟨{(2,3),(3,4)},{(7,8),(8,1)}⟩ overlaps much with a vertical stroke. However, P contains an edge (7,8) which in our intuition should be part of a horizontal stroke.

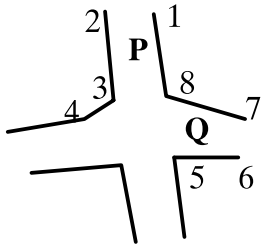

Figure 7. The edge (6,7) is contained in both the horizontal and vertical stroke.

Note that, in Figure 7, if the cross formed by the horizontal and vertical stroke occupies a large proportion of the character, such as in English letter 't,' then the pairing between edges (3,4) and (7,8) would be eliminated due to their wide distance. However, when the cross only occupies a small fraction of the character (e.g. some Chinese character), the pairing would not be eliminated, giving rise to the above problem.

In a series of papers [9−11] reporting our former experimental works, we viewed the problem as generated by the conflict between two paths. One of them is path

P=⟨{(2,3),(3,4)},{(7,8),(8,1)}⟩, and the other is path Q=⟨{(7,8)},{(5,6)}⟩. Since both P and Q contain edge (7,8), there is a conflict between them. Our approach was then to use rules to resolve the conflict based on the shape of P and Q. The disadvantage of this approach lies in the fact that the shape of paths can vary drastically and it is rather difficult to come up with reasonable rules for conflict resolution.

This paper uses a new approach. Rather than determining the shape of each path individually, we determine simultaneously the shape of all paths that attach to the same intersection region. Since the location of the intersection region can not be known in advance, our approach relies on the following process. We start with a path P, and look for any path neighboring to it, and also any path neighboring to its neighbor, etc. When such a process leads back to P again, we have found an intersection set.

One immediate benefit of the above process is the following. Those paths which do not participate with any other paths to form an intersection set would be eliminated. The second benefit of the process is to provide a way to resolve the conflicts between paths, now formulated in the following way. When moving from one path to the next in the process, there might be ambiguities as to where to set a break point between two neighboring paths. This happens when two paths share some common edge(s), as in the example of Figure 7. To solve this problem, we allow ourselves to consider all possible combinations of candidates for break point. We then base on a measure of the shape of each possible combination to select the one with the best shape.

We shall now describe the detailed implementation of the algorithm in the forthcoming subsections.

4.1 Description of the Plan

The next two subsections carry the heaviest load of technical contents in this article, and deserve an overview here.

First we define a few terms. A pair of points forms an *end* of path, if one of them is the first endpoint of a flank, and the other is the last endpoint of the opposite flank. Thus, in the exam-

ple of Figure 8a, vertices α and β constitute an end of path P=⟨A,B⟩, where α is the last endpoint of flank A, and β is the first endpoint of flank B. Path P has another end, in this case degenerating into a single vertex γ. In the example of Figure 8b, both ends of path Q consist of two vertices.

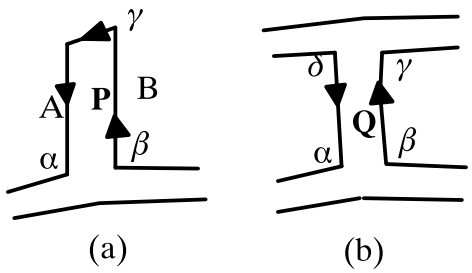

Figure 8. Points labeled by Greek letters constitute the endpoints of paths P and Q.

When an end consists of one vertex only or it consists of two vertices connected by edges belonging to the path, it is called a *closed end*. The endpoints of a closed end are called *terminals*. If an end is not a closed end, it is called an *open end*. When two endpoints α and β form an open end, we shall denote them as [α,β]. The order within the bracket is immaterial.

Recall that if j and k are two edges with $M_H[j][k]=M_H[k][j]=1$, then the area enclosed by j, k and two horizontal lines is called the region between j and k (cf. Figure 2). Now, if P is a H−path, then the region of P=⟨A,B⟩ is defined as the union of all the regions formed between pairs j and k, where j falls in A, k in B, and $M_H[j][k]=M_H[k][j]=1$. Note that the region of P is the area enclosed by A, B and two horizontal lines, called end−lines (Change et al. [16], Appendix A). If the region of a V−path is defined in a similar way, then we define the region of an extended path T to be the union of regions formed by all the H− and V−paths that are comprised in T.

If two paths do not share any contour point but their regions have non−trivial intersection, namely, their intersection is more than single points, then they are said to *cross* each other (Figure 9a). The necessary and sufficient conditions for two paths to cross each other is the following: both paths have two open ends and one of them is a H−path and the other a V−path and their end−lines intersect with each other ([16], Appendix A).

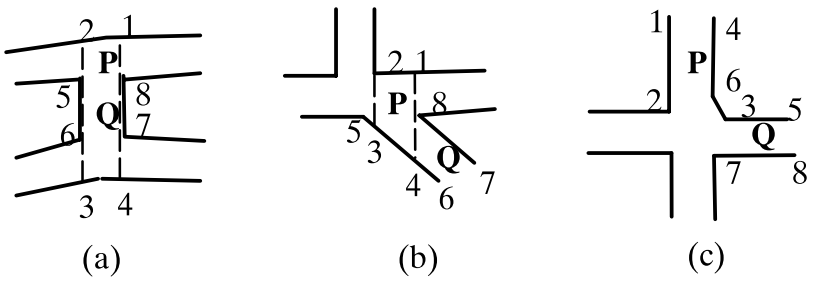

Figure 9. Let P=⟨{(1,2)},{(3,4)}⟩ and Q=⟨{(5,6),(7,8)}⟩. (a) P crosses Q. (b) P inserts in Q. (c) P makes normal intersection with Q.

If a flank of P is a proper subset of Q and another flank of P does not overlap with Q, then P is said to *insert* in Q (Figure 9b). On the other hand, if P does not cross or insert in Q, nor does Q insert in P, but P has a flank overlapping with a flank of Q, then P is said to make a *normal intersection* with Q (figure 9c).

Thus, for any two paths whose regions have non−empty intersection, only one of the following conditions can hold between them. (i) They cross each other. (ii) One of them inserts in the other. (iii) They make a normal intersection.

When condition (i) or (ii) in the above holds between two paths, each of them is called a *traversable* path. The rationale for this terminology is given as follows.

When two paths cross each other, or when one path inserts in another path, there is a conflict between the two paths and one of them must be completely or partially eliminated. Whether a path is retained or eliminated has much to do with its shape. As mentioned in the above, the shape of a path is determined by its relationship with all the paths that forms a synergy, to be called intersection set. Thus, we allow various synergies to be formed with or without the conflicting paths. Thus, a conflicting path can be excluded from a synergy even when it stands in the middle of the paths within this synergy. For this purpose, we adopt the following definition.

Path Q is a *neighboring* path of P, if one of the following conditions holds. (i) Q makes a normal intersection with P. (ii) A flank of P is connected with a flank of Q by some edges having no counterpart. (iii) A flank of P is connected with a flank of Q by some edges of a traversable path.

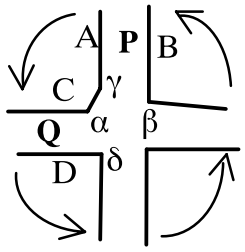

Figure 10. Movement from the flank A and endpoint α of path P.

To form a synergy of paths, we start with a path P=⟨A,B⟩ and an open end [α,β] of P. Our goal is to find a neighboring path Q=⟨C,D⟩ of P and an open end [γ,δ] of Q that is supposed to attach to the same intersection region as [α,β] (Figure 10). Note that it is *not* necessary that [α,β] and [γ,δ] share any common point. From Q and [γ,δ], we go on to find a path neighboring to Q and its associated open end, etc. If, by continuing this operation, we are able to get back to P and [α,β], then we have found a collection of paths that forms an *intersection set*.

It is possible that there exist more than one intersection set for a given open end of a given path. Thus, we need a way for deciding the most appropriate intersection set for it. When a unitary intersection set has been found, we can further determine the break points between the neighboring paths in such a set. The technique for finding the possible intersection sets, deciding the unitary one, and setting the break points will be described in Subsection 4.2.

With the technique developed there, we proceed in the following way.

(1) First, we examine all the paths which are in conflict with each other. They are the paths that cross or insert in other paths. For each open end of such paths, we find the associated unitary intersection set. With the break points between neighboring paths determined, we are then able to compute the length−to−width ratio of each conflicting path, and use this metric to determine whether to eliminate or retain each of the conflicting paths.

(2) The next step is to construct all the intersection sets for the remaining paths. For this purpose, we follow a simple sequential procedure. We look for a path in which there are still some unmarked edges, and find the unitary intersection set associated with each of its open ends. When such an intersection set is obtained, we mark all the edges and open ends comprised in it. We then go on to look for another path with unmarked edges and repeat the above

process. The construction of intersection sets is completed when we are left with no unmarked edges.

(3) With a modified technique, we can also obtain a unitary intersection set associated with an embedding path P and two separate parts of the path in which P embeds. We can then compute the length−to−width ratio of P and use this metric to determine whether to eliminate or retain P.

(4) For each of the remaining paths, we compute its length−to−width ratio and use this metric to determine whether to eliminate or retain the path. When a path is eliminated, a new intersection set will be constructed out of its neighboring paths.

The implementation details of the above listed steps will be developed in Subsection 4.3.

Having read through the overview as given in this subsection, the readers who are not anxious to immerse themselves in the technicality involved in the procedures can proceed to Section 5.

4.2 The Procedure for Finding Intersection Sets

This subsection is devoted to the procedure for finding, from an arbitrarily given starting point, all possible intersection sets and deciding the most appropriate one among them.

4.2.1 Starting with an Initial Path

Given a path P=⟨A,B⟩ and an open end [α,β] of P, with α being the last endpoint of A and β the first endpoint of B. To look for a neighboring path Q of P, we always start with the flank whose last endpoint falls within the open end [α,β]. By assumption, this is flank A. We now look for a path Q in connection with A. There are two possibilities. (i) Q does not have any flank sharing edges with A. (ii) Q has a flank sharing some edge with A. We shall consider both of them.

For the first possibility, we move from vertex α along the contour and look for the first edge e having a counterpart f (Figure 11a). Here, again, we assume that the edges are ordered according to the Convention specified in Subsection 3.1. We then look for a path Q with one flank

containing e and another flank containing f. Q is certainly not the same as P. If Q happens to be a traversable path, then we also look for the first edge e′ outside Q that has a counterpart f′. Let Q′ be the path with one flank containing e′ and another flank containing f′. By definition, both Q and Q′ are neighboring paths of P.

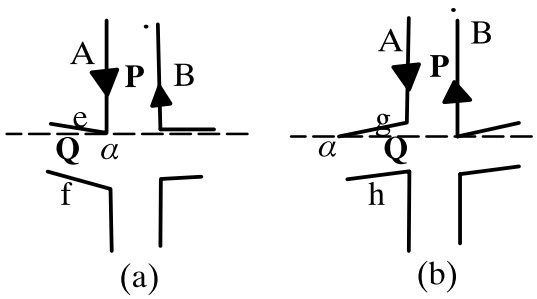

Figure 11. (a) Edge e lies outside of flank A. (b) Edge g lies within flank A.

For the second possibility, we look for all those edges within A which have a counterpart not in B. Let g be one such edge within A and let h be a counterpart of g not contained in B (Figure 11b). We then look for any path with one flank containing g and another flank containing h. This path is certainly not the same as P, since h is not contained in B. There may be more than one such g and h, and thus more than one neighboring path of P.

4.2.2 Proceeding from a Neighboring Path

Let Q=⟨C,D⟩ be a neighboring path of P=⟨A,B⟩. We want to determine the open end [γ,δ] of Q which would, intuitively speaking, attach to the same intersection region as the open end [α,β] of P.

Assuming that flank C is connected with flank A and α is an endpoint of A, we determine [γ,δ] in the following way. Since α is the last endpoint of A, γ is taken as the first endpoint of C (Figure 12). When γ has been determined, δ is taken as the endpoint of D such that [γ,δ] forms an open end of Q. Note that δ must be the last endpoint of D.

Having determined the open end of Q, we can then proceed as in Subsection 4.2.1 to find neighboring path R of Q, with a flank of R connected with flank D of Q (Figure 12).

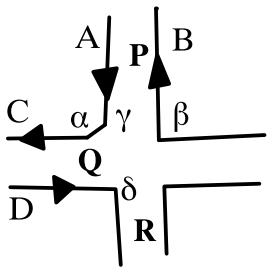

Figure 12. Determination of open end [γ,δ] of Q and a neighboring path R of Q.

4.2.3 Finding Intersection Sets

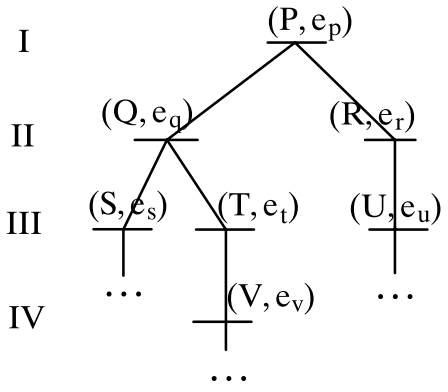

Figure 13. Tree representation of a procedure.

The procedure for finding intersection sets can be represented as a tree, illustrated in Figure 13, where the node (horizontal bar) represents a pair consisting of a path X and an open end e_x of X, and the link between a node A at level i and a node B at level i+1 signifies the fact that, when we apply the operation as described in Subsections 4.2.1 or that in 4.2.2 to A, we would obtain B. Thus, our procedure starts with the root of the tree, namely node (P, e_p) at level I. It then splits into as many sub−procedures as the number of possible neighboring paths obtained by the operation. Each of the sub−procedure, in turn, splits into one or more than one sub−procedure.

There are two stopping criteria for the tree development. (1) No branch will be developed from any node which is located at the N−th level. This means that we are only interested in intersection set with no more than N−1 paths. The number N, of course, varies in applications.

(2) When a node at any level between 1st and N−th is found to be identical to the root, no branch will be further developed from it.

We define T−sequence as a sequence N_1, N_2, ... such that N_i is a node at level i, and there is a link between N_i and N_{i+1}, for i= 1, 2, ... The purpose of our procedure is to find a T−sequence $N_1, N_2, ...$ such that $N_{J+1}=N_1$ and, moreover, no path appears twice in the subsequence N_1, N_2, ..., N_J. When this is the case, we say that there is an *intersection set* N_1, N_2, ..., N_J found within such a tree structure. Thus, for example, if node (V, e_v) in the tree structure of Figure 13 is identical to (P, e_p), then we have found an intersection set (P, e_p), (Q, e_q), and (T, e_t).

We now consider two examples.

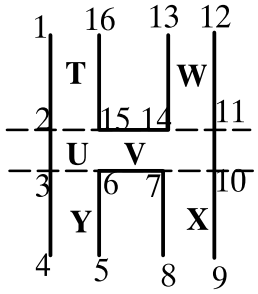

Figure 14. First example.

In the first example (Figure 14), we assume that U= ⟨{(2,3)},{(10,11)}⟩ is a path whose width does not exceed 1/6 width of the character (not completely shown). From U and e_u=[2,11], one obtains W=⟨{(13,14)},{(11,12)}⟩ as a neighboring path of U and e_w=[11,14] as the open end associated with W. Form W and e_w, one obtains V=⟨{(14,15)},{(6,7)}⟩ and e_v=[14,7]. Thenceforth, one obtains X=⟨{(7,8)},{9,10}⟩ and e_x=[7,10], and U and $e_{u'}$=[3,10]. Note that $e_{u'}$ is *not* the same as e_u. From U and $e_{u'}$ one further obtains the following subsequence: Y=⟨{(3,4)},{(5,6)}⟩ and e_y=[3,6], V and $e_{v'}$=[15,6], T=⟨{(1,2)},{(15,16)}⟩ and e_t=[2,15], and finally U and e_u. Thus, one has finally obtain U and e_u. However, the subsequence leading from U and e_u to themselves is not deemed as an intersection set, since paths U and V appear twice before U and e_u are ever reached.

Note that in this example paths U and V cross each other, and V serves as a ”wormhole,” through which one can travel from one end of U to its other end.

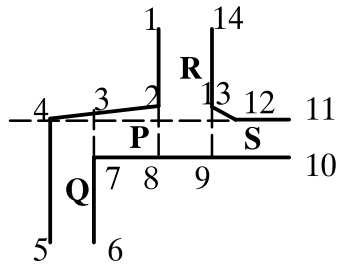

Figure 15. Second example.

In the second example (Figure 15), let P=⟨{(2,3)},{(7,8)}⟩. Note that path P does not end at point 4 but ends at point 3, due to the fact that the polygon edge (2,4) is divided into two pieces and the piece (3,4) becomes an edge (cf. Figure 4a for a similar case). Thus, from P and e_p=[2,8], one would obtain the following subsequence: S=⟨{(11,12),(12,13)}, {(9,10)}⟩ and e_s=[9,13], R=⟨{{(1,2),(2,3),(3,4)},{(12,13),(13,14)}⟩ and e_r=[4,12], Q= ⟨{(4,5)},{(6,7)}⟩ and e_q=[4,7], P and $e_{p'}$=[3,7]. The sequence leading from Q and e_q can not possibly contain an intersection set, since P appears twice and e_p is not yet reached.

In this example, path P inserts in path R and R serves as a "wormhole" for one to travel between two ends of P.

4.2.4 Determination of Break Points

Given an intersection set consisting of paths P_1, P_2, ..., P_J, we need to determine the break points between neighboring paths. For i=1, 2, ..., J, the *break point* between P_i and P_{i+1} is denoted as β_i, where P_{J+1} is assumed to be P_1.

Let us first determine the *candidates* for the break point between P_i and P_{i+1}. If the two paths meet at a single vertex γ, then γ is the only candidate. If, on the other hand, P_i and P_{i+1} share some edges, then we recruit all the vertices and midpoints of the common edges as the candidates. If P_i and P_{i+1} do not have any common edge or common endpoint, then their endpoints must be connected by some edges with no counterpart or some edges of a traversable path. In this case, the candidates are taken as the vertices and midpoints of those edges.

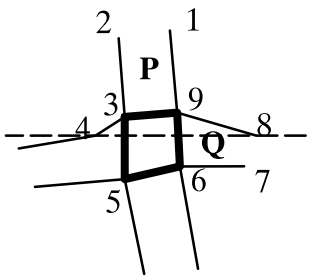

Figure 16. Vertices 3, 4, 5, 6, 8 and 9, and midpoints of (3,4) and (8,9) are candidates for break points.

In Figure 16, for example, P=⟨{(2,3),(3,4)},{(8,9),(9,1)}⟩ and Q=⟨{(8,9)},{(6,7)}⟩ share (8,9) as a common edge. Thus, vertices 8 and 9 and the midpoint of (8,9) are the candidates for break point between P and Q.

Let $S_i=\{\gamma: \gamma$ is a candidate for break point between P_i and $P_{i+1}\}$, for i=1, 2, ..., J. We now consider all the possible sequences of candidates $\gamma_1, \gamma_2, ..., \gamma_J$, where γ_i is in S_i for i=1, 2, ..., J. For each candidate sequence, we define its shape measure as $\sum_{i=1}^{J} s(\overline{\gamma_i\gamma_{i+1}})$, where s(.) is the size (i.e., length) of a given line segment. We then choose the sequence whose shape measure is smallest among all candidate sequences. The set of points in the chosen sequence will be called *junction.*

In the example of Figure 16, the junction consists of points 3, 5, 6 and 9. We use thick lines to connect the consecutive points in the junction. The region enclosed by the thick lines will be called *junction region*.

4.2.5 Selection of Intersection Sets

The existence of traversable paths (cf. Subsection 4.1) implies the possibility of obtaining more than one intersection set from the same open end of the same path. Two intersection sets are said to be *competing* with each other, if they both are obtained from the starting point. In this subsection, we describe the rule for eliminating intersection sets and that for selecting among competing intersection sets.

Let an intersection set IS comprise paths $P_1, P_2, ..., P_J$ and the break point between P_i and P_{i+1} be β_i for i=1, 2, ..., J. We set

$$d(IS) = \max_{i=1}^{J} s(\overline{\beta_1 \beta_i}),$$

namely, the maximal distance of β_1 to the other break points in IS. We also set

$$m(IS) = \max_{i=1}^{J} s(\overline{\beta_i \beta_{i+1}}),$$

namely, the size of the longest line segment connecting the break points in IS.

First, we stipulate the following rule.

Rule A. If an intersection set IS has $d(IS) > Th_J$, then IS is aborted.

The value of Th_J is taken as $\max(Th_H, Th_V)$, where we recall that Th_H (Th_V) is the threshold that regulates the distance between an edge and its H− (V−) counterpart. The reason to set such a restriction is to avoid mis−identifying as intersection set the smeared part of a character caused by certain kinds of pollution.

Next, we use the following rule as the selection rule for competing intersection sets.

Rule B. Let IS_1, IS_2, ..., IS_n be the intersection sets that are obtained from the same open end e_p of P. We select IS_i as the unitary intersection set associated with (P, e_p), provided $m(IS_i) = \min_{j=1}^{n} m(IS_j)$.

Referring back to the example of Figure 14, we see that path U and V cross each other. Thus, if we start from the pair (T, e_t), we obtain the following three intersections sets.

IS_1: (T, e_t), (Y, e_y), (V, e_v),

IS_2: (T, e_t), (U, e_u), (W, e_w),

IS_3: (T, e_t), (Y, e_y), (X, e_x), (W, e_w).

One can see that the longest line segment in IS_2 is spanned by break point 2 and 11, that within IS_3 is spanned by 14 and 15, and that within IS_3 is spanned by 15 and the midpoint of (2, 3). Thus, $m(IS_2) > m(IS_3) > m(IS_1)$ and IS_1 is the chosen winner, according to Rule B. We observe that the longer the path V, the better the chance for IS_1 to beat the other two intersection sets, since the longest line segment in IS_2 and that in IS_3 actually traverse through the region of V.

In the example of Figure 15, we see that path P inserts in path R. Since the two paths are traversable, it follows that if one starts with the pair (R, e_r), one would obtain two different intersection sets.

IS_4: (R, e_r), (P, e_p), (Q, e_q),

IS_5: (R, e_r), (Q, e_q).

Here again, we observe that the longer the path P, the better the chance for IS_4 to beat IS_5, since the only line segment connecting the break points in IS_5 traverses through the region of P. The two observations motivate us to define a shape measure called length−to−width ratio, to be described in Subsection 4.3.1.

In the end of this subsection, we put forth some remarks regarding how to accelerate the procedure for finding intersection sets.

Because of the existence of traversable paths, we may waste time on generating all possible nodes, while many of them are really useless. Thus, to avoid the repetitive search for the same neighboring paths from the same given paths, we can pre−store the relationship as represented by {(P, e_p), (Q, e_q)}, indicating that (Q, e_q) can be obtained from (P, e_p) by the procedure as described in Subsection 4.2.1 or that in 4.2.2. Moreover, we can also pre−store the candidates for break−point between (P, e_p) and (Q, e_q), so as to facilitate the procedure for determining break points for each intersection set that is found.

Next, in developing the tree structure as described in Subsection 4.2.3, we can do the following step to avoid conducting useless searches to further depth. For every node Nd_i generated at level i, we define

$$v(i)=\min\{s(\overline{\gamma_1\gamma_i}): \gamma_i \text{ is a candidate for break point between } Nd_i \text{ and } Nd_{i+1}\}.$$

If $v(i)>Th_J$, then we stop generating any new node from Nd_i. This is reasonable, since $v(i) \le s(\overline{\beta_1\beta_i}) \le d(IS)$, where β_i is the break point between Nd_i and Nd_{i+1} and IS is the intersection set, if existing, obtained by following this pathway. So if $v(i)>Th_J$, then so is d(IS). By Rule A, IS will be aborted.

4.3 Application to Feature Analysis

In the previous subsection, we described the procedure for obtaining the unitary intersection set from an arbitrarily given path and its open end. But we have not specified the situations to which the procedure is applied. In fact, we shall employ the procedure in the following order. First, we use it to make selections among the paths that are in conflict with each other. This work is to be described in Subsection 4.3.2. Next, we apply the procedure to establish all intersection sets for the remaining paths, to be described in Subsection 4.3.3 and 4.3.4. As a result, we are able to identify and thus eliminate spurious paths, to be described in Subsection 4.3.5. As we proceed along this line, a shape measure, called length−to−width ratio, for paths will be employed. We now define this measure as below.

4.3.1 Determination of Length−to−Width Ratio for Paths

For a given path P, we assume that a unitary intersection set has been found for each open end of P. We first consider the case that P has two open ends (Figure 17a). In this case, P has a neighboring path at each of its ends. Let A be a flank of P, and β and γ the break points that separate A from neighboring paths of P. Then all the edges between β and γ will be assigned to the *legitimate zone* of P, denoted as LZ(P).

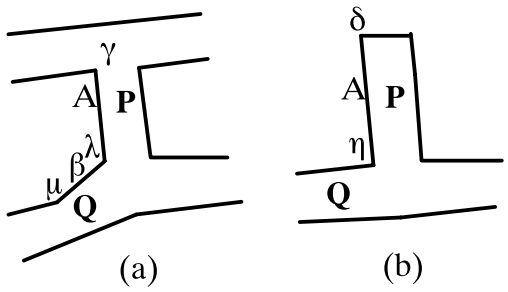

Figure 17. (a) All the edges between β and γ are assigned to LZ(P). Moreover, if β is a break point between P and Q, then $\overline{\lambda\beta}$ is assigned to LZ(P), and $\overline{\beta\mu}$ is assigned to LZ(Q). (b) All the edges between δ and η are assigned to LZ(P).

Moreover, if β lies within the interior of an edge (λ, μ), where $\lambda \rightarrow \mu$, then the line segment $\overline{\lambda\beta}$ is also assigned to LZ(P), while $\overline{\beta\mu}$ is assigned to the legitimate zone of Q, where Q is neighboring path of P (Figure 17a). Similar assignments are made if γ lies within the interior of an edge.

Let us now consider the case that P has only one open end (Figure 17b). In this case, let δ be a terminal of P, lying on flank A of P. Let η be the break point that separates A from a neighboring path of P. Then, all the edges between δ and η are assigned to LZ(P) (Figure 17b). When η happens to fall within the interior of an edge, similar assignments are also made as before.

LZ(P) will actually take the role that used to be played by P. We shall consider LZ(P) as consisting of two flanks. Thus far, all the edges and parts of edges assigned to LZ(P) constitute one flank of LZ(P). The other flank of LZ(P) can be obtained similarly.

Now, we determine the length−to−width ratio for P as follows.

Let l(P) be defined as $\sum_e s(e)/2$, where e ranges over all edges or parts of edges that are assigned to LZ(P). The metric l(P) is supposed to measure the length of P.

If P has two open ends (Figure 18a), let α, β, γ and δ be the break points that separate P from its neighboring paths. Moreover, we assume that α and β belong to the same flank of LZ(P) and $\alpha \rightarrow \beta$. We also assume that similar relation holds between γ and δ. Then we define w(P) as $(s(\overline{\alpha\delta}) + s(\overline{\beta\gamma}))/2$. In this case, α (β) is said to *pair with* δ (γ).

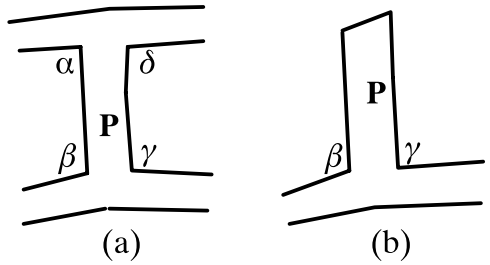

Figure 18. (a) P has two open ends and $w(P)=(s(\overline{\alpha\delta})+s(\overline{\beta\gamma}))/2$. (b) P has only one open end and $w(P)=s(\overline{\beta\gamma})$.

If, however, P has only one open end (Figure 18b) and thus has only two break points, say β and γ, then w(P) is defined as $s(\overline{\beta\gamma})$. In this case, β is said to pair with γ.

The metric w(P) is supposed to measure the width of P.

Now, we define the length−to−width ratio of P as r(P)=l(P)/w(P).

4.3.2 Selection among Conflicting Paths

Two paths are said to be in conflict, if they cross each other, or one of them inserts in the other. In the former case, one of the conflicting paths must be eliminated. In the latter case, both conflicting paths may be retained. The decision as to whether to retain or eliminate a conflicting path is given as follows.

Let us assume that P crosses some other path(s). Note that P must have two open ends ([16], Appendix A). Thus, for each open end of P we can find the associated intersection set. Using the procedure as described in Subsection 4.3.1, we can then determine the length−to−width ratio r(P) of P.

Let Q be the path which crosses P. We follow the same procedure as described in the above to determine r(Q). We then check the two paths by the following rule.

Rule 1. When path P crosses path Q and $r(P)<r(Q)$, P is eliminated.

Thus, in the example of Figure 14, it can be easily seen that path U has much smaller length−to−width ratio than path V, and is eliminated according to the above rule.

Rule 1 has the following implication. Whenever P is found to have smaller length−to−width ratio than a path which crosses it, P is eliminated, regardless of whether P has larger length−to−width ratio than some other path which also crosses it. Thus, for example, if we have three paths P, Q and R such that P crosses Q and P also crosses R, while $r(P)<r(Q)$ and $r(P)>r(R)$, then both P and R are eliminated.

Let us now assume that P is a path that inserts in path Q. Here, again, P must have two open ends ([16], Appendix B). Thus, we use the same procedure as described in Subsection 4.3.1 to compute the length−to−width ratio r(P). We then check path P by the following rule.

Rule 2. For each path P that inserts in another path, if $r(P)<1/2$, then P is eliminated.

An example of a path that inserts in another path is P in Figure 15, wherein the readers might agree that the length−to−width ratio is a reasonable criterion for the elimination of P.

When a traversable path P is eliminated, the counterpart relationship between all the edges of P will be canceled. That is, $M_H(i,j)$, $M_H(j,i)$, $M_V(i,j)$ and $M_V(j,i)$ will be reset to zero for all

pairs of i and j, with i belonging to a flank of P and j belonging to another flank of P. On the other hand, when path P is retained, P will no longer be marked as traversable path.

4.3.3 Establishment of All Intersection Sets

Having completed the procedure as described in the previous subsection, we are now left with all the paths which are not in conflict with any other paths. One of the remaining work is to construct intersections sets for them.

To generate all the intersection sets, we follow a very simple sequential procedure. Starting with a pair (P, e_p), where P is a path and e_p an open end of P, we go find the unitary intersection set by the technique that has been described in Subsection 4.2. When we have successfully found an intersection set, we mark all the edges and open ends comprised in this set.

We then start with another open end of P, if existing, and look for the intersection set associated with this new starting point. Note that, under the present circumstance, when we develop the tree structure as described in Subsection 4.2.3, we would put forth a new constraint as follows. Whenever a node generated at certain level contains a marked open end, we would stop generating any new node out of it. When the new intersection set is found, we shall once again mark all the open ends and edges in the set.

To continue, we can pick up another path Q. It can be proved that if Q has some unmarked edges, then one must be able to find an intersection set with each open end of Q serving as starting point ([16], Appendix C). Thus, if there are unmarked edges in Q, we shall apply the above procedure to each open end of Q, and also mark all the open ends and edges in the associated intersection set. If, however, there is no unmarked edge in Q, then we abort Q.

We go on to find paths with unmarked edges and repeat the same process as before. The construction of intersection sets is completed, when we have exhausted all the unmarked edges on the character contour.

To save computational efforts, we may employ the intersection sets which have been constructed for the paths that were marked as traversable paths and are now kept as ordinary paths. But before employing these sets, we have to subject them to a check. That is, when an

unmarked open end of a path is known to be comprised in an intersection set of this sort, we have to ensure that the intersection set does not contain a path that has already been eliminated, and it does not contain an open end that has already been marked. When such an intersection set passes the check, we would accept it as a newly generated set and mark all the open ends and edges in it accordingly.

4.3.4 Dealing with Embedding Paths

Thus far, we have only dealt with the paths which do not embed in any other paths. In this subsection, we begin to deal with the embedding paths. As is shown in Figure 19, path R=⟨{(6,7)},{(4,5)}⟩ can form an intersection set with two separate parts of path S=⟨{(1,2)}, {(3,4),(4,5),(5,6),(6,7),(7,8)}⟩. This happens when R is a path embedding in P. To find such an intersection set, however, the procedure described previously do not apply. We need the following modification of the procedure.

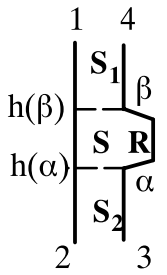

Figure 19. Path R can form an intersection set with two separate parts S_1 and S_2 of S, where R embeds in S.

Let us first define two functions. For each point α that lies on an edge j, we define h(α) (v(α)) as the point at which the horizontal (vertical) line passing α intersects with edge k, where k is a H− (V−) counterpart of j.

Suppose a V−path R embeds in a path S (Figure 19). We first seek a way to divide S into two separate parts. Let Σ be the set of edges in R for which there is a H−counterpart. Let α and β be the first and last vertex of the edges in Σ. Taking h(α), h(β) as newly added vertices, we can use them together with α and β to subdivide path S into two separate parts. In the exam-

ple of Figure 19, the two separate parts are $S_1=\langle\{(1,h(\beta)\},\{(\beta,4)\}\rangle$ and $S_2=\langle\{(h(\alpha),2\},\{(3,\alpha)\}\rangle$.

Treating now S_1 and S_2 as new paths, we can use them together with path R to construct an intersection set. When the break points in this intersection set have been determined, we are able to compute the length−to−width ratio r(R) of R. We now stipulate the following.

Rule 3. If r(R) is greater than or equal to 1/2, then we eliminate S and keep R, S_1 and S_2 as ordinary paths. Otherwise, we eliminate path R, S_1 and S_2, and retain S as an ordinary path.

4.3.5 Identification and Elimination of Spurious Paths

In the example of Figure 20a, Path $P=\langle\{(2,3)\},\{(3,4)\}\rangle$ fits our intuitive notion of a spurious path, because it does not correspond to any significant part of a character. One possible characterization of P is that it splits at its one end into two paths, i.e., $Q=\langle\{(1,2)\},\{(7,8)\}\rangle$ and $R=\langle\{(6,7)\},\{(4,5)\}\rangle$. This characterization, however, can not be used as a criterion for spurious paths, since path $S=\langle\{(2,3),(3,4)\},\{(5,6),(6,7)\}\rangle$ in Figure 20b meet the same criterion but should not be taken as a spurious path.

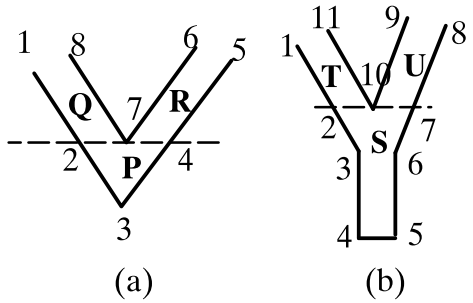

Figure 20. Path P and S split into two paths at the upper end.

In the example of Figure 21, we have a similar situation in which path $V=\langle\{(1,2),(2,3)\},\{(5,6),(6,7)\}\rangle$ in Figure 21a splits into two paths at both ends and should be taken as a spurious path, while path $W=\langle\{(7,8),(8,9)\},\{(11,12),(12,13)\}\rangle$ in Figure 21b meet the same criterion but should not be taken as a spurious path.

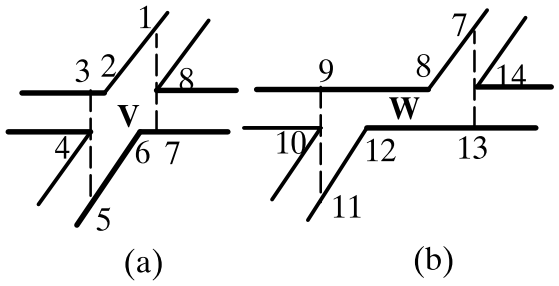

Figure 21. Both path V and W split into two paths at each end.

In the above examples, what actually discriminates one path from the other (P from S in Figure 20, or V from W in Figure 21) is their length−to−width ratio, rather than their relation to neighboring paths. Thus, we stipulate the following.

Rule 4. If the length−to−width ratio r(P) of P is less than 1/2, then P is taken as a spurious path.

Thus, in Figure 20a, the break points between path P and Q are 7 and 2 and those between P and R are 7 and 3. The junction associated with path P, Q and R then consists of points 7 and 3. Thus, LZ(P) consists of 7 and 3 only. It follows that l(P)=0 and P is a spurious path.

In Figure 21a, the break points between V and one of its neighboring path consists of points 2, 4, and 6, and that between V and its other neighboring path consists of points 2, 6, and 8. LZ(V) then consists of 2 and 6 only. It follows that l(V)=0 and V is a spurious path.

When a path P is judged to be spurious, P and the intersection set(s) associated with open end(s) of P are eliminated. Moreover, the counterpart relationship between all the edges within P will be canceled. When P is eliminated, we shall find a new intersection set starting from a path that used to be neighboring to P. Thus, for example, in Figure 20a, the new intersection set is found to consist of path Q and R and in Figure 21a, the new intersection set consists of the four neighboring paths of V.

5. CONSTRUCTION OF THINNED LINES

Having constructed all the intersection sets, we are now ready for constructing thinned lines for the paths that fall within each intersection set.

Recall that in the previous section, we were able to determine the points, called break points, that demarcate each pair of neighboring paths falling in the same intersection set. Moreover, we determined the legitimate zone LZ(P) of a path P as the collection of edges lying between the break points associated with P. The break points and legitimate zones, taking the role played by endpoints and paths, were useful in many operations. Now, we also use them for the construction of thinned lines.

For a given path P, let A be a flank of LZ(P). If LZ(P) has two open ends, we let α and β be the break points between LZ(P) and its neighboring paths. If, however, LZ(P) has only one open end, then either α or β does not exist and we shall use the terminal of A instead.

Among all the vertices υ with $\alpha \rightarrow \upsilon \rightarrow \beta$, we choose the one with maximal projected distance to the line connecting α and β. This vertex is denoted as v_A. Let B be another flank of LZ(P) and v_B be the vertex similarly defined as v_A. If $pjd(v_A) > pjd(v_B)$, we proceed to work on A. Otherwise, we go on to work on B.

Assuming that $pjd(v_A) > pjd(v_B)$, we thus proceed to work on A. If $pjd(v_A)$ is greater than the width w(P) of P, defined in Subsection 4.3.1, then we subdivide A into two parts. We apply the same subdivision procedure recursively to each of the divided parts. The subdivision procedure is terminated at a part, say S, if the maximal projected distance of the vertices on S to the line connecting the endpoints of S is smaller than w(P).

When we have completed the subdivision procedure on A, we shall construct the thinned line for P in the following way.

For each subdivision point σ, we determine its corresponding point $c(\sigma)$ as below. If σ is a terminal (for example, point 1 in Figure 22), we define $c(\sigma)$ to be the other terminal (possibly σ itself) at the same closed end. If, on the other hand, σ is a break point (for example, point 3 in Figure 22), then $c(\sigma)$ is defined as the break point that pairs with σ. If σ is not a terminal or a break point (point 2 in Figure 22), let τ and υ be two subdivision points neighboring to σ (point 1 and 3 in Figure 22) and L be the line passing through σ and perpendicular to the line connecting τ and υ. Then $c(\sigma)$ is the point at which L intersects with an edge of B.

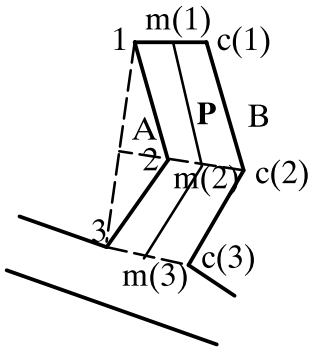

Figure 22. Thinned line constructed for P.

Now, we define m(σ) to be the midpoint between σ and c(σ). For each pair of consecutive points, say σ and τ, we form the line segment connecting m(σ) and m(τ). The union of all such line segments is the thinned line for P.

6. EXPERIMENTAL RESULTS

In the following table, we lay down the average time consumption of our method, applied to two hundred Chinese characters of size 88 × 88 pixels. Following the first line of the table are those that show the time consumption of each module of our method. The time is measured on a SUN SPARC−2 machine at the unit of millisecond.

	Mean	S.D.
Line Sweep Thinning	**71**	**42**
Contour Tracing	20	21
Polygon Approximation	18	8
Path Formation	17	15
Feature Analysis	13	16
Thinning	3	1

It is clear from the above table that there is a decreasing tendency in the time consumption by the involved modules. This result is reasonable, since the number of items worked by each module is reduced, with the help of the previous one, and so is the time consumption. Thus, for example, the first module works on the whole plane figure. The second module works on

the contour extracted by the first one. The third module works on the polygon edges obtained by second one, the fourth on the paths obtained by the third one, etc.

It is worthwhile to compare the thinning method as presented in this article with a pixel−based thinning method. To allow a fair comparison, we let both methods work on the same set of two hundred characters, from which the table in the above for line sweep thinning method is obtained. The result of pixel−based thinning method is shown as below.

	Mean	S.D.
Pixel−Based Thinning	**99**	**17**

Note that the pixel−based thinning method adopted in the comparison consists of a minimal set of operations: (i) they are parallel, in the sense that pixels are examined based on the previous iteration only, (ii) decision on each pixel is based on an efficient table lookup method, and (iii) no pre− or post−processing is included. Such a type of algorithms can be found in, but is not limited to, Stefanelli and Rosenfeld [17]. We also note that although the pixel−based thinning method consumes more time than the line sweep thinning method, it only obtains a skeleton as outcome, while the latter obtains skeleton, paths and intersections sets, all of which are useful for the purpose of character recognition.

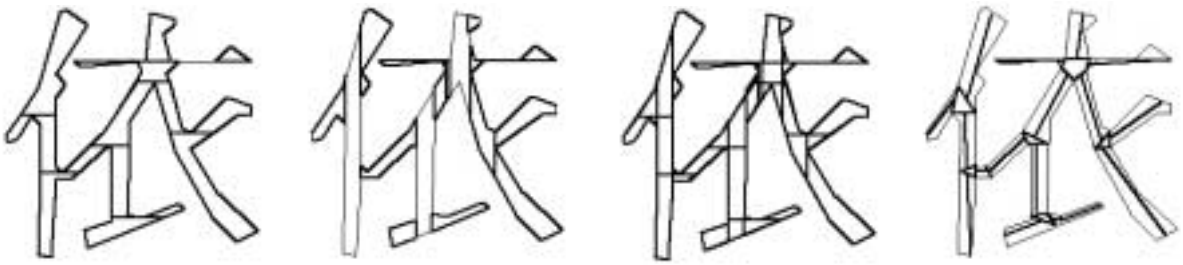

Figure 23. The intermediate and final results obtained from line−sweep thinning algorithm: (a) all H−paths, (b) all V−paths, (c) all extended paths, and (c) skeletons, paths, and intersection sets.

In Figure 23, we display the intermediate and final results of applying the line−sweep thinning algorithm to a Chinese character in Ming Font.

In Figure 24, we display the final results of applying the algorithm to the same Chinese character in three other fonts.

(a) (b) (c)

Figure 24. The final results obtained from the line−sweep thinning algorithm, applied to the same Chinese character in (a) Black Font, (b) Song Font, and (c) Kai Font.

We now comment about the sensitivity of the parameter values used in our algorithm. The first parameter is $\mathrm{Th_H}$ ($\mathrm{Th_V}$), the threshold to regulate the distance between an edge and its H− (V−) counterpart. Here, the value is set as 1/6 width (height) of the enclosing box of a given character. Note that the selection of a larger value would have the effect of creating many paths which will be detected as spurious at a later stage, and is certainly not worthwhile in terms of computational cost. The selection of smaller value, on the other hand, has the effect of eliminating some genuine paths.

The second parameter we employed is $\mathrm{Th_J}$, the threshold to regulate the size of a junction region. Here, again, the selection of a larger value than that chosen by us would be costly in computation, for it sets free the search for intersection sets along pathways which stretch too faraway from the starting point. A selection of smaller value has the devastating effect, since it completely destroies a genuine intersection set and there is no way to restore the set by any other means.

The third and last parameter is the threshold to regulate length−to−width ratio of paths. The determination of the length−to−width ratio is certainly very robust, as one may be convinced by the examples as in 4.3.5. The threshold value, 1/2, as chosen by us is somewhat arbitrary. But it serves the purpose for eliminating spurious paths, because most of those paths have zero or nearly zero length−to−width ratio. Moreover, the arbitrariness of the regulatory

value of length−to−width ratio reflects the arbitrariness of human writings or printed fonts themselves. In fact, in quite many Chinese characters, small pieces of strokes that appear in some fonts, corresponding to our paths with extremely small length−to−width ratio, disappear in some other fonts.

Note that there still exists means to restore some of those paths with too small length−to−width ratio. One possible means is to check the intersection sets comprising the paths in question. If the intersection sets carry certain significant patterns, then we can retain the paths, even though they have below−threshold length−to−width ratio.

Finally, we note that the thinning method as described in this article can serve as the front end to a character recognition system. The outputs of this method can be employed for character recognition in several ways. For example, the types of intersection set, such as hook, end, T shape, cross and corner, can be used as character features (Xie and Suk [12]). On the other hand, the paths as well as their neighboring relations (i.e., the intersection sets in which they participate) can be used as primitives for a recognition system. Examples of such systems are Kahan, Pavlidis and Baird [13], and Cheng, Hsu and Kuo [14]. The paths and intersection sets can also be used as a basis for identifying more complex features in characters. Those features are either single strokes, which are collections of paths extended along a single direction, or sub−character patterns, which are collections of paths extended in various directions. The recognition of characters are then based on a hierarchical matching method (Chang, Cheng and Huang [15]). We also note that this type of thinning algorithm can be also useful in the field of drawing and map processing. For instance, see Sakauchi and Ohsawa [18] and Lu, Ohsawa and Sakauchi [19].

7. CONCLUSION

The line sweep operation provides neat and structural information about the plane figures. The paring of polygon edges, for one thing, can be efficiently derived and easily implemented on this basis. However, the computation of skeletons can not be simply made on these pairs

of edges, because possible conflicts may exist among them. To resolve the conflicts, one can again exploit the structural information acquired from the procedure. Our approach is to look for the set of paths that end at the same intersection region and then to determine the break points between each pair of neighboring paths within this set. This method not only settles the boundary of each path, but also helps to determine whether certain paths should be eliminated. Our experimental results also show that the line sweep thinning algorithm is more efficient in computation than pixel−based thinning method. While the former is able to obtain thinned lines and also to extract intersection sets out of a character figure, the latter requires more amount of computing time to obtain skeletons only.

APPENDIX A

Recall that the crossing of two paths refers to the intersection of their regions. We denote the region of path P as rgn(P).

LEMMA 1. Two paths cross each other if and only if one of them comprises a H−path P and the other a V−path Q such that P and Q cross each other.

Proof. No H−paths can intersect with any H−paths. Neither can V−paths intersect with each other. So when two paths cross each other, one of them must comprise a H−path P which intersects with a V−path Q comprised in the other path. Also, rgn(P) an rgn(Q) can not share any edge, since the regions of the paths in which they are comprised do not share any edge.❑

DEFINITION. Suppose that P is a H−path and Q is a V−path. Then

the left layer of rgn(P) = $\{\pi: \pi$ lies on a left edge of P$\} \cap$ rgn(P),
the right layer of rgn(P) = $\{\pi: \pi$ lies on a right edge of P$\} \cap$ rgn(P),
the upper layer of rgn(Q) = $\{\pi: \pi$ lies on an upper edge of Q$\} \cap$ rgn(Q), and
the lower layer of rgn(Q) = $\{\pi: \pi$ lies on a lower edge of Q$\} \cap$ rgn(Q).

DEFINITION. Suppose that P is a H−path. Let α and β be the first and last point on the left layer of rgn(P), and γ and δ be the first and last point on the right layer of rgn(P). Then the

upper end−line of rgn(P) is defined as the line segment $\overline{\alpha\delta}$, and the lower end−line of rgn(P) is defined as the line segment $\overline{\beta\gamma}$ (Figure 25a).

Similarly, suppose that Q is a V−path, with η and ϕ as the first and last point on the upper layer of rgn(Q) and $\varkappa$ and λ as the first and last point on the lower layer of rgn(Q). Then the left end−line of rgn(Q) = $\overline{\phi\varkappa}$, and the right end−line of rgn(Q) = $\overline{\eta\lambda}$ (Figure 25b).

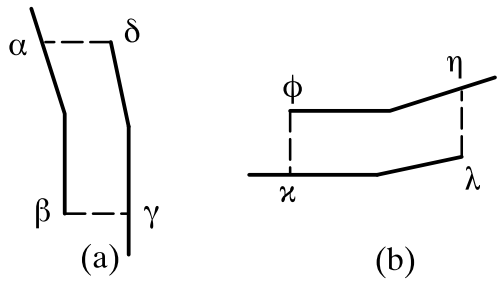

Figure 25. (a) Region of a horizontal paths and its upper and lower end−lines. (b) Region of a vertical path and its left and right end−lines.

Note that both the upper and the lower end−line of a H−path are horizontal line segments. Moreover, all points on the two line segments, except their endpoints, are interior points (namely, points within the character interior). Similar remarks can be made about the left and right end−line of a V−path.

LEMMA 2. Suppose that P is a H−path, Q is a V−path and they cross each other. Let x(.) and y(.) denote the x− and y−coordinate of a give point. Also, let

S_1^p={x(π): π lies on the left layer of rgn(P)},

S_2^p={x(π): π lies on the right layer of rgn(P)},

S_1^q={y(π): π lies on the upper layer of rgn(Q)},

S_2^q={y(π): π lies on the lower layer of rgn(Q)},

$x_1=\max S_1^p$, $x_2=\min S_2^p$, $y_1=\max S_1^q$, and $y_2=\min S_2^q$ (Figure 26a).

Then, $x_1 \le x(\beta) \le x_2$ for every point β in rgn(Q), and $y_1 \le y(\alpha) \le y_2$ for every point α in rgn(P).

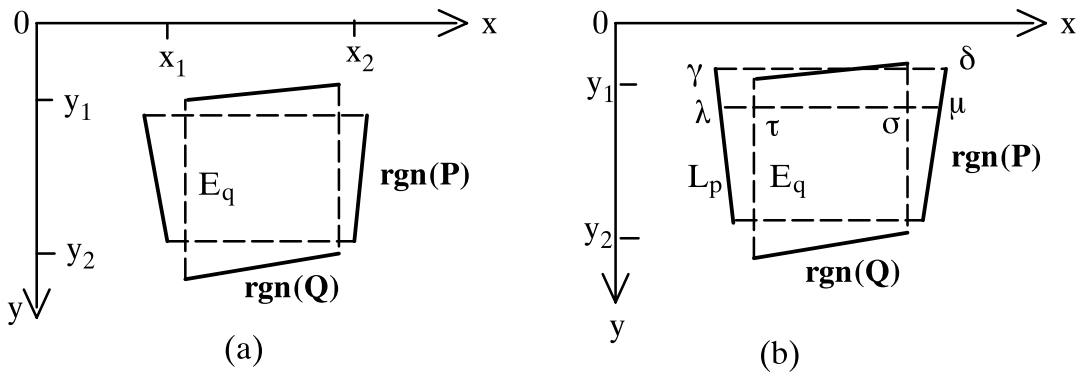

Figure 26. (a) A H−path P crosses a V−path Q. (b) The assumption that rgn(P) contains a point whose height is lower than y_1 can lead to contradiction.

Proof. There are four inequalities to be established. We only prove the following one: $y_1 \leq y(\alpha)$ for every point α in rgn(P). The other three inequalities can be proved similarly.

Assume, on the contrary, that rgn(P) contains a point whose y−coordinate is lower than y_1 (Figure 26b). Then, there must exist a point γ on the left layer of rgn(P) with $y(\gamma)<y_1$. Let δ be the point on the right layer of rgn(P) and is of the same height as γ. Since rgn(P), being the union of horizontal line segments, overlaps with rgn(Q), there must exist a pair of points λ and μ such that λ lies on the left layer of rgn(P), μ on the right layer of rgn(P), λ and μ are of the same height, and $\overline{\lambda\mu}$ intersects with rgn(Q).

Let τ and σ be the leftmost and rightmost point in $\text{rgn}(Q) \cap \overline{\lambda\mu}$. Point τ can not have lower x−coordinate than λ, since this would imply that λ, being a contour point, lies within character interior. Point τ can not locate at the same position as λ either. For if it were, τ would be a contour point in $\text{rgn}(P) \cap \text{rgn}(Q)$. Since rgn(P) and rgn(Q) do not share any edge, and λ is not an endpoint, we reach a contradiction. We thus conclude that τ has higher x−coordinate than λ. Similarly, we can show that σ has lower x−coordinate than μ. In summary, $x(\lambda)<x(\tau)<x(\sigma)<x(\mu)$. This implies that τ and σ are interior points.

Point τ assumes the lowest x−coordinate among all the points of Q that lie on the same horizontal line, and τ is an interior point. Thus, τ must be a point that lies on the left end−line E_q of rgn(Q). On the other hand, λ is a point that lies on the left layer L_p of rgn(P). Since L_p can not intersect with E_q, L_p must always lie on the left−hand side of E_q. Similarly, the right layer

of rgn(P) must always lie on the right−hand side of the right end−line of rgn(Q). These two facts imply that part or all of the upper layer of rgn(P) lies within the area enclosed by: (i) the line segment $\overline{\gamma\delta}$, (ii) the lower end−line of rgn(P), (iii) the left layer of rgn(P) and (iv) the right layer of rgn(P). But this area is part of the character interior. This is an absurd outcome.

Thus, we conclude that the assumption we start with in the Proof is invalid.❑

THEOREM 1. A H−path crosses a V−path, if and only if the end−lines of rgn(P) intersect with those of rgn(Q).

Proof. Using the same notations as in lemma 2, we prove that if P and Q cross each other, then the end−lines of rgn(P) intersect with those of rgn(Q).

First, we note that the left end−line of rgn(Q), being a vertical line segment, has x−coordinate x_L=min{x(β): β in rgn(Q)}, and the right end−line of rgn(Q) has x−coordinate x_R= max{x(β): β in rgn(Q)}. Therefore, we have that $x_1 \le x_L < x_R \le x_2$, by Lemma 2. On the other hand, the left endpoint ξ of the upper end−line of rgn(P) lies on the left layer of rgn(P), while the right endpoint ζ of the upper end−line of rgn(P) lies on the right layer of rgn(P). Thus, we have that $x(\xi) \le x_1 < x_2 \le x(\zeta)$. From these facts, we conclude that $x(\xi) \le x_L < x_R \le x(\zeta)$, or equivalently, the upper end−line of rgn(P) intersects with both the left and the right end−line of rgn(Q). We can similarly prove that the lower end−line of rgn(P) also intersects with the left and the right end−line of rgn(Q).

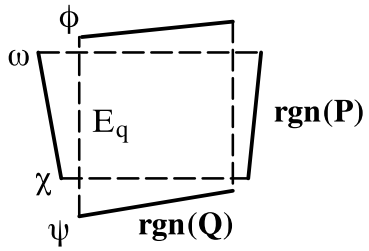

Figure 27. The end−lines of rgn(P) intersect with those of rgn(Q).

Now, we want to prove that if P is a H−path and Q a V−path with the end−lines of rgn(P) intersecting with those of rgn(Q), then P and Q cross each other (Figure 27). Since the two

end−lines of rgn(P) intersect with those of rgn(Q), it follows that rgn(P) overlaps with rgn(Q). We now prove that rgn(P) and rgn(Q) do not share any edge.

Let E_q be the left end−line of rgn(Q). The upper endpoint ϕ of E_q must lie above the upper end−line of rgn(P), and the lower endpoint ψ of E_q must lie below the lower end−line of rgn(P). These facts imply that E_q is not just a single point. It also follows that all points in E_q, except ϕ and ψ, are interior points.

Let ω be the left endpoint of the upper layer of rgn(P), χ the left endpoint of the lower layer of rgn(P). Both ω and χ must lie on the left−hand side of E_q. Therefore, the left layer of rgn(Q), which is a contour curve connecting ω and χ, can not overlap with E_q and must lie on the left−hand side of E_q.

We can similarly prove that the upper layer of rgn(Q) lies above the upper end−line of rgn(P), and the lower layer of rgn(Q) lies below the lower end−line of rgn(P). It follows that the left layer of rgn(P) can not share any edge with both the upper and the lower layer of rgn(Q). We can similarly prove that the right layer of rgn(P) can not share any edge with the upper and the lower layer of rgn(Q).❑

The following proposition can be easily obtained from Jordan Curve Theorem, and is thus stated without proof.

LEMMA 3. When a closed circuit Ψ intersects with another closed circuit Φ at two points α and β, an arc of ψ that connects α and β lies within the interior of Φ (Figure 28).

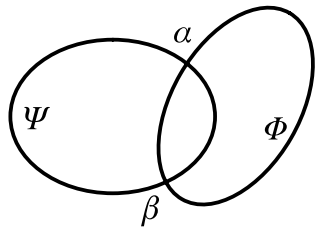

Figure 28. Two closed circuits intersect at two points.

THEOREM 2. If a path crosses another path, it must have two open ends.

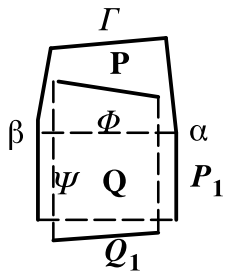

Figure 29. Path P with a closed end crosses path Q.

Proof. Suppose that path P has a closed end and P crosses path Q (Figure 29). By Lemma 1, we can assume that there exists a H−path P_1 comprised in P and a V−path Q_1 comprised in Q such that P_1 and Q_1 cross each other. Let α and β be a pair of points lying on the opposite layers of rgn(P_1) with y(α)=y(β). We label as Γ the part of contour curve that passes the closed end of P and ends at α and β. The curve Γ and line segment $\overline{\alpha\beta}$ forms a closed circuit, and we label it as Φ. We also label as Ψ the closed circuit that encircles rgn(Q_1). Note that Ψ consists of two end−lines and two layers of rgn(Q_1).

By Lemma 2, the line segment $\overline{\alpha\beta}$ intersects two end−lines of rgn(Q_1). The two intersection points are the only points at which Φ intersects ψ, because of the following reasons. (1) Γ, being part of path P which crosses Q, can not intersect any layer of rgn(Q_1). (2) Γ, being part of character contour, can not intersect any end−lines of Q_1, lying within character interior. (3) The line segment $\overline{\alpha\beta}$, being part of character interior except its two endpoints, can not intersect any layer of rgn(Q_1).

But then, by Lemma 3, part of Ψ must lie within the interior of Φ, and this part includes either the upper layer or the lower layer of rgn(Q_1). This is a contradiction, since any layer of rgn(Q_1) is part of character contour and can not possibly lie within the interior of Φ, which is part of character interior.❑

THEOREM 3. If two paths cross each other, then one of them is a H−path and the other a V−path.

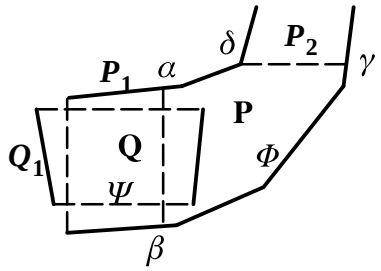

Figure 30. A V−path P_1, comprised in P, crosses a H−path Q_1, comprised in Q. Furthermore P_1 is connected with P_2, also comprised in P.

Proof. Let P and Q be two paths that cross each other (Figure 30). By Lemma 1, we can assume that there is a V−path P_1, comprised in P, that crosses a H−path Q_1, comprised in Q. Suppose that there is, furthermore, a H−path P_2 comprised in P. Since both P_2 and Q_1 are H−paths, they can not have any intersection.

Let α and β be a pair points lying on the opposite layers of P_1 with $x(\alpha)=x(\beta)$, and γ and δ be a pair of points lying on the opposite layers of P_2 with $y(\gamma)=y(\delta)$. Moreover, we assume that α and δ lie on the same flank of P, while β and γ lie on the other flank of P. We label as Φ the closed circuit that comprise the line segment $\overline{\alpha\beta}$, the part of character contour connecting β and γ, the line segment $\overline{\gamma\delta}$, and the part of character contour connecting δ and α. Also, we label as Ψ the closed circuit that encircles $rgn(Q_1)$.

Then, one can easily show, as in the proof for Theorem 2, that Φ and Ψ intersect only at the points where $\overline{\alpha\beta}$ intersect with two end−lines of $rgn(Q_1)$. By Lemma 3, part of Ψ lies within the interior of Φ, and it includes either the right layer or left layer of Q_1, a contradiction. Thus, we conclude that P can not comprise any H−path. By similar argument, we can also show that Q can not comprise any V−path.❑

COROLLARY. Two paths cross each other if and only if both of them have two open ends and one of them is a H−path and the other a V−path and their end−lines intersect with each other.

APPENDIX B

THEOREM. If path P inserts in path Q, then P must have two open ends.

Proof. Let P=⟨A,B⟩. By definition, a flank of P, say A, is a proper subset of Q and another flank, B, of P does not overlap with Q. Suppose that P has a closed end. Then A and B are connected. But then, A is not a proper subset of Q (Figure 31a) or B overlaps with Q (Figure 31b). In either case, we reach a contradiction.❑

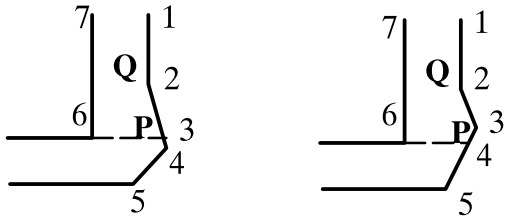

Figure 31. (a) P=⟨A,B⟩=⟨{(2,3),(3,4)},{(4,5)}⟩ and Q=⟨{(1,2),(2,3)},{(6,7)}⟩. (b) P=⟨A,B⟩ =⟨{(2,3)},{(3,4),(4,5)}⟩ and Q=⟨{(1,2),(2,3),(3,4)},{(6,7)}⟩.

APPENDIX C

Recall that the procedure for finding intersection sets starts from an open end of a given path. We refer to such a starting point as a *seed*. When a unitary intersection set has been found from a seed, we apply the same procedure to a new seed, with the restriction that the open ends of paths incorporated in any intersection sets found previously can not be employed in the new intersection set again.

We now stipulate the following assumptions. (A) In a character figure all the conflicts among traversable paths have already been resolved by the operations as described in Subsection 4.2.5. (B) No path embedding in other paths are considered here.

LEMMA. With any open end of any path employed as the first seed, the procedure for finding intersection sets can always succeed at finding an intersection set.

Proof. We prove by induction. Given with a character contour C and two matrices M_H and M_V, specifying the mutually visibility between edges of C, we let S={j: j is an edge on C for which

there is some edge k on C with $M_H[k][j]=1\}$. We also let num(C,M_H,M_V) be the total number of edges in S. Suppose that the proposition is true for every triple ⟨C,M_H, M_V⟩ with num(C,M_H,M_V)<N. We now prove that it is also true for every triple with num$(\cdot,\cdot,\cdot)$=N.

So, let a triple ⟨C,M_H,M_V⟩ with num$(\cdot,\cdot,\cdot)$=N be given. Let P be a path with open ends. Moreover, let $\overline{M}_H$ and $\overline{M}_V$ be the matrices which are obtained from M_H and M_V by resetting $\overline{M}_H[m][n]=0$ and $\overline{M}_V[n][m]=0$ for those m and n contained in opposite flanks of P. We now consider the following two cases.

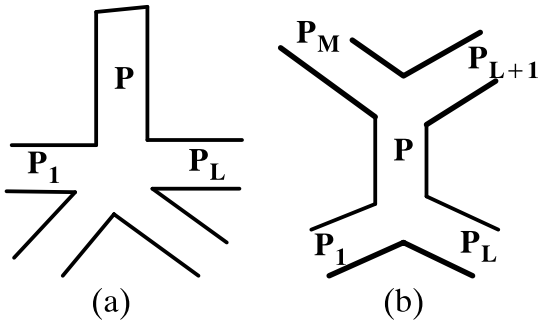

Figure 32. (a) Path P has only one open end. (b) P has two open ends.

Case 1. P has only one open end (Figure 32a).

Since num(C,$\overline{M}_H$,$\overline{M}_V$)<N, it follows from our induction hypothesis that there exists, with respect to ⟨C,$\overline{M}_H$,$\overline{M}_V$⟩, an intersection set IS_1 comprising P_1, ..., P_L., with a flank of P_L connected with that of P_1 via the edges of P. So we must have an intersection set, with respect to the original triple ⟨C,M_H,M_V⟩, comprising P, P_1, ..., P_L.

Case 2. P has two open ends (Figure 32b).

Again, we construct a new triple ⟨C,$\overline{M}_H$,$\overline{M}_V$⟩, in which $\overline{M}_H[m][n]=0$ and $\overline{M}_V[n][m]=0$ for those m and n contained in opposite flanks of P. Then we can show similarly as in Case 1 that there exists, with respect to ⟨C,$\overline{M}_H$,$\overline{M}_V$⟩, an intersection set consisting of P_1, P_2, ..., P_L, P_{L+1}, ..., P_M, such that P_1 and P_M are connected via the edges on one flank of P, while P_L and P_{L+1} are connected via the edges on another flank of P. It follows that, with respect to the triple ⟨C,M_H,M_V⟩, P, P_1, ..., P_L compose an intersection set, and P, P_{L+1}, ..., P_M compose another intersection set.❑

If a path P has some unmarked edges, namely, edges not employed in any intersection sets found previously by the procedure, then every open end of P can be used as a new seed. We now prove the following theorem.

THEOREM. With an open end of a path employed as a new seed, the procedure for finding intersection sets can always succeed at finding an intersection set

Proof. Let the pair (P, e_p) be a given seed. By the above lemma, there exists an intersection set IS, with (P, e_p) employed as the first seed. If every entry in IS participates in only one intersection set, then IS must be the intersection set found by the procedure, even when (P, e_p) is not the first seed ever employed.

So we assume that there exists a pair (R, e_r) in IS which is comprised in a intersection set, say IS_1, found previously by the procedure. Let W be a path which is not in IS but is a neighboring path of R in IS_1 (Figure 33a). We now state and sketch the Proof for the following listed statements.

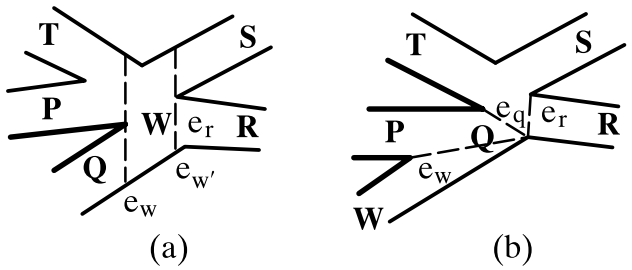

Figure 33. Intersection set IS comprises paths P, Q, R, S and T. (a) W and S are comprised in another intersection set. (b) W has some edges lying outside of the contour curve of IS.

(1) The edges of W must completely lie within the contour curve C of IS.

Suppose W contains some edge lying outside C. The only possibility for this to occur, while assumption (A) is not violated, is that W overlaps with some path, say Q, in IS (Figure 33b). In this case, the intersection set IS_1 must also comprise (P, e_p). But P contains some unmarked edges, a contradiction.

(2) W must have two open ends, say e_w and $e_{w'}$.

Suppose that W has a closed end. Then W has a single connected component. Let Q be the neighboring path of R in IS such that a flank of Q is connected with the flank of R via some edges of W. In order for a flank of Q to get connected with a flank of R, it must comprise all edges of W, since W locates between the two flanks. This means that W embeds in Q, violating assumption (B).

(3) If $(W, e_{w'})$ is comprised in IS_1, then (W, e_w) is comprised in another intersection set, say IS_2, which also comprises (P, e_p) (Figure 33a).

Suppose that $(W, e_{w'})$ is comprised in IS_1. There must exist a pair (Q, e_q) in IS such that Q is a neighboring path of R and Q overlaps with W. Moreover, since (R, e_r) is connected to $(W, e_{w'})$, (Q, e_q) must be connected to (W, e_w), at an endpoint of e_w. Similarly, there must also exist a pair (T, e_t) in IS which is connected to (W, e_w), at the other endpoint of e_w. Then, all the pairs from (P, e_p) to (Q, e_q), followed by (W, e_w) and all the pairs from (T, e_t) back to (P, e_p) would form an intersection set. This intersection set, denoted as IS_2, is called a *derived* intersection set.

(4) From IS_2, one can obtain an intersection set which comprises (P, e_p) and does not contain any marked paths, namely, paths which are comprised in an intersection set found previously by the procedure.

Suppose that IS_2 is not itself the desired intersection set, namely, it does contain any unmarked path. Then, by repeating the same process, one can obtain a derived intersection set IS_3 from IS_2, also comprising (P, e_p). Moreover, IS_3 comprises fewer number of paths originally in IS than IS_2. Thus, by repeating the same process no more than N times, where N is the total number of paths comprised in IS, one can obtain a minimal intersection set IS_m comprising (P, e_p). IS_m can not contain any marked path, for if it does, then (P, e_p) must have been marked, a contradiction.❑

REFERENCES

[1] L. Lam, S. W. Lee, and C. Y. Suen, "Thinning Methodology − a comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell*., Vol. PAMI−14, pp. 869−885, 1992.
[2] T. Pavlidis, "A Vectorizer and Feature Extractor for Document Recognition," *CVGIP*, Vol. 35, pp. 112−127, 1986.
[3] S. A. Mahmoud, I. A. Haiba, and R. J. Green, "Skeletonization of Arabic Characters Using Clustering Based Skeletonization Algorithm (CBSA)," *Pattern Recognition*, Vol. 24, No. 5, pp. 453−464, 1991.
[4] B. Li and C. Y. Suen, "A Knowledge−Based Thinning Algorithm," *Pattern Recognition*, Vol. 24, No. 12, pp. 1211−1121, 1991.
[5] H. Nishida, T. Suzuki, and S. Mori, "Thin Line Representation from Contour Representation of Handprinted Characters," *Proc. International Workshop on Frontiers in Handwriting Recognition,* Bonas, France, pp. 27−68, September 23−27, 1991.
[6] C. Chouinard and R. Plamondon, "Thinning and Segmenting Handwritten Characters by Line Following," *Machine Vision and Applications*, Vol. 5, pp. 185−197.
[7] F. P. Preparata and A M. Shamos, *Computational Geometry: An Introduction*, Springer Verlag, New York, 1985.
[8] T. Pavlidis, "Scan Conversion of Regions Bounded by Parabolic Splines," *IEEE Computer Graphics and Applications*, Vol. 5, pp. 47−53, 1985.
[9] F. Chang, Y. C. Chen, H. S. Don, W. L. Hsu, and C. I. Kao, "Stroke segmentation as a basis for structural matching of Chinese characters," *Proc. 2nd Int. Conf. Document Anal. and Recogn*., Sukuba, Japan, pp. 35−40, 1993.
[10] F. Chang and Y. P. Cheng, "Graphical representation of characters via decomposition method," *Proc. 1994 Int. Conf. on Compt. Proc. of Oriental Languages,* Korea, pp. 435−440, 1994.
[11] F. Chang, Y. P. Cheng, T. Pavlidis and T. Y. Shuai, "A Line Sweep Thinning Algorithm," *Proc. 3rd Int. Conf. Document Anal. and Recogn*., Montreal, Canada, pp. 227−230, 1995.
[12] S. L. Xie and M. Suk, "On Machine Recognition of Hand−Printed Chinese Characters by Feature Relaxation," Pattern Recognition, Vol. 21, No. 1, pp. 1−7, 1988.
[13] S. Kahan, T. Pavlidis, and H. S. Baird, "Building a Font and Size Independent Character Recognition System," *IEEE Trans. Patt. Anal. Machine Intell*., Vol. PAMI−9, pp. 274−288, 1987.
[14] F. H. Cheng, W. H. Hsu and M. C. Kuo, "Recognition of Handprinted Chinese Characters via Stroke Relaxation," Pattern Recognition, Vol. 26, No. 4, pp. 579−593, 1993.
[15] F. Chang, Y. P. Cheng, and Y. S. Huang, "Hierarchical Flexible Matching for Recognition of Chinese Characters," *Proc. Third Int. Conf. Document Anal. and Recogn.*, Montreal, 1995.
[16] F. Chang, Y. C. Lu, and T. Pavlidis, "Line Sweep Thinning Algorithm: A Complete Techni-

cal Report, TR−IIS−97−011, Institute of Information Science, Academia Sinica, Taipei, Taiwan, 1997.
[17] R. Stefanelli and A. Rosenfeld, "Some Parallel Thinning Algorithms for Digital Pictures," *Journal of ACM*, Vol. 18, No. 2, pp. 255−264, 1971.
[18] M. Sakauchi and Y. Ohsawa, "The AI−MUDAMS: the Drawing Processor Based on the Multidimensional Pattern Data Structure," IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, pp. 154−161, 1985.
[19] W. Lu, Y. Ohsawa, and M. Sakauchi, "A Database Capture System for Mechanical Drawing Using an Efficient Multi−Dimensional Graphical Data Structure," Proc. 9th ICPR, pp. 266−269, 1988.