Short Papers

Singularities and Regularities on Line Pictures via Symmetrical Trapezoids

Jairo Rocha and Rafael Bernardino

Abstract—An algorithm that decomposes a line image into singular and regular regions is presented. We define a contour trapezoid as the building block of regular regions, and postulate a Maximal Trapezoid Set as the core of regular regions. Then, we describe an algorithm that calculates a Maximal Trapezoid Set of a polygon and show how to use it to find a skeleton of a polygonal approximation of a contour. Experiments are explained to show the behavior of the new concept on real images as compared to previous algorithms.

Index Terms—Contour trapezoid, maximal trapezoid set, regular region, singular region, skeleton.

1 Introduction

THERE are hundreds of thinning algorithms that have been proposed in the literature [3]. However, just in the last decade, some researchers have noticed the importance of dividing line figures into regular and singular regions [4]. Regular regions correspond to the ones surrounded by smooth not crossing quasi-parallel contour arcs. They are the parts of the strokes not affected by crossings or branches. The other regions, the ones on crossings or endings, or near branches or nonsmooth contour lines are the singular regions. Most skeletonization methods give similar results on regular regions and different results on singular regions.

Regular regions are extracted by matching of opposite contour pixels by [7]. In their algorithm, hereafter, *SM*, a pixel on one side of the contour is matched to a pixel on the other side of the contour using only the similarity between the local tangents: The difference on their tangent angles must be less than a threshold. For real images, the threshold should be set relatively high, otherwise, there are lots of contour parts that are not found opposite, due to noise, or because opposite contour parts can deviate a lot from being parallel. Therefore, great flexibility for matching is allowed and no criterion is given for deciding among possible opposite pixels.

We define in this paper a new paradigm which gives formal support not only to the way we find *opposite segments* but to algorithm *SM*. We are able to consider several opposite segments to a segment and use them to get a better decision, and take the "best" one according to some well-defined global criterion.

Chang et al. [2] also find regular and singular regions from polygonal contours. Their line sweep method finds efficiently opposite segments, but it has cases that are conceptually complex, and they have not proved that their method keeps the rotational invariance.

The final result of our method is a skeleton. Skeletons are graphs. Regular strokes are represented as polylines and correspond to graph edges, and singular regions are graph joints. Following a representation similar to the concept of *link* in [1], edges carry the geometrical information of the length and orientation of true strokes, and joints carry the logical information of adjacency of strokes but extension of the strokes is not performed, so no real geometrical connection of segments is forced. This separation between logical and geometrical connections makes it possible to avoid the skeletonization of regions that actually do not have a natural skeleton. Thus most common skeleton deformations are avoided.

• *The authors are with the Departament de Matematiques i Informatica, Uniersity of the Balearic Islands, 07071 Palma de Mallorca, Spain. E-mail: jairo@ipc4.uib.es.*

Manuscript received 17 Feb. 1997; revised 6 Mar. 1998. Recommended for acceptance by M. Mohiuddin.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 106567.

2 Basic Definitions

Assume that we have the contour of a line region approximated by polyline segments using a split-and-merge algorithm. Notice that some information is lost from a detailed contour that could be extracted from gray-level images or binary images.

We call a trapezoid *interior* if it lies completely inside the region defined by the contour. In other words, it does not include points outside the region. We will work only with interior trapezoids. A trapezoid is allowed to be reduced into a triangle.

DEFINITION 1 (Regular and Contour Trapezoids). *A trapezoid is* regular *if it is symmetrical with respect to an axis perpendicular to its parallel edges. Therefore, it has two edges that are parallel and the other two have equal length.*

A contour trapezoid *is a regular trapezoid such that its two edges of equal length lie on the contour of the region. These two edges are called* contour edges.

Hereafter, all trapezoids are assumed to be interior contour trapezoids. Fig. 1a shows some trapezoids of a region. Given a trapezoid, its *bisectrix* is the segment that divides the regular trapezoid into two symmetrical parts.

Our basic assumption is that a trapezoid is a building block for regular regions. Its symmetry is a guarantee of contour regularity, and, therefore, its bisectrix is a natural part of the skeleton. Thus, to find all trapezoids, it is on the way of finding the skeleton.

DEFINITION 2 (Compatible Trapezoids). *Two trapezoids are* compatible *if their intersection area is null.*

We consider sets of trapezoids that are pair-wise compatible. Let us consider two noncompatible trapezoids. There are two cases: Either they share a contour part or not. In the first case, alternative opposite segments for the contour part are suggested, as shown in Fig. 1c. In the second case, we say that they *cross* each other. See Fig. 1b.

As it can be easily seen, there are lots of trapezoid sets for a given region, but perceptually we would prefer some of them. The *trapezoid length* is the length of its bisectrix. The *trapezoid width* is the average length of the parallel sides of the trapezoid.

DEFINITION 3 (Degree of Opposition). *The* degree of opposition *of a trapezoid is the cosine of the angle between the lines on which their contour edges lie.*

The higher the degree of opposition, the more parallel the contour edges are.

DEFINITION 4 (Preference). *Consider two noncompatible trapezoids t_1 and t_2. $t_1 \preceq_w t_2$ if their contour sides do not intersect, and, t_1 has a larger or equal width than t_2. $t_1 \preceq_{\|} t_2$ if their contour sides intersect and t_2 has a higher or equal degree of opposition than t_1. In both cases, it is said that t_2 is preferred to t_1.*

The reason behind this preference is perceptual. For example, consider that the region of analysis is a rectangle. Its "best" skeleton is its longer symmetrical axis. This is the reason why the longer bisectrix is taken. In the case of trapezoids that share parts of the

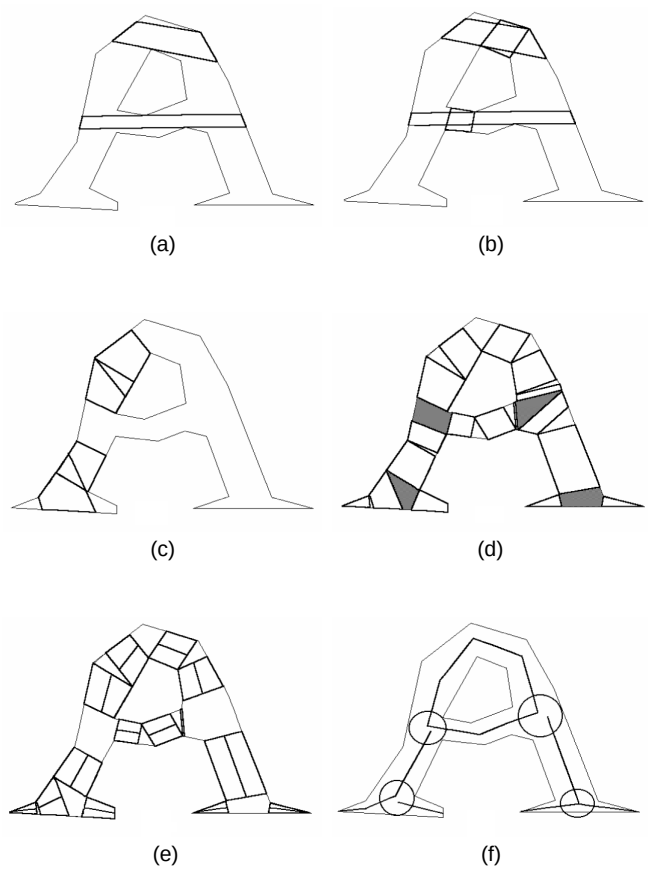

Fig. 1. Process to get regular and singular regions: some interior trapezoids (a) that are eliminated by crossing trapezoids (b); some incompatible trapezoids (c) will appear shortened on a MTS (d), which shows singular regions in dark. Bisectrices are basic skeleton lines (e), and the final skeleton with node graphs shown by circles (f).

contour, it is postulated that the one with greater edge opposition should be preferred.

3 Algorithm for Maximal Trapezoid Sets

The main steps of an algorithm to find a *Maximal Trapezoid Set* (*MTS*), in the sense that is defined at the end of this section, are: Starting from the polygonal segments, *opposite* segments are found. Then, all trapezoids defined by a pair of opposite segments are computed. Finally, trapezoid crossings and noncompatibilities are eliminated in the order given by the preference relation.

Given two segments, their *bisectrix segment* is the intersection of the two orthogonal projections of the segments over their bisectrix line. If segments are parallel, their bisectrix is the middle line parallel to the segments.

DEFINITION 5 (Opposite Segments). *Two segments are* opposite *if the angle between them is less than 90 degrees and the bisectrix segment between them is not empty.*

Two opposite segments define zero, one, or more interior trapezoids, according to the holes in the region. Once all interior trapezoids are found, the noncompatible ones are eliminated in the following order. If two trapezoids do not share contour parts and intersect their areas, e.g., they cross each other, we favor the one with less width. Then, the second type of incompatibilities is eliminated by adding subtrapezoids with the highest degree of opposition and as big as possible. Fig. 2 describes the algorithm.

Now, we define formally the set we have calculated. Consider the set of all trapezoids inside a region of analysis. We define a

```
Algorithm Maximal-Trapezoid-Set
Input = P, contour polygons
Output = T', a trapezoid set
for each pair of segments of P do
       if the segments are opposite then
       generate all its interior trapezoids.
end-for
Sort the trapezoids upwards by width.
Assign empty to T.
for each trapezoid t do
       Add t to T if it does not cross anyone in T.
end-for
Sort T by degree of opposition.
Assign empty to T'.
for each trapezoid t do
       Add to T' the largest subtrapezoid of t that is
       compatible with T'.
end-for
end-Algorithm
```

Fig. 2. Algorithm to find Maximal Trapezoid Sets.

maximal set T of noncrossing trapezoids under the $\preceq_w$ relation, in the following sense.

DEFINITION 6 ($\preceq_w$-Maximality). *A trapezoid set T is $\preceq_w$-Maximal if no two trapezoids cross each other and for any other trapezoid set S, and for each trapezoid $t_1 \in S$, there is a trapezoid t_2 of T such that $t_1 \preceq_w t_2$ or $t_1 \subseteq t_2$.*

Once we have found a $\preceq_w$-Maximal set T, we work only with subtrapezoids of trapezoids of T. When a trapezoid set is made of subtrapezoids of T, we say that it is *relative to* T. A trapezoid set relative to T does not contain trapezoids that cross each other. We consider a compatible trapezoid set relative to T that is maximal under the $\preceq_{||}$ relation, for all trapezoid sets relative to T.

DEFINITION 7 ($\preceq_{||}$-Maximality). *A trapezoid set T' relative to T is $\preceq_{||}$-Maximal if it is a set of compatible trapezoids and for any other trapezoid set S relative to T and for each trapezoid $t_1 \in S$ there is a trapezoid t_2 of T' such that $t_1 \preceq_{||} t_2$ or $t_1 \subseteq t_2$.*

Although the relations defined above are not orders, we are interested in finding compatible trapezoid sets that are maximal in the sense defined before. In this way, we cover the region as much as it is possible, in order to find the skeleton. The proposition below shows that the algorithm finds what we wanted.

PROPOSITION 1 (Correctness). *The algorithm* Maximal-Trapezoid-Set *calculates a $\preceq_{||}$-maximal trapezoid set relative to a $\preceq_w$-maximal trapezoid set.*

PROOF. To show that T in Fig. 2 is $\preceq_w$-maximal, let S be a trapezoid set and $t_1 \in S$. Consider the two contour segments over which the trapezoid contour edges lie. Since the algorithm finds all trapezoids generated for every pair of segments, there is a trapezoid $t_2 \supseteq t_1$, for the opposite segments. If $t_2 \in T$, then we are done. Otherwise, t_2 was removed because it intersected a trapezoid t_3 that was already in T and $t_2 \preceq_w t_3$. Notice that t_1 has the same width of t_2.

Similarly, to prove that T' in Fig. 2 is $\preceq_{||}$-maximal, let S be a trapezoid set relative to T and $t_1 \in S$. Then, $t_1 \subseteq t_2$ for some $t_2 \in T$. If $t_2 \in T'$, then we have proven what we wanted. Otherwise, t_2 was removed because it intersected a trapezoid t_3 that was already in T' and $t_2 \preceq_{||} t_3$. Let $t_4 \subseteq t_2$ the largest subtrapezoid

added to T' that keeps the compatibility. If $t_1 \subseteq t_4$, we are done again. Otherwise, if $t_1 \cap t_3 = \emptyset$, then $(t_1 \cup t_4) \cap t_3 = \emptyset$, and therefore t_4 would not be the largest because $t_1 \cup t_4$ would be larger. Thus, $t_1 \cap t_3 \neq \emptyset$, and since t_1 and t_2 have the same degree of opposition, $t_1 \preceq_{||} t_3$. This yields the result. □

3.1 Complexity Analysis

Suppose we have n contour segments and B black pixels. To get the interior trapezoids of two segments, only black pixels are visited (it is not a tight upper bound). Hence, to get the interior trapezoids, it takes $O(Bn^2)$ steps. If T is the number of interior trapezoids, to generate the nonintersecting set takes $O(T^2)$ steps which is greater than what is needed to sort them twice. It is possible to find the final compatible trapezoids in $O(T)$ steps, because the incompatible ones to a given trapezoid are on its contour segments. Therefore, the overall complexity is $O(Bn^2) + O(T^2)$. The complexity is rather big, because this implementation uses the region pixels to test for visibility. It was found experimentally that the number T of trapezoids has a lineal dependency on the number n of segments. Also, the average for n was 51.33 and for T was 65.73 on the images of 100 Chinese characters. See the computer time used in the results section.

4 Skeleton

The purpose of this section is to describe how trapezoids are used to find regular and singular regions, and then, a skeleton.

Given a polygon, possibly with polygonal holes, we can find a maximal trapezoid set. Sequences of adjacent trapezoids (see, for example, Fig. 1d) form regular regions. All bisectrix segments associated with trapezoids are part of the skeleton. These segments should be elongated or removed from the skeleton near corners to improve their estimated stroke length, and we follow the same strategy that algorithm *SM* has. The only difference is that crossings—regions where several regular regions meet—are not replaced by a point, because it makes segments around a joint to deform unnaturally to force the connection. Instead, we extend the skeleton segments adjacent to a singular region without changing its direction, and no geometrical connection is found. Only logical adjacency of segments is stored in the relational skeleton graph.

5 Results

We have devised three methods to test the new algorithm. First at all, we compared it to the cross sections sequence method [7] on 100 Chinese characters. We call this algorithm *SM*, as stated before. Second, a recognition algorithm uses the new skeletons and its performance is compared to the case where *SM* algorithm is used. Third, we compared to the *reference skeletons* of several Latin letters that according to [5] should be used for testing new skeletonization algorithms; due to space limitations, the results on Latin letters are not included but are available upon request.

5.1 On Chinese Characters

Several printed characters were processed taking advantage of their complex structures in order to test the algorithm against its ancestor, *SM*. Both algorithms were implemented in C and applied to 100 characters. Fig. 4, Fig. 6, and Fig. 7 show the results for six characters that represent most of the differences. The left columns display the results for the previous algorithm and the right ones, for the new algorithm. On the images, both algorithms produce graphs as output, hence not only geometrical but also *logical* information is generated. Segments represent skeletons of regular regions, and circles are graph nodes situated on singular regions, endings, and corners, and their diameter represents the size of the singular region; both algorithms know the exact extent of singular regions, but for simplicity, only circles are displayed. On junctions or crossings, there is always a graph node that logically connects strokes, and it is represented by the circle. The threshold for the angle of opposition is 60 degrees (e.g., to allow matching, the maximum angle difference between two contour pixel directions is 60 degrees) for *SM* algorithm. Also, this algorithm displays a polygonal approximation of the skeleton, while the new algorithm displays the extended bisectrices, as explained above.

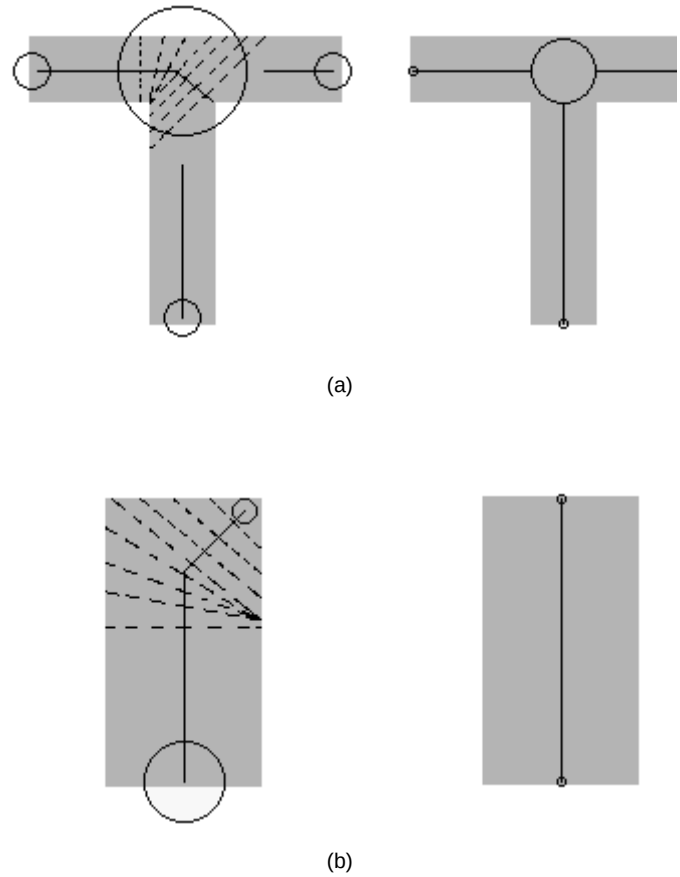

Fig. 3. Artificial images.

The new algorithm takes 0.32 second per character in the average, while *SM* takes 0.20 second on a Sparc IPX. This is due to the fact that a straightforward implementation of the method was done, instead of using efficient data structures to manage segments on the plane.

First at all, let us show two artificial images of skeletons without extension. Fig. 3a and Fig. 3b (left) show that *SM* algorithm with the threshold for the angle of opposition set to 60 degrees produces distortions on skeletons. The reason is that pixels on the corners can match, nonintuitively and within the threshold, pixels on the opposite side (dash lines show the pixel matching). When the threshold is reduced from 60 to 40 degrees, both algorithms have the same result, but *SM* algorithm misses lots of opposite contour parts, due to noise, if this setting is used for the Kanji characters. These examples show that the *SM* algorithm needs parameter adjustment to get its best results, while the new algorithm has no parameters.

On the Kanji characters, when looking at the images, the first impression is the similarity of the results: Both algorithms are concentrated on finding regular regions. As a result, crossings and T

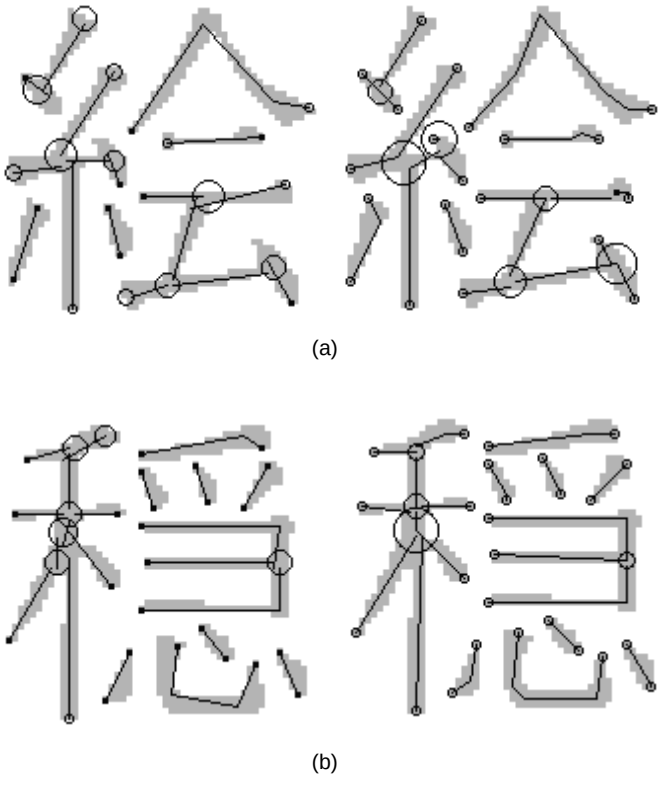

Fig. 4. Tests on Chinese characters.

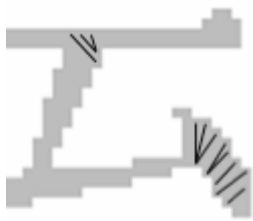

Fig. 5. Part of Fig. 4a showing how the Suzuki and Mori algorithm does the matching.

junctions were analyzed well, avoiding spurious strokes. Below, we discuss the differences. For the character in Fig. 4a, there is one stroke on the bottom-right corner of the image that forms a T junction with a longer, almost horizontal stroke (see Fig. 5). Its skeleton has only one segment below the singular region, on the left image. On the right image, it has two segments one on each side of the singular region (T junction). The new algorithm finds one trapezoid on each side of the T junction; each side of the trapezoid matches to its best opposite segment. Algorithm *SM* matches contour pixels from below the T junction to pixels above the T junction, because it starts matching pixels near the lower end of the stroke and expands the matching too much, even between pixels almost on top of the stroke with pixels just below the junction. In fact, it never checks that there are better pixels to match the top part of the stroke. In short, *SM* matches pixels that are opposite enough, while the new algorithm matches segments to its best possible opposite.

In Fig. 5, and in the same region of (a), there is a T junction between a horizontal stroke and an almost vertical one. Notice how the new algorithm totally avoids the distortion, while *SM* matches pixels from the horizontal contour to pixels of the tilted contour.

Again in Fig. 4a, on the left part of the character, the horizontal bar is distorted by the new algorithm, due to polygonal approximation errors. However, observe how well branches are now detected.

In Fig. 4b, on the left part of the character, there is a double branching of two tilted strokes from the vertical stroke. *SM* gener-

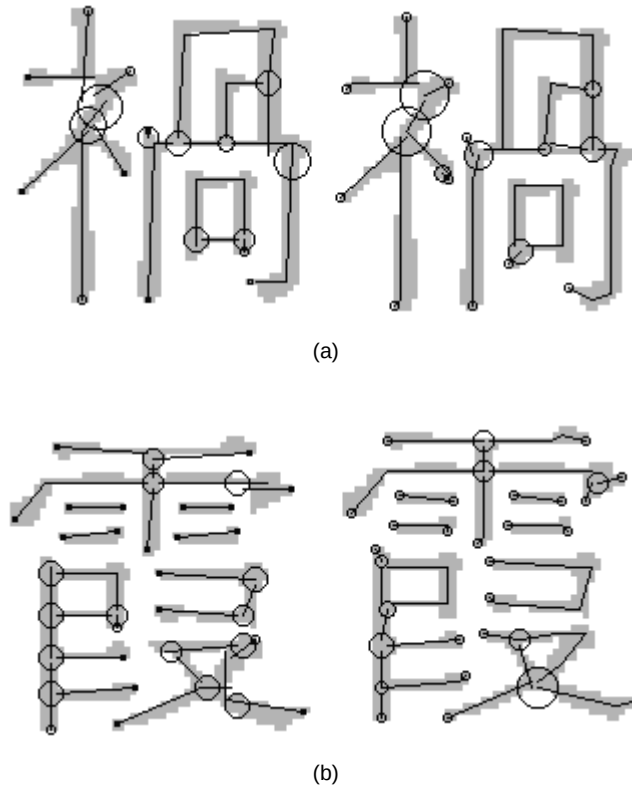

Fig. 6. Tests on Chinese characters (continued).

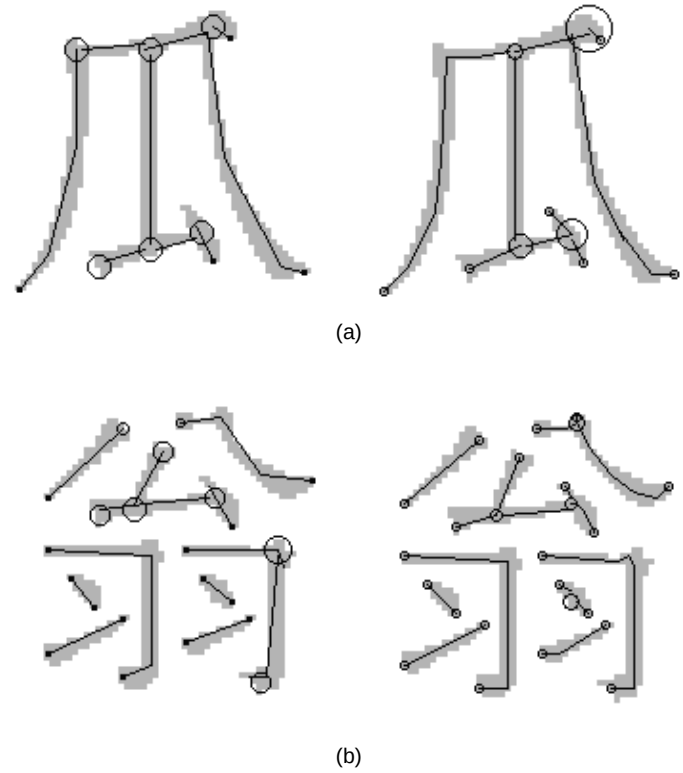

Fig. 7. Tests on Chinese characters (continued).

ates two singular regions and one spurious "neck" between them, while the new algorithm connects the two vertical strokes and the two tilted strokes in only one singular region. The reason for this lies on a better global analysis done by the new algorithm, due to the polygonal approximation. *SM* finds the orientation of the spurious stroke from two pixels on both sides. In general, the new algorithm is more robust to noise, however, it is also true that the polygonal approximation is not the original image.

In Fig. 6a, in general, all segments and their connections are better placed for recognition by the new algorithm. In particular, branches on the left part of the character are better placed. We will see the effect of the better placement of nodes and estimation of lengths on recognition in the next subsection.

In Fig. 6b, the structure on the bottom-right part is better analyzed by the new algorithm: The crossing strokes are generated without the spurious branch, due to a better estimation of the angles. The *SM* algorithm is confused by noise, while the polygonal approximation is more robust.

Fig. 7a and Fig. 7b each show a better T-branch analysis for the same reason as in Fig. 4a: In the new algorithm, each region of the contour is matched to its best opposite. Observe also the better T-junction on the upper part of Fig. 7a between a horizontal stroke and an almost vertical stroke under it: There is no deformation near the branch.

5.2 Substructure Recognition

An algorithm was developed [6] to recognize 10 basic character components (substructures) of any size anywhere on a Kanji string, even if they touched other components. The algorithm receives as input a skeleton graph and performs stroke grouping, indexing of structural features on a previously generated look-up table, structural verification of hypothesis using model graphs, and geometrical verification by an array of neural nets, each one specialized on the geometry of each model. Using the output of the *SM* algorithm, it retrieves 98 percent of substructures with 92 percent precision rate; this means that the system retrieved 674 substructures from 100 Chinese characters but failed to retrieve 13 substructures (98 percent retrieval rate); of the 674 substructures retrieved, 55 were wrong on 18 characters, in the sense that their shape did not correspond to the suggested model (92 percent precision rate). The most damaging errors are the retrieval failures, because subsequent parts of the system will use the presence or absence of a substructure for final recognition of a full character, but they will not generate new structures. The causes of this type of errors are, in order of importance:

- skeleton generation errors, 31 percent (especially, loss of small segments and direction change);
- connectivity requirement error, 31 percent (paths are not connected as expected);
- neural net errors, 23 percent (the net rejects the generated shape instance); and
- collinearity grouping errors, 15 percent (paths are counter-intuitive due to local decisions).

Using the new skeletons, the system solved 50 percent of the errors caused by bad formed skeletons, so it is now the least important cause of errors, and the retrieval rate improved almost half a percent, while the precision rate was the same.

In short, the new algorithm is more useful than the previous one for Kanji character analysis.

6 Conclusions

We have defined Maximal Trapezoid Sets as a new paradigm to find regular and singular regions. We have also explained an algorithm to calculate them, and we were successful at showing its usefulness to find skeletons.

This new paradigm gives formal support to more algorithmic definitions of regular regions such as the one given in [7]. However, the algorithm explained here should be improved for efficiency by using data structures to handle segments on the plane.

The results on real images are very useful for recognition, especially for their complete logical information attached to nondistorted geometrical values.

Acknowledgments

We would like to thank Prof. Pavlidis for suggesting the problem and Dr. Wacef Guerfali and Dr. Marc Bourdeau from the Ecole Polytechnique de Montréal for supplying the digital images of the reference skeletons.

References

[1] C. Bjorklund and T. Pavlidis, "Global Shape Analysis by k-Syntactic Similarity," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 3, no. 2, pp. 144-155, 1981.

[2] F. Chang, Y.C. Lu, and T. Pavlidis, "Line Sweep Thinning Algorithm for Feature Analysis," *Fourth ICDAR*, pp. 123-127, Ulm, Germany, 1997.

[3] L. Lam and C.Y. Suen, "Thinning Methodologies: A Comprehensive Survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 9, pp. 869-885, Sept. 1992.

[4] T. Pavlidis and D. Lee, "Residual Analysis for Feature Extraction," J.C. Simon, ed., *From Pixels to Features*. New York: North-Holland, 1989, pp. 219-227.

[5] R. Plamondon, C. Suen, M. Bourdeau, and C. Barriere, "Methodologies for Evaluating Thinning Algorithms for Character Recognition," *Int'l J. PRAI*, vol. 7, no. 5, pp. 1,247-1,270, 1993.

[6] J. Rocha and H. Fujisawa, "Substructure Shape Analysis for Kanji Character Recognition," *Advances in Structural and Syntactic Pattern Recognition*, vol. LNCS 1,121. New York: Springer, pp. 361-370, 1996.

[7] T. Suzuki and S. Mori, "Structural Description of Line Images by the Cross Section Sequence Graph," *Int'l J. PRAI*, vol. 7, no. 5, pp. 1,055-1,076, 1993.