SHAPE, STRUCTURE AND PATTERN RECOGNITION

Nahariya, Israel 4–6 October 1994

Edited by

Dov Dori

Faculty of Industrial Engineering & Management
Technion, Israel Institute of Technology
Technion City, Haifa 32000, Israel

Alfred Bruckstein

Faculty of Computer Science
Technion, Israel Institute of Technology
Technion City, Haifa 32000, Israel

World Scientific
Singapore • New Jersey • London • Hong Kong

Published by

World Scientific Publishing Co. Pte. Ltd.

P O Box 128, Farrer Road, Singapore 9128

USA office: Suite 1B, 1060 Main Street, River Edge, NJ 07661

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

SHAPE, STRUCTURE AND PATTERN RECOGNITION

ISBN 981-02-2239-4

Printed in Singapore by Uto-Print

FINDING AND RANKING BASIC STRUCTURES ON COMPLEX LINE PATTERNS

Sergey V. ABLAMEYKO[+], Carlo ARCELLI[*], Gabriella SANNITI DI BAJA[*]
[+]*Institute of Engineering Cybernetics, Belarusian Academy of Sciences, Minsk, Belarus*
[*]*Istituto di Cibernetica, Consiglio Nazionale delle Ricerche, 80072 Arco Felice, Napoli, Italy*

ABSTRACT

The skeleton of a digital shape is a line pattern consisting of two basic structures, the branches and the loops, which are interleaved through the branch points. An iterative parallel algorithm is illustrated to find these basic structures, no matter how complex the skeleton is. In this way, the digital shape can be decomposed into regions, which reflect the topological properties of the corresponding skeleton structures. A key point of the proposed method is the notion of interiority degree, which allows one to order skeleton structures in a hierarchical way.

1. Introduction

When working with a line pattern, it is often of interest to identify and extract its basic structures [1]. Moreover, if the line pattern is complex, establishing a hierarchy among its components is profitable to favour recognition [2,3].

A well known line pattern is the skeleton of a digital shape [4]. It is a linear subset of the shape and is characterized by the same topological and geometrical properties. Skeleton branches are in correspondence with elongated regions that can be understood as ribbons having variable width and orientation; skeleton loops represent closed ribbons. If the basic structures of the skeleton (i.e., branches and loops) are identified, a decomposition of the represented pattern into ribbons and closed ribbons becomes available. However, an immediate decomposition of the skeleton, obtained by splitting it at the branch points, does not allow to identify the desired basic structures. In fact, loops may result divided into a number of branches, so that the correspondence between closed ribbons and skeleton loops cannot be established.

In this paper, we present a decomposition method to identify the basic structures of the skeleton. In particular, we identify a number of loops equal to the number of holes in the pattern represented by the skeleton, each loop surrounding one hole of the pattern. An iterated tracing-and-deleting technique, guided by the notion of interiority degree is used. At each iteration, all the skeleton structures having the currently smallest interiority degree are vectorized. To access to more internal skeleton subsets, the already vectorized basic structures are removed. Pixel deletion is accomplished in such a way to keep constant the total number of loops (i.e., the number of the already vectorized loops plus the number of the loops still to be identified in the skeleton). The definition of peripheral branch and peripheral loop is introduced to identify at each iteration of the process the less internal structures. Though peripheral branches and peripheral loops generally simultaneously

coexist in complex skeletons, we regard as more significant to assign the peripheral branches to hierarchy levels lower that those including peripheral loops, because of the trivial topological difference between branches and loops. Thus, in the resulting hierarchy the levels do not include both types of skeleton structures.

Shape decomposition into ribbons and closed ribbons can be obtained, if desired, by expanding the identified skeleton components, by means of the reverse distance transformation [5]. The recovered regions can be hierarchically ordered according to the interiority degree of their corresponding skeleton subsets.

2. Preliminaries

Let S be the 8-connected skeleton representing a digital pattern. We do not elaborate on how to compute the skeleton since a number of algorithms can be found in the literature [6-8]. Without loss of generality, we suppose that S consists of one component. The pixels of S are also referred to as black pixels and the pixels of $\overline{S}$, complement of S, as white pixels.

The neighbors of a pixel p of S are the eight pixels surrounding p. The end points are the pixels of S having only one black neighbor. They identify the starting points of peripheral skeleton branches. The branch points are the pixels with more than two black neighbors. They identify the meeting points of skeleton branches and/or loops. All the remaining pixels of S are called normal points.

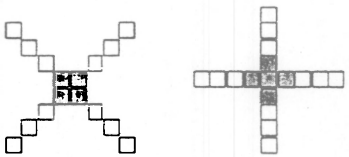

Figure 1. Examples of branch point clusters (grey pixels).

Branch points are usually grouped into connected components (clusters), when skeleton branches meet in a common region. Examples of clusters are shown in Figure 1. Every cluster is labeled by means of a recursive connected component labeling technique, so as to associate the same cluster identifier to all the branch points in the cluster.

3. Ranking via the Interiority Degree

3.1 Peripheral-branch-tracing-and-deleting

Skeleton branches and loops can be ranked according to the notion of interiority degree i_d. Let us first consider a skeleton free of loops. Peripheral branches are the less internal branches; their interiority degree is 1. To identify branches with higher and higher interiority degree, we refer to the following peripheral-branch-tracing-and-deleting process, accomplished in parallel fashion and in such a way to preserve skeleton connectedness. First, every peripheral skeleton branch is identified by tracing it from its

first extreme (the end point) up to its second extreme (the first encountered branch point). Then, all peripheral branches are deleted by removing the traced pixels, except for the branch points delimiting them. In this pruned skeleton, new peripheral branches are likely to be originated. In fact, some pixels, identified as branch points before pruning S, play the role of end points in what remains of the skeleton. The interiority degree of the current peripheral branches is 2. In general, if k iterations of the parallel peripheral-branch-tracing-and-deleting process are necessary in order a branch point can play the role of end point, the interiority degree of the peripheral skeleton branch starting from it is k+1. As an example, refer to the skeleton sketched in Figure 2.

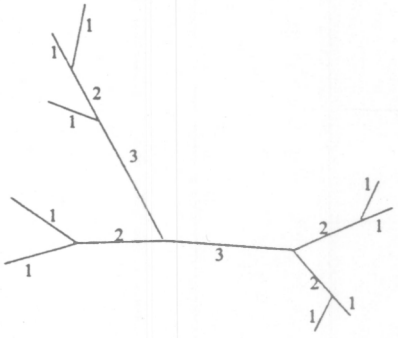

Figure 2. Skeleton branches are denoted by their respective interiority degree.

3.2 Loop-tracing-and-deleting

Let us now refer to the general case in which the skeleton includes both branches and loops, and suppose that parallel pruning has been applied until no more peripheral branches are left in the skeleton. Two cases are possible: 1) no pair of loops sharing common subsets of the skeleton exists, i.e., the loops are linked through linking branches; 2) loops directly linked to each other exist.

3.2.1 Peripheral-loop-tracing-and-deleting

Let us consider the case in which no pair of loops share a common subset of the skeleton. An example is sketched in Figure 3.

A peripheral-loop-tracing-and-deleting parallel process is introduced, which similarly to the peripheral-branch-tracing-and-deleting process is aimed at identifying the loops according to the interiority degree. The loops are traced from their inside to count the clusters of pixels still active as branch points; loops for which only one cluster exists are the peripheral loops. The pixels of each peripheral loop are deleted, except for those in the (unique) cluster of branch points, shared with a (more internal) linking branch. As soon as loop removal causes some of the linking branches to become peripheral branches, peripheral-branch-tracing-and-deleting is newly taken into account. This process is continued through more internal branches as far as peripheral branches are created by peripheral-branch-tracing-and-deleting. Then, it alternates with peripheral-loop-tracing-and-deleting until all the skeleton structures are ranked according to the interiority degree.

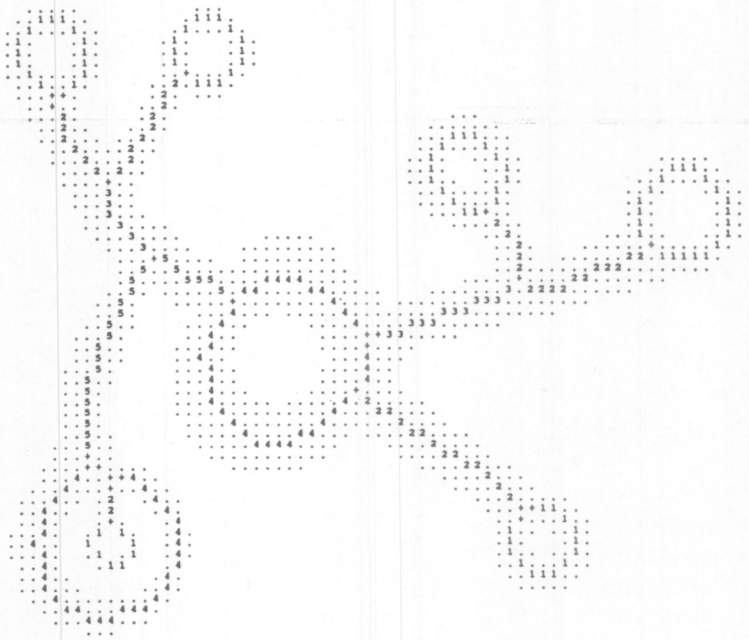

Figure 3. Pixels are labeled with the interiority degree of the skeleton structure to which they belong. Branch points are denoted by "+".

3.2.2 Loop-labeling-and-deleting

Let us now consider the second case, when pairs of loops are likely to share a number of pixels. Suppose that the branch-tracing-and-deleting process and the loop-tracing-and-deleting process have been alternated until, during the loop process, all the loops have more than one cluster of still active branch points. A parallel loop-labeling-and-deleting process is introduced, which at each iteration identifies the peripheral loops. All the loops are traced from the inside, and their pixels are labeled in such a way to record the number of times they have been visited. Suppose that initially all the skeletal pixels are assigned label 1. Then, each time a pixel of the skeleton is visited its label is increased by 1. After the loops have been traced, branch points are marked (by multiplying their label by -1). Pixels labeled 1, 2 and more than 2, respectively belong to linking branches, to only one loop, and to more than one loop. An example is shown in Figure 4.

If pixels labeled 1 do not exist, only loops (directly linked to each other) remain in the skeleton. In this case, we assign the same interiority degree to all the loops. Otherwise, only the loops including a unique component of pixels with label different from 2 are currently peripheral loops and have the same interiority degree. In fact, this condition guarantees that any such a loop does not play the role of linking element among skeleton structures. For the current peripheral loop l_i, only the pixels labeled 2 (i.e., the pixels belonging exclusively to that loop) can be safely deleted. Pixels labeled differently from 2

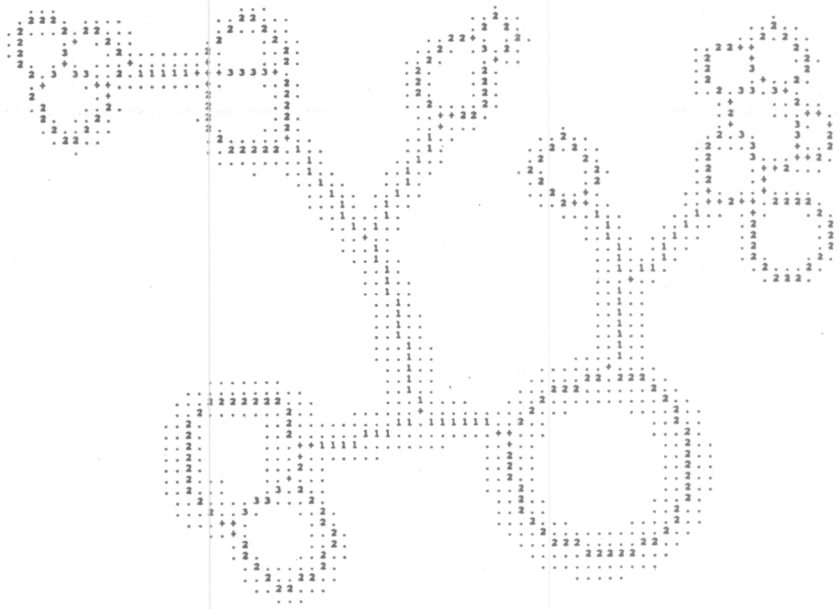

Figure 4. Labeling resulting after loops have been traced from the inside.

also belong to some other loop or are branch points; they cannot be deleted since their removal would alter the total number of loops, or the skeleton connectedness. The absolute value of the label of these pixels is diminished by 1, so as to account for the deletion of l_i, when deleting an adjacent peripheral loop. Before deleting any other peripheral loop with the same interiority degree, the pixels of l_i with negative label are checked. In fact, due to deletion of l_i, they could have become superfluous for connectedness preservation, or could no longer play the role of branch points in the remaining skeleton. These pixels are accordingly deleted or have their label changed to positive.

The loop-labeling-and-deleting process is iterated and more and more internal loops are identified. As soon as loop removal causes some linking branches to become peripheral branches, peripheral-branch-tracing-and-deleting is newly taken into account, and is repeatedly applied as far as peripheral branches are found.

It can be easily verified that in the general case it is sufficient to alternate peripheral-branch-tracing-and-deleting and loop-labeling-and-deleting to rank all the basic structures. These two processes are taken into account in the following .

4. The algorithm

We assume that the skeleton has a structure complex enough to justify its

decomposition. Each basic structure is vectorized by recording coordinates (and distanc label, if a labeled skeleton is used) of its pixels. Basically, the previously illustrate tracing-and-deleting processes are employed to identify and rank branches and loo according to the interiority degree (which coincides with the iteration number of th adopted tracing-and-deleting process).

The two dimensional picture P constitutes the input data to the process. For ease description, we consider P as a binary picture, where $S=\{1\}$ and $\bar{S}=\{0\}$. If a labele skeleton is used, P should be understood as a binary version of the picture including th skeleton.

To avoid using a large number of raster scans of P, a preliminary step of the proce is devoted to the construction of three lists, L_H, L_E and L_B, which are used to access th skeleton pixels from which the tracing-and-deleting processes start. The list L_H record for each loop of S, the coordinates of an entry point, i.e., a pixel located inside the loc which will be used to guarantee that loop tracing is accomplished from the inside. (Th entry points are found after the components of $\bar{S}$ have been labeled by means of connected component labeling process.) The lists L_E and L_B record the coordinates of th end points and of the branch points, respectively. The branch points are also marked on I so as to guarantee correct termination of skeleton branch tracing. Indeed, besides th coordinates of each branch point p, L_B includes two further fields, F_1 and F_2. F_1 is use to record the identifier of the cluster including p; F_2 stores the number denoting at whic iteration p has been reached during tracing and, hence, the interiority degree of the trace branch having p as its second extreme.

4.1. Branch-tracing driven by the end points

Starting from each end point e, directly accessed on P by using the list L_E, th corresponding peripheral branch is traced until a marked pixel b (the second extreme of th skeleton branch) is met. Vectorization is accomplished while tracing the peripheral branch At the same time, all the pixels of the branch are removed from P except for b. A completion of branch tracing, an updating of L_B is done as concerns the record relative t b: the interiority degree of the branch (i.e., the iteration number 1) is stored in F_2.

Once the list L_E has been exhausted, all the branches of S with $i_d=1$ are vectorized.

4.2 Branch-tracing driven by the branch points

Among all the branch points recorded in L_B, only those in correspondence of whic the field F_2 contains the value $i_d=1$ may have been modified into end points on the prunec skeleton. However, the value $i_d=1$ could have not been stored in the field F_2 of all the branch points of the same cluster, i.e., with the same contents stored in F_1. (An illustrative example is shown in Figure 5, where the pixel a, though belonging to the same cluster as pixels b and c, has not been reached after tracing the branches terminating in b and c. Accordingly, the field F_2 relative to the pixel a is empty.) Thus, for each cluster, F_2 is updated and assumes the maximal value among those pertaining to the visited branch points of the cluster.

The branch points for which $F_2=1$ in the list L_B are directly accessed on P one after the other, so as to vectorize the corresponding skeleton branches, if found to be peripheral

branches. For the current branch point p, the number of black neighbors is computed. If this number is greater than 1, p does not play the role of end point. If p has only one 8-connected component of black neighbors [9], p is removed. After this check, the next pixel is taken from L_B. For instance, pixel a in Figure 5 is not identified as an end point, if it is accessed before pixels b and c from the list of the branch points. Otherwise, there are still two cases.

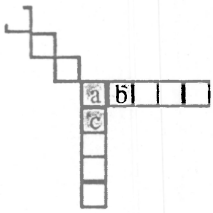

Figure 5 Dot edged pixels belong to already traced-and-deleted skeleton branches. Pixels b and c are the branch points, second extremes of these branches. Solid edged pixels belong to a not yet traced branch.

In the first case, there is only one black neighbor of p, marked as a branch point. Then, p does not identify the starting extreme of a peripheral skeleton branch, but its removal does not alter the connectedness of S. The pixel p is then removed from P and the corresponding field F_2 is set to 0 in L_B, so as to avoid further examination of that pixel. This situation occurs for the pixel out of b and c in Figure 5, which is the second to be inspected.

In the second case, there is only a black neighbor of p , which is not marked. Then, p plays the role of end point, starting extreme of a peripheral skeleton branch with i_d=2, and the tracing-and-deleting process newly starts from p. Also in this case, the field F_2 corresponding to p is set to 0 before starting branch tracing, since p is removed from P. When the second extreme q of the skeleton branch is met, the field F_2 relative to q in L_B assumes the value i_d=2, to denote that the traced skeleton branch belongs to the second hierarchy level. As before, branch vectorization is accomplished during tracing, and the pixels of the branch are removed except for q.

One inspection of the pixels of P, for which F_2=1 in the list L_B, is not enough to guarantee that the second hierarchy level has been completed. This can be checked still with reference to Figure 5. If the pixel a is accessed before pixels b and c, the skeleton branch starting from it cannot be vectorized since the pixel a does not play the role of end point. Only after pixels b and c have been accessed and removed, the pixel a plays the role of end point. To guarantee the completion of the second hierarchy level, the pixels still having the field F_2=1 in the list L_B are repeatedly accessed and the above process is newly accomplished. This is done as far as F_2 can be set to 0 for some branch point of L_B.

Before starting the third iteration of the tracing-and-deleting process to build the third hierarchy level, an updating of F_2 is accomplished so as to assign to F_2 the maximal value,

among those pertaining to the branch points with the same cluster identifier.

The above process is repeated proceeding towards more internal skeleton branch until none of the pixels in the list L_B plays the role of end point. The loop-tracing proce is then used in place of the branch-tracing process.

4.3. Loop-tracing driven by the entry points

Starting from the pixels identified on P as the entry points of L_H, the correspondi loops are traced from the inside and their pixels labeled with the number of times they ha been visited. Branch points are marked. Loops including a unique component of pixe labeled differently from 2 are the first ones to be vectorized, being the currently le internal skeleton structures. Loop removal is effective only on the pixels labeled 2, whi the absolute value of the label of pixels labeled more than 2 is diminished by 1. Pixels wi negative label are properly updated, as explained in Section 3.2.1.

Loop-labeling-and-deleting has to be repeatedly applied, since loop removal and lab decrease may cause further loops to become removable (or linking branches to becon peripheral branches). Before repeating the above process (to vectorize more internal loop guided by the list L_H, the branch points are accessed on P to simplify the skeleton structu by removing marked pixels which have only one 8-connected component of blac neighbors (refer to Section 4.2, where a similar process is described).

As soon as loop removal causes some linking branches to become peripher branches, branch-tracing-and-deleting is accomplished as far as peripheral branches ai found, before alternating again with loop-tracing-and-deleting.

As soon as loop removal causes some linking branches to become peripher branches, branch-tracing-and-deleting is accomplished as far as peripheral branches ai found, before alternating again with loop-tracing-and-deleting.

5. Conclusion

A skeleton decomposition method has been presented, based on the degree c interiority of the skeleton basic structures. The method is implemented by iterating an alternating suitable tracing-and-deleting processes, accomplished in such a way t guarantee that skeleton connectedness is not altered, and that the number of found loops i equal to the connectivity order of the skeleton.

The obtained hierarchy is such that the levels exclusively include either branches o loops. In fact, although both peripheral branches and peripheral loops could b simultaneously detected on the skeleton, we synchronize the tracing-and-deletin processes in such a way to extract first the branches. If a different synchronization i chosen, the obtained hierarchy may be different as concerns number of levels and o structures per level. In particular, both types of structures can be present in the same level For instance, branch-tracing-and-deleting and loop-tracing-and-deleting can be used withir the same iteration, instead of being alternated at different iterations. An example is shown in Figure 6, where two hierarchies are built for the same line pattern, by differently synchronizing the processes. In Figure 6a, peripheral branches (identified via the enc points and the branch points) and peripheral loops (identified via the entry points and the

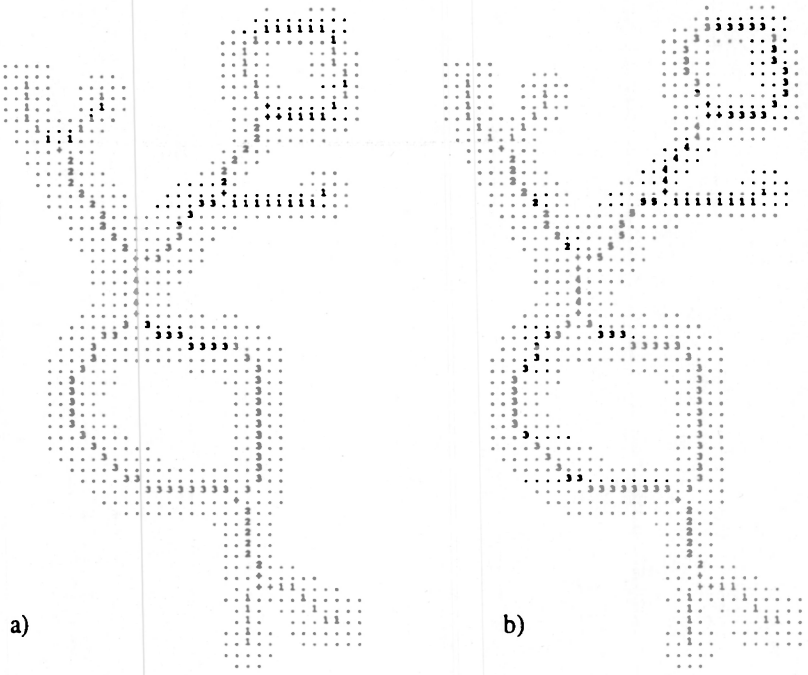

Figure 6. Hierarchies obtained using different synchronizations among the tracing-and-deleting processes.

loop-tracing-and-deleting process) are simultaneously extracted, so that they may coexist at the same level (1 and 3, in the example). Figure 6b shows the hierarchy obtained by following the synchronization described in the paper, where each level includes either branches or loops.

A peculiarity of the method, is the fact that loops are correctly identified as closed curves also in case of a complex skeleton, where pairs of loops share common skeleton subsets. This allows one to establish a correspondence between skeleton loops and closed ribbons.

The computational cost of the algorithm is rather modest. In fact, all the operations performed on the image, are accomplished by directly accessing pixels of the skeleton and by tracing skeleton subsets.

Although the method has been described with reference to the skeleton, it can be employed to identify branches and loops in other types of line patterns, e.g., flow charts and engineering drawings.

References

1 C. Shih, R. Kasturi, "Extraction of graphic primitives from images of paper based line