HIERARCHICAL DECOMPOSITION OF DISTANCE LABELED SKELETONS

SERGEY V. ABLAMEYKO
Institute of Engineering Cybernetics, Belarusian Academy of Sciences
Surganova Str.6, 220012 Minsk, Belarus

CARLO ARCELLI and GABRIELLA SANNITI DI BAJA
Istituto di Cibernetica, National Research Council of Italy
Via Toiano 6, 80072 Arco Felice, Napoli, Italy

The distance labeled skeleton of a two-dimensional digital object is hierarchically decomposed into loops, branches and concatenations of branches. Decomposition is obtained by using an iterated tracing-and-deleting process alternately applied to more and more internal skeleton subsets. Every skeleton subset is associated a feature, called the relevance, which is related to the spatial extension of the object region represented by that subset. The relevance is used to decide on how to perform concatenations, as well as to rank skeleton decomposition components ascribed to the same hierarchy level. The decomposition is stable under object rotation and can be used in a recognition process accomplished via graph matching.

Keywords: Labeled skeleton, relevance, skeleton tracing, loop, branch concatenation, hierarchical decomposition.

1. INTRODUCTION

The homotopic skeleton has been regarded as adequate to represent and analyze images like maps or engineering drawings.[1–3] It is also a convenient tool to decompose and describe two-dimensional objects whose shape can be understood as the superposition of elongated regions. These regions can be obtained via skeleton decomposition, and their spatial relations can be derived while tracing the skeleton. In this respect, the skeleton has been used when the information leading to object decomposition is carried on by contour curvature, object thickness or elongation (e.g., Refs. 4–7).

A straightforward decomposition can be obtained by splitting the skeleton at the branch points, i.e., the pixels where branches or loops meet each other. However, this decomposition does not give complete evidence to the structure of the underlying object since it fragments the loops into their constituting segments and divides the skeleton into a number of branches that may be too large.

Besides preserving the loops, a desirable decomposition should produce more meaningful skeleton subsets and also establish a hierarchy among them, so as to favor the subsequent recognition task. A step in this direction can be done by taking into account the work on the chessboard-distance transformation of line patterns described in Refs. 8–10, since the skeleton is itself a line pattern. In this case, the end points of the skeleton are considered as the sources from which distance information

International Journal of Pattern Recognition and Artificial Intelligence Vol. 10 No. 8 (1996) 957–970

propagates along the skeleton, and a decision on how to continue propagation is taken at the branch points. Distance information allows one to count the number of pixels constituting any skeleton branch. When skeleton branches meet each other at a branch point, a decision on which of the incoming branches should be concatenated with a more internal branch is taken in terms of the number of pixels. The incoming branch characterized by the maximal (minimal) number of pixels is suggested for the concatenation. The resulting skeleton decomposition turns out to be articulated into branches and concatenations of branches, among which a hierarchy can be established.

Although the branch having the maximal (minimal) number of pixels is intuitively expected to represent the more (less) elongated continuous region, counting the pixels on a skeleton branch is not enough to provide a concatenation stable under object rotation. In fact, due to the discrete nature of the digital plane, the number of pixels of a skeleton branch (and even the number of pixels constituting the digitized version of the continuous region mapped into that skeleton branch) depends on the orientation of the region. Thus, one should require that the distance transformation of line patterns be computed on the skeleton by using a reasonable approximation of the Euclidean distance, so as to account more closely for the elongation of the continuous underlying region.

In a labeled skeleton, i.e., a skeleton whose pixels are labeled with their distance from the complement of the digitized object, the numerical value attached to every label can be used to measure object local width. We consider a labeled skeleton, computed according to a weighted distance providing good approximation to the Euclidean distance (e.g., Refs. 11–13). This allows us to measure in a quasi-Euclidean way object local width, whatever object orientation.

In this paper, we obtain a hierarchical decomposition of the labeled skeleton, articulated into loops, branches and concatenations of branches, and stable under object rotation. Branches and concatenations of branches represent regions of the object that can be understood as ribbons, while loops represent closed ribbons. Similarly to Ref. 8, only one of the branches incoming into the same branch point will be concatenated with a more internal skeleton branch. To decide on how to perform concatenations, a feature (called the relevance) is associated with every incoming skeleton branch. The relevance accounts for the spatial extension of the region mapped into the skeleton branch. The quasi-Euclidean measure of the length of the branch and the quasi-Euclidean distance labels of its pixels are used to compute the relevance, and hence to guarantee decomposition stability.

The hierarchical decomposition is obtained by means of an iterated tracing-and-deleting process, that at each iteration builds a new hierarchy level. The process identifies the currently peripheral skeleton subsets (branches or loops) and, in case of branches, decides on the possible concatenations. Every concatenation of branches is assigned to the highest level, among those pertaining to the branches constituting the concatenation itself. Components at the same hierarchy level are ranked according to their respective relevance.

The algorithm provides a graph representation of the object that can be used in a recognition process, accomplished via graph matching.[14] Though not intended for a specific application, the method is of potential interest in areas such as automatic visual inspection, recognition of engineering drawings, maps and logic circuit diagrams.

2. PRELIMINARIES

Let S be the 8-connected labeled skeleton, computed according to a weighted distance function. Here we consider the (3,4)-distance,[15] and refer to the skeleton computed by using the algorithm described in Ref. 13. Without loss of generality, we suppose that S consists of one component.

The skeleton of a flat mechanical object, characterized by a variable width and by a number of holes, is illustrated in Fig. 1, superimposed over the digitized object. This skeleton will be used through the paper as a running example, to show the performance of the decomposition algorithm.

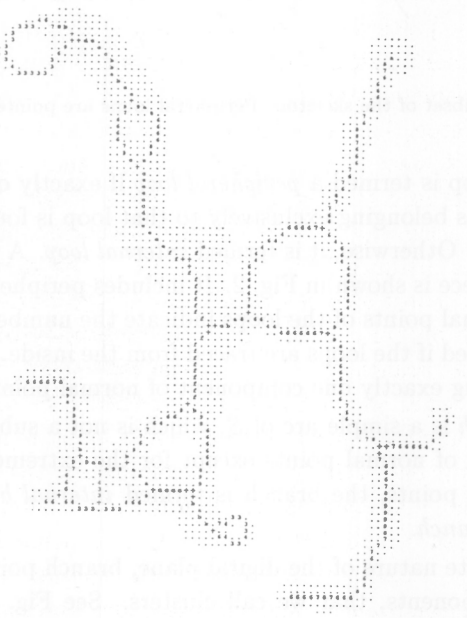


Fig. 1. The (3,4)-distance labeled skeleton of a mechanical object. Branch points are denoted by "+". Some of them are grouped into clusters.

End points, normal points and *branch points* are pixels of S having respectively one neighbor, two neighbors and more than two neighbors in S.

A loop is any simple closed curve of S surrounding one component of the complement of S, say H, and having its pixels all 4-adjacent to that component. A loop can be traced by starting from any of its pixels, grasped by one of its neighbors in the complement of S. When the grasping neighbor is a pixel of H, we say that the loop is traced from the inside. A loop includes normal points and, generally, a few

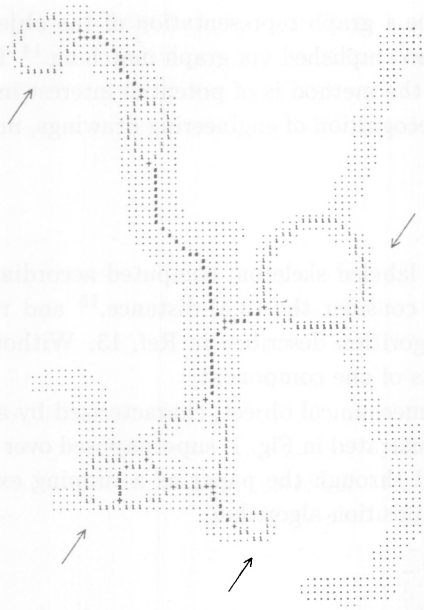


Fig. 2. A subset of the skeleton. Peripheral loops are pointed by arrows.

branch points. A loop is termed a *peripheral loop* if exactly one connected component of normal points belonging exclusively to that loop is found, when tracing the loop from the inside. Otherwise, it is termed *internal loop*. A subset of the skeleton of the mechanical piece is shown in Fig. 2. It includes peripheral loops; the integers attached to the normal points of the loops indicate the number of times the normal points would be visited if the loops are traced from the inside. The peripheral loops are the ones including exactly one component of normal points visited once.

A skeleton *branch* is a simple arc of S, which is not a subset of a skeleton loop and consists entirely of normal points except for the extremes of the arc. If both extremes are branch points, the branch is termed *internal branch*, otherwise it is termed *peripheral branch*.

Due to the discrete nature of the digital plane, branch points are often grouped into connected components, that we call clusters. See Fig. 1. We assign to the branch points of a cluster the same identifier, so as to make it clear that they all identify the same branching configuration, i.e., the region where branches or loops meet each other. For every branching configuration (including either a single branch point or a cluster), a *representative point* is selected and used as the actual meeting point for the skeleton subsets converging therein; we choose the branch point that is the topmost and leftmost pixel in the cluster.

A *concatenation* is a sequence of distinct skeleton branches, such that each pair of successive branches shares a representative point.

Branches, concatenations of branches and loops are called the components of the decomposition.

The *relevance* of a skeleton component is defined as the average value of the distance labels of its pixels multiplied by the length of the component. The length is obtained by using the (3,4)-distance to measure the unit moves necessary to trace the component.

3. METHOD

Skeleton decomposition is obtained by an iterated tracing-and-deleting process, which applies in a slightly different manner depending on whether branches or loops are examined. Branches and loops are never processed during the same iteration. Branch processing always has priority over loop processing.

At the first iteration, the process is parallelwise applied to the peripheral branches or, in the case where no peripheral branches exist, to the peripheral loops. As a result of the first iteration the skeleton is modified (pruned), and some branches or loops, that were internal in the initial skeleton, become peripheral. The next iteration is then performed on the current peripheral branches or, in their absence, on the current peripheral loops. Tracing-and-deleting is iterated until all the skeleton has been visited.

In an ideal skeleton, where all the branching configurations include a single branch point, tracing-and-deleting can be directly applied to more and more internal branches and loops. In a real skeleton, branching configurations are likely to be mapped into clusters of branch points. Thus, a process aimed at cluster simplification becomes necessary to identify new peripheral subsets, after some skeleton subsets have been pruned.

Tracing-and-deleting is coupled with a vectorization process, aimed at storing the decomposition components in compact form. The subsets, which are peripheral at the current iteration, are individually vectorized in different files, and their relevance is computed. If these subsets are branches, some of them are vectorized in the same files already used to vectorize branches at a previous iteration, so creating concatenations of branches. In particular, a branch is concatenated to the already vectorized branch having the largest relevance among those sharing the same representative point.

Every component is assigned to a level of the hierarchical decomposition, depending on the iteration number denoting the iteration at which the component is obtained. The level of a concatenation is the largest number among those pertaining to the iterations at which the branches constituting the concatenation were vectorized. Components at the same level are also ranked according to their relevance.

4. PROCEDURE

A preliminary step of the procedure is devoted to the construction of three lists, L_H, L_E and L_B, which are used to access the skeleton pixels from which tracing-and-deleting starts. For each loop of S, L_H records an entry point, i.e., a pixel of the component of the complement of S surrounded by the loop. The entry point will be used to grasp the loop and guarantee that loop tracing is accomplished from

the inside. (The entry points can be recorded provided that the components of the complement of S have been identified by means of connected component labeling.) The lists L_E and L_B record the pixels initially classified as end points and branch points, respectively. For every branch point p, L_B also records the identifier of the cluster including p, and the representative point of the cluster. Moreover, if p is itself a representative point, L_B records the relevance of the already traced and vectorized skeleton subsets incoming into the cluster, and the pointers to these subsets. Branch points are also marked on the skeleton, so as to easily recognize them while tracing the skeleton. The decomposition algorithm is outlined below.

1. Trace-and-delete and vectorize all peripheral branches.
2. Simplify clusters of branch points.
3. Repeat 1 and 2 as far as new peripheral branches are created.
4. Trace-and-delete and vectorize all peripheral loops.
5. Simplify clusters of branch points.
6. Repeat 1–5 until the skeleton is completely decomposed.

In the following, we describe cluster simplification, as well as tracing-and-deleting for the branch case and for the loop case, respectively.

4.1. Cluster Simplification

The pixels of any cluster of branch points are sequentially accessed via L_B. The current pixel p is removed if it has at least one marked neighbor and one 8-connected component of skeletal pixels in its neighborhood. Cluster simplification is repeated until no more deletable pixels exist in the cluster. With reference to the subset of the skeleton shown in Fig. 3, it can be seen that whichever is the order in which the branch points are accessed, the pixels denoted by "*" are the only removable ones.

4.2. Branch Tracing-and-Deleting

Suppose that the list L_E is not empty, so that a number m of peripheral branches exist in the initial skeleton. The kth peripheral branch, $k \leq m$, is traced-and-deleted from its first extreme (end point) up to the first encountered marked pixel p (branch point), this pixel excluded because its deletion might locally disconnect the remaining part of the skeleton. During tracing-and-deleting, the peripheral skeleton branch is vectorized in a file A_k. The pruned branch is temporarily ascribed to the first level of the hierarchy. Thereafter, the relevance of and the pointer to the kth branch are stored in the record of L_B relative to the representative point of the cluster including p.

When the list L_E has been exhausted, all the branches initially peripheral have been pruned. The skeleton resulting after the removal of the initial peripheral branches is shown in Fig. 3.

Due to branch removal and cluster simplification, some pixels, classified as branch points in the initial skeleton, may have only one neighbor in the modified skeleton and are classified as end points, if accessed by using the list L_B at the

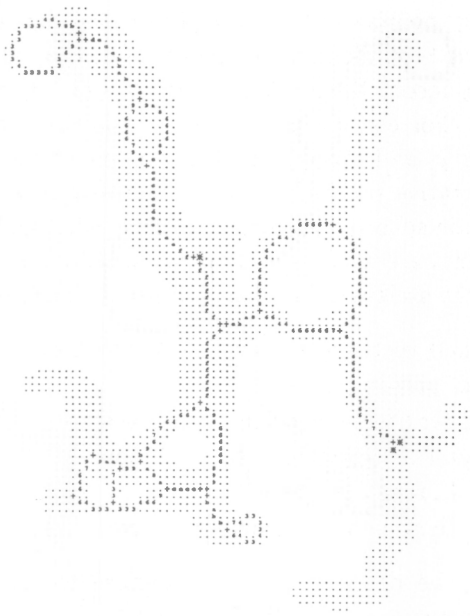


Fig. 3. Skeleton resulting after the first iteration of tracing-and-deleting. Branch points denoted by "*" are removed during cluster simplification.

second iteration. These pixels are taken as starting points of the new peripheral branches. Tracing-and-deleting is applied to every new peripheral branch which, differently from before, is vectorized in an already existing file A_k, so as to build a concatenation. In fact, in correspondence with the representative point of the cluster including the current starting point q, the pointer to the proper file A_k can be found in L_B. This A_k is the file used to vectorize the skeleton branch whose relevance is the greatest among those pertaining to the skeleton branches entering into the cluster including q. Every concatenation where a current peripheral branch is appended is (temporarily) moved to the second level of the hierarchy.

In the running example, only one new peripheral branch is created after the first iteration. The skeleton resulting after the second iteration is shown in Fig. 4.

After the list L_B has been exhausted, the second iteration is completed. Again, new peripheral branches may be generated, due to the removal of skeleton subsets accomplished during the second iteration and cluster simplification. Tracing-and-deleting is then iterated by a newly inspecting L_B. Any traced branch is appended in the proper file A_k, as the continuation of an already existing concatenation. This concatenation is (temporarily) moved to the third level of the hierarchy.

The process is repeated until new peripheral branches are generated in the pruned skeleton. When no more peripheral branches are found, the files A_k are regarded as completed and the hierarchy levels to which they have been assigned become permanent. These levels are as many as the iterations performed, but some of them may be empty. In fact, all the (concatenations of) branches temporarily ascribed to a given level might have been moved to successive levels.

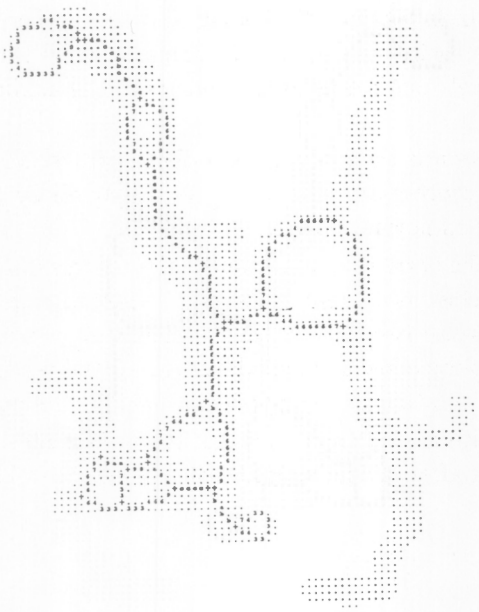


Fig. 4. Skeleton resulting after the second iteration.

Components belonging to the same hierarchy level are ranked according to their relevance. Spatial relations among components at different levels can be derived by analyzing the list L_B. In fact, the pointers to components sharing a given representative point are recorded in L_B, in correspondence with that representative point.

4.3. Loop Tracing-and-Deleting

To identify the loops peripheral at the current iteration, all loops are traced from the inside by using the list L_H of entry points. Tracing is accomplished on a copy of the current skeleton, stored in an auxiliary array. In this array, pixels of the loops are labeled with the number of times they are visited during tracing (see Fig. 2). Pixels marked as branch points and encountered during loop tracing, are labeled only if they currently have only two neighbors in the skeleton, i.e., they can be classified as normal points at this iteration. This case can be checked with reference to Figs. 2 and 4. Labelling the pixels of the loops in the auxiliary array facilitates identification of the peripheral loops, since it allows us to count easily the components of normal points belonging exclusively to a single loop.

Tracing-and-deleting is applied to each loop peripheral at the current iteration, and a new hierarchy level is filled in. Every peripheral loop is entirely vectorized in a file B_k. Differently from the branch case, not all skeletal pixels of the current peripheral loop are deleted (in both arrays). Otherwise, loops classified as non-peripheral at the current iteration could be destroyed. Loop pixels labeled "1" in the auxiliary array can be safely removed, as they belong exclusively to the periph-

eral loop at hand; pixels labeled "2" belong also to another loop and cannot be removed. They have their label set to "1" so as to allow their removal during a future inspection; pixels marked as branch points in the auxiliary array cannot be removed.

Cluster simplification is then performed in both arrays. The marked pixels still remaining after this process are labeled "1" in the auxiliary array, provided that they can be classified as normal points.

The relevance of the current loop, as well as the representative points associated with all the visited clusters is kept track of. The relevance is used to rank the loops traced at the same iteration, hence belonging to the same hierarchy level. The representative points allow us to establish the spatial relations among any loop found at the current iteration and components found at different hierarchy levels.

The skeleton resulting after the removal of the peripheral loops detected on the skeleton of Fig. 4 and after cluster simplification is shown in Fig. 5.

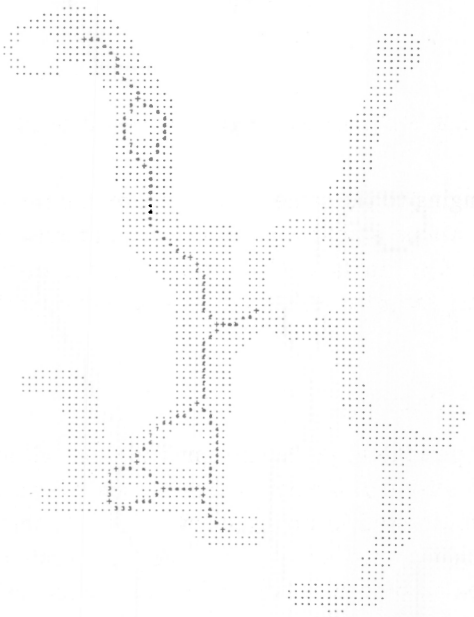
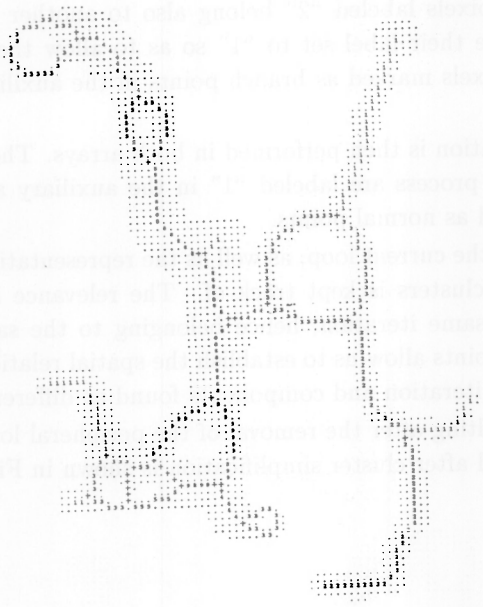
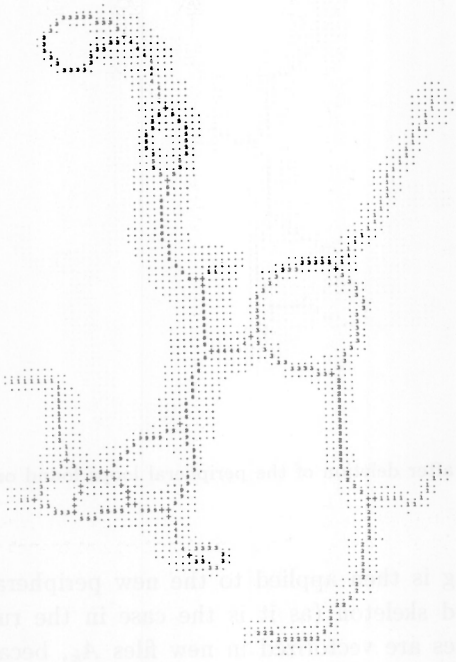


Fig. 5. Skeleton resulting after deletion of the peripheral loops found on the skeleton shown in Fig. 4.

Tracing-and-deleting is then applied to the new peripheral branches possibly present in the modified skeleton (as it is the case in the running example, see Fig. 5). These branches are vectorized in new files A_k, because they cannot be concatenated to any skeleton branch found before loop processing. If no peripheral branches exist, new peripheral loops are searched by counting the components of loop pixels labeled "1" in the auxiliary array. Tracing-and-deleting is then applied and the peripheral loops are individually vectorized in new files B_k.

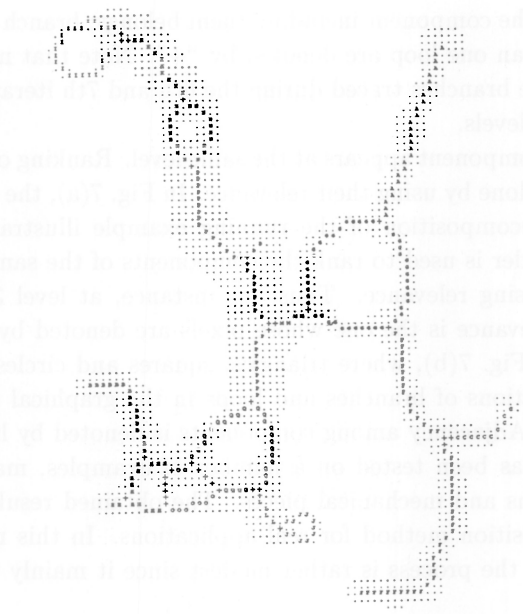


(a)

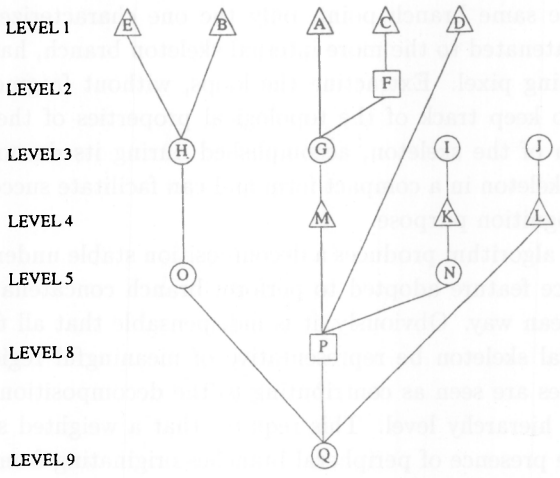


(b)

Fig. 6. The same decomposition is obtained for the running example, (a), and for the skeleton of the object slightly rotated, (b). Numbers assigned to the skeletal pixels denote the levels of the hierarchy.



(a)



(b)

Fig. 7. Letters denote ranking of the components of the hierarchy levels as found in Fig. 6(a). Triangles, squares and circles denote branches, concatenations of branches, and loops.

5. CONCLUDING REMARKS

The performance of the decomposition algorithm is shown in Fig. 6, for the skeleton of the mechanical object as it is in the running example and after a 10° rotation of the object. In both cases, the same set of components and hierarchy levels are obtained. The integers attached to the skeletal pixels denote the level of the

hierarchy to which the component including them belongs; branch points and pixels common to more than one loop are denoted by "+". Note that no pixel is labeled "6" or "7", since the branches traced during the 6th and 7th iterations were moved to higher hierarchy levels.

More than one component appears at the same level. Ranking of the components at the same level is done by using their relevance. In Fig. 7(a), the resulting ranking is shown for the decomposition of the running example illustrated in Fig. 6(a). The alphabetical order is used to rank the components of the same hierarchy level according to decreasing relevance. Thus, for instance, at level 3 the component with the largest relevance is the one whose pixels are denoted by "G". The same lettering is used in Fig. 7(b), where triangles, squares and circles represent single branches, concatenations of branches and loops in the graphical representation of the decomposition. Adjacency among components is denoted by linking paths.

The algorithm has been tested on a number of examples, mainly referring to logic circuit diagrams and mechanical pieces. The obtained results encourage the use of this decomposition method for real applications. In this respect, we point out that the cost of the process is rather modest since it mainly involves skeleton tracing.

Summarizing, a method to hierarchically decompose the labeled skeleton into loops and concatenations of branches has been illustrated. Among the branches incoming into the same branch point, only the one characterized by the largest relevance is concatenated to the more internal skeleton branch, having that branch point as its starting pixel. Extracting the loops, without fragmenting them into arcs, allows us to keep track of the topological properties of the original object. The vectorization of the skeleton, accomplished during its decomposition, allows one to store the skeleton in a compact form and can facilitate successive tasks, e.g., matching for recognition purpose.

The proposed algorithm produces a decomposition stable under object rotation, since the relevance feature adopted to perform branch concatenation is measured in a quasi-Euclidean way. Obviously, it is indispensable that all the branches and loops of the initial skeleton be representative of meaningful regions. In fact, all loops and branches are seen as contributing to the decomposition and as such are assigned to some hierarchy level. This requires that a weighted skeleton be used, so as to avoid the presence of peripheral branches originating from convex contour configurations only in the correspondence of given orientations of the object. Moreover, the adopted skeletonization algorithm should include a picture cleaning step to remove salt-and-pepper noise and, possibly, a skeleton postprocessing step so as to guarantee that the resulting skeleton is free of noisy loops and branches. This is the case for the skeletonization algorithm we have taken into account in this paper.

ACKNOWLEDGEMENTS

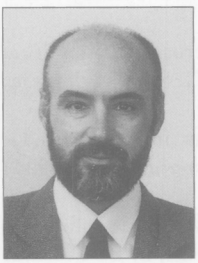
We gratefully acknowledge the help of Oleg Okun for implementing the algorithm and Salvatore Piantedosi for preparing the illustrations.

REFERENCES

1. R. Kasturi, S. Siva, and L. O'Gorman, "Techniques for line drawing interpretation: An overview", *Proc. IAPR Workshop on Machine Vision Applications*, Tokyo, 1990, pp. 151–160.
2. S. Suzuki, "Graph-based vectorization method for line patterns", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Ann Arbor, 1988, pp. 616–621.
3. S. Ablameyko, V. Bereishik, N. Paramonova, A. Marcelli, S. Ishikawa, and K. Kato, "Vectorization and representation of large-size 2-D line-drawing images", *J. Visual Communication and Image Representation* **5** (1994) 245–254.
4. C. Arcelli and G. Sanniti di Baja, "An approach to figure decomposition using width information", *Comput. Vision Graphics Image Process.* **26** (1984) 61–72.
5. P. P. Cortopassi and T. C. Rearick, "A computationally efficient algorithm for shape decomposition", *Proc. 2nd Int. Conf. on Computer Vision and Pattern Recognition*, Ann Arbor, 1988, pp. 597–601.
6. C. Arcelli, R. Colucci, and G. Sanniti di Baja, "On the description of digital strips", *Proc. Int. Conf. on Artificial Intelligence Applications and Neural Networks*, Zurich, 1990, pp. 193–196.
7. G. Sanniti di Baja and E. Thiel, "(3,4)-weighted skeleton decomposition for pattern representation and description", *Pattern Recognition* **27** (1994) 1039–1049.
8. J.-I Toriwaki, N. Kato, and T. Fukumura, "Parallel local operations for a new distance transformation of a line pattern and their applications", *IEEE Trans. Systems, Man, and Cybernetics* **9** (1979) 628–643.
9. J.-I Toriwaki and S.Yokoi, "Distance transformations and skeletons of digitized pictures with applications", in *Progress in Pattern Recognition*, eds., L. N. Kanal and A. Rosenfeld, North Holland, Amsterdam, 1981, pp. 187–264.
10. I. Ragnemalm and S. Ablameyko, "On the distance transform of line patterns", *Proc. 8th Scandinavian Conf. on Image Analysis*, Tromsø, 1993, pp. 1357–1363.
11. L. Dorst, "Pseudo-Euclidean skeletons", *Proc. 8th Int. Conf. on Pattern Recognition*, Paris, 1986, pp. 286–288.
12. C. Arcelli and M. Frucci, "Reversible skeletonization by (5,7,11)-erosion", in *Visual Form Analysis and Recognition*, eds., C. Arcelli, L. P. Cordella and G. Sanniti di Baja, Plenum, New York, 1992, pp. 21–28.
13. G. Sanniti di Baja, "Well-shaped, stable and reversible skeletons from the (3,4)-distance transform", *J. Visual Communication and Image Representation* **5** (1994) 107–115.
14. H. Bunke and B. T. Messmer, "Efficient attributed graph matching and its application to image analysis", in *Image Analysis and Processing*, eds., C. Braccini, L. De Floriani and G. Vernazza, Lecture Notes in Computer Science, vol. 974, Springer, Berlin, 1995, pp. 45–55.
15. G. Borgefors, "Distance transformation in digital images", *Comput. Vision Graphics Image Process.* **34** (1986) 344–371.

Received 16 January 1996; revised 20 June 1996.

Sergey Ablameyko received the Diploma of Higher Education in Mathematics in 1978, the Ph.D. degree in 1984, the Doctor of Sciences degree in 1990 and was appointed as Professor in 1992. Dr. Ablameyko is head of the Image Processing and Recognition Laboratory of the Institute of Engineering Cybernetics, Minsk, Belarus. He has published more than 120 publications, including books, in the field of image processing. He has contributed to the setting up of image processing systems for maps, engineering drawings, remote sensing images, PCB image applications. He was involved in the organization of a number of international conferences. He is Advisory Editor of the *International Journal of Computer Graphics & Vision*, Fellow of the IEE, Senior Member of the IEEE, member of the Governing Board of the International Association for Pattern Recognition (IAPR), member of the IAPR Publicity & Publications Committee, President of the IAPR Belarusian Society, and Vice-President of the Belarusian Association for Artificial Intelligence.



Carlo Arcelli received the doctoral degree in physics from the University of Bologna in 1969. Since 1970, he has been working at the Istituto di Cibernetica of the National Research Council of Italy, Naples, where he has done research in the fields of picture processing and digital geometry.

His current interest are mainly in shape analysis and description.

Gabriella Sanniti di Baja received the doctoral degree in Physics from the University of Naples, Naples, Italy, in 1973. Since then, she has been working in the field of image processing and pattern recognition at the Istituto di Cibernetica of the National Research Council of Italy, Naples, where she currently has the position of Director of Research.

Her main research activities concern two-dimensional shape representation, decomposition and description.