

A Minimal Line Property Preserving Representation of Line Images*

M. Burge, Georgia, U.S.A., and **W. G. Kropatsch**, Vienna

Received May 28, 1998; revised November 17, 1998

Abstract

In line image understanding a minimal line property preserving (MLPP) graph of the image complements the structural information in geometric graph representations like the run graph. With such a graph and its dual it is possible to efficiently detect topological features like loops and holes and to make use of relations like containment. We present a new rule based method on dual graph contraction for transforming the run graph and its dual into MLPP graphs. A parallel $O(\log(\text{longest curve}))$ algorithm is presented and results given.

Key Words: Document image analysis, dual graph contraction, run graph.

1. Introduction

The goal of line image analysis is to convert paper or microfilm based line images into an electronic form for easier manipulation, processing, and searching. The high scanning resolutions used during conversion result in very large images for which efficient processing methods and storage are necessary. If algorithms to manipulate and process the line images are to be efficient and have low time and space complexities they can not work directly on the $O(n^2)$ iconic representation of the image. Instead a base representation of the line image that losslessly compress its geometric structure and topology so that algorithms can efficiently access it is needed. We propose and give an efficient method of computing a new combined representation which meets these requirements.

The run graph [10] is a geometric graph where the nodes are either *node areas* or *touching point nodes* which are connected by edges which are *edge areas*. The run graph, like other encodings based on maximal rectangles, is information preserving. It provides the local topology of each image element and we can compute its dual graph to obtain the global topology of the image. In addition, the runs underlying the nodes and edges provide the spatial extent or shape of the image. The run graph uses simple local definitions to encode an image geometrically. The local

*This work was supported by the Austrian Science Foundation S7002-MAT.

nature of these definitions results in extraneous, in terms of line properties, nodes and edges.

We present a new algorithm based on Dual Graph Contraction (DGC) to transform the run graph into its Minimum Line Property Preserving (MLPP) topological form which, when implemented in parallel, requires $O(\log(\text{curvelength}))$ steps. A MLPP graph of a line image complements the structural information in geometric graph representations like the run graph. With such a graph and its dual it is possible to efficiently detect topological features like loops and holes and to make use of relations like containment.

In Section 2 we review representations for line drawings in terms of their topological and geometrical encoding ability and describe a geometric graph representation, the run graph, which provides a compact structural base representation for line drawings. Section 3 summarizes dual graph contraction (DGC) and Section 3.1 presents contraction kernel rules for transforming planar graphs and their duals into their minimal line property preserving (MLPP) form. Section 3.4 shows that the algorithmic time complexity of the new method is $O(\log(\text{longest curve}))$ when implemented in parallel. Section 4 summarizes our results and provides some examples of computing the MLPP versions of real line images from their run graphs.

2. Run Graph Encoding

In a *run length* or *interval encoding* [1] of an image, maximal sequences of black pixels in a column or row are stored. These $1 \cdot l$ rectangles form an information preserving and compressed representation of the image. Both different size rectangles and maximal squares have been used to extend this representation, the later being another representation of the Medial Axis Transform. The advantages of these representations are that they are inexpensive to compute, information preserving, and compressed. Unfortunately it is difficult to extract structural or topological information from this representation without re-encoding it.

Definition 1. (Run). *A maximal sequence of black pixels in either the horizontal (i.e., horizontal run) or the vertical (i.e., vertical run) direction. Runs can be encoded compactly by four values: a binary direction value indicating either a horizontal or vertical run, two integers the start row and start column of the run, and a single integer for the length of the run. Instead of the length which is a relative value, the end row (end column) for vertical (horizontal) runs can be stored. \square*

Definition 2. (Adjacent Run). *Two runs p and q are adjacent if they are both of the same orientation $\text{direction}(p) = \text{direction}(q)$ and if for horizontal runs the condition $(p.\text{start row} = q.\text{start row} + 1) \wedge (p.\text{end column} \geq q.\text{start column}) \wedge (p.\text{start column} \leq q.\text{end column})$ or for vertical runs the condition $(p.\text{start column} = q.\text{start column} + 1) \wedge (p.\text{end row} \geq q.\text{start row}) \wedge (p.\text{start row} \leq q.\text{end row})$ is true. \square*

A simple re-encoding of the run length representation is the *Line Adjacency Graph* (LAG) [11] in which vertical columns of pixels are encoded into runs, each run is considered a node, and adjacent runs are connected by edges. This simple graph re-encoding as shown in Fig. 1a does provide access to the local topological structure of the image, but the shape of an object is only available by examining each of its nodes.

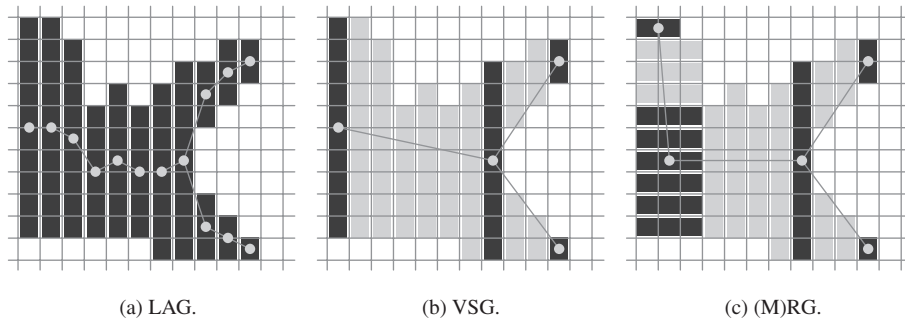


Figure 1. Graph encodings for line images, dark and light runs indicate nodes and edge respectively

2.1. Vertical Simple Graph

In the LAG every vertical run is a node of the graph and graph edges serve only to connect adjacent runs. Boatto [2] reformulate the LAG using Definitions 3 – 5 to obtain the *Vertical Simple Graph* (VSG) representation of Fig. 1b.

Definition 3. (Crossing Point Node). A crossing point node (see Fig. 2a) is a run adjacent to more than one run on a side. \square

Definition 4. (Extreme Point Node). An extreme point node (see Fig. 2b) is a run adjacent to only one other run. \square

Definition 5. (Line Edge). A run adjacent to only one run on each side is a line edge. \square

Using Definitions 3 – 5 the LAG can be reformulated as a VSG in which each node is either a crossing node or an extreme node and the edges between nodes consist of line edge runs. The definitions of extreme and crossing point nodes can be extended from single runs to sets of adjacent runs to create extreme and crossing areas. Still the VSG is built up entirely of vertical runs and when a line is horizontal it will be encoded inefficiently as a series of many vertical runs. To remedy this problem graphs built of mixed horizontal and vertical runs can be constructed [10].

2.2. Mixed Run Graph

The mixed run graph representation is built from both vertical runs (e.g., the VSG and LAG) and horizontal runs. It is conceptually a merging of vertical and horizontal simple graphs as can be seen in Fig. 1c. The following definitions for constructing the run graph are based upon the extensions of Monagan [10] to the formulation by Boatto [2] of the run graph. First maximal vertical and horizontal runs are found, then by applying Definitions 7 – 13 adjacent horizontal runs become an edge and vertical runs become either a node or an edge. Nodes are always encoded by vertical runs, whereas edges are encoded by either horizontal or vertical runs depending upon their slope. See Fig. 3 for an example of a run graph encoding.

Definition 6. (Predecessor and Successor). *The run p is the predecessor of q and q is the successor of p if they are adjacent and for horizontal runs (q .start row = p .start row + 1) and for vertical runs (q .start column = p .start column + 1). \square*

Definition 7. (Regular and Singular). *A run is regular if it has one predecessor and one successor, otherwise it is singular. \square*

Definition 8. (Conjugate). *Runs are conjugate if they cross (i.e., have exactly one pixel in common). \square*

Definition 9. (Short Vertical Runs). *A vertical run is a short vertical run if it is regular and not longer than all its conjugates. \square*

Definition 10. (Short Horizontal Runs). *A horizontal run is a short horizontal run if it is regular and shorter than all its conjugates. \square*

Definition 11. (Edge Area). *An edge area is a maximal sequence of adjacent short horizontal runs or short vertical runs. \square*

Definition 12. (Node Area). *A node area is a maximal sequence of adjacent horizontal runs and sub-runs which belong to neither vertical nor horizontal short runs. \square*

Definition 13. (Touching Point Node). *A touching point node (see Fig. 2c) ensures that eight-connectivity is maintained. It is created when a horizontal and a vertical run only touch at a diagonal. \square*

¹ We will refer to the mixed run graph representation as simply the run graph representation from here on.

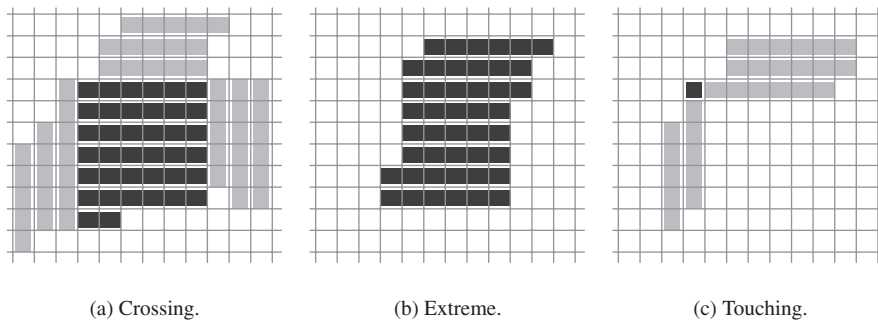


Figure 2. Types of run graph nodes

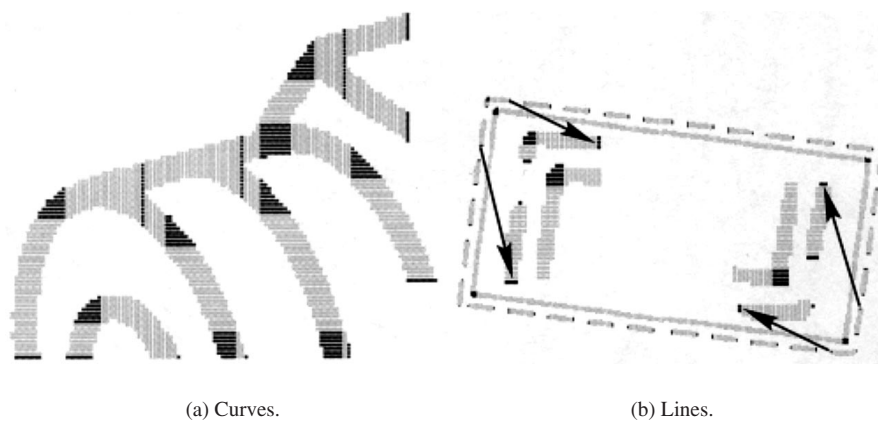


Figure 3. Line images encoded as run graphs

2.3. Run Graph Extensions

We extend the run graph to eliminate some artifact nodes and edges. Often in the run graph, diagonal lines and lines with spur pixels are divided into a number of small node-edge-node sequences some of which can be removed by using Definition 14.

Definition 14. (Artifact Node Area). *If a node area has exactly one predecessor and exactly one successor edge and both of these edges have similarly oriented short runs then it is labeled as an artifact node area (e.g., Fig. 4a) and merged with the adjacent edge areas to form a single edge. □*

In addition using Definition 15 we can eliminate artifact edge areas which arise when extreme point nodes are separated from crossing point nodes by a single vertical short run. (e.g., Fig. 4b).

Definition 15. (Artifact Edge Area). *If an edge consisting of a exactly one run connects an extreme point node and a crossing point node then it is labeled as an*

artifact edge area (e.g., Fig. 4b) and it is merged with the two adjacent node areas to form a single node. \square

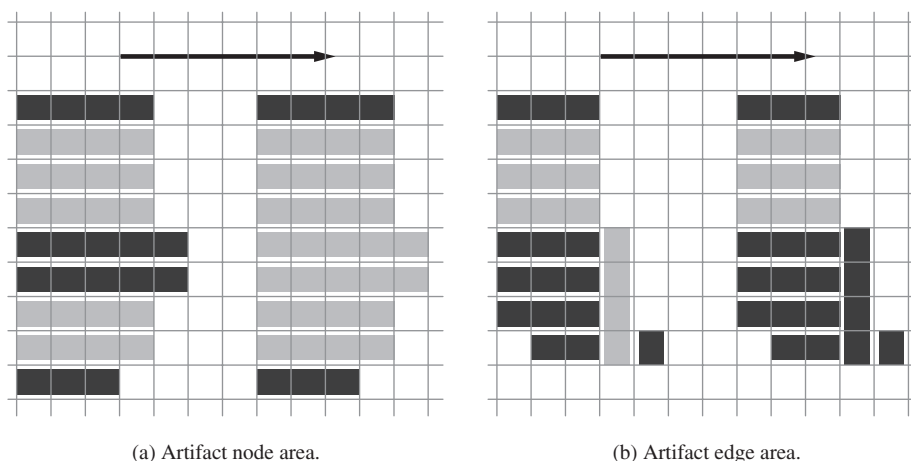


Figure 4. Artifact areas eliminated by Definitions 14, 15

Definitions 14 and 15 are purposely limited since the information available in the run graph representation is relatively local. Additional processing of the run graph to remove artifacts requires making assumptions about the domain of the image and consequently reduce its usefulness as a general base representation for document images. If such processing is to be done, it should occur at a later stage and make use of both domain knowledge and global information to adapt the definitions. Note in all figures Definition 14 has been applied but Definition 15 has not.

Definition 16. (Artifact Edge Area (Extended)). *If an edge area consisting of less than n runs connects an extreme point node and a crossing point node then it is labeled as an artifact edge area and it is merged with the two adjacent node areas to form a single node. \square*

3. Dual Graph Contraction

Dual graph contraction is the basic process [13] that builds an irregular graph pyramid by successively contracting the dual image graph of one level into the smaller dual image graph of the next level. Since dual image graphs are typically defined by the neighbor relation of image pixels we present the transformation in such terms even though the actual implementation starts at the run graph level. It is based on the operation of edge contraction as explained in Definition 17.

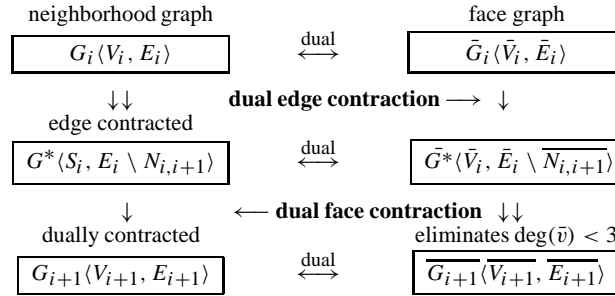


Figure 5. Dual Graph Contraction $(G_{i+1}, \bar{G}_{i+1}) = C[(G_i, \bar{G}_i), \langle S_i, N_{i,i+1} \rangle]$

Definition 17. (Edge contraction). Given an edge, e , between two vertices, p and q , let E_q be the edges adjacent to vertex q and E_p those adjacent to vertex p . If p (q) is selected as the surviving vertex then contracting e results in the removal of vertex q (p) and the addition to vertex p (q) of the edges E_q (E_p). \square

Dual graph contraction [12] contracts a graph, $G_i = \langle V_i, E_i \rangle$, according to a contraction kernel, $N_{i,i+1} \subset E_i$, while preserving (at least) its connectivity, planarity, and topology.

It can be divided (e.g., Fig. 5) into two parallel and simultaneous steps [5]:

- *Dual edge contraction:*
 The contraction kernel $N_{i,i+1}$ defines the subset of edges of G_i to contract and the direction of contraction is given by selecting a surviving vertex for each contraction this forms a subgraph $\langle S_i, N_{i,i+1} \rangle$ which must be a spanning forest of G_i . The edges are contracted into the surviving vertices S_i , and their duals are removed from \bar{E}_i .
- *Dual face contraction:*
 Faces in the dual graph of degree 1 and 2 are contracted to remove unnecessary multi-edges and self-loops.

3.1. Curve Labeling Rules

In general the lines of a line drawing represent curves connecting end points and junctions. The discrete sampling resulting from digitizing the image splits these curves into small curve segments which must be correctly reconnected during contraction. As in the regular curve pyramid [6], Curves and cells are related by several cell classes (Fig. 6) and to simplify comparison with previous methods [8] we use a grid graph base where each pixel cell is a node with edges between each













Label	Examples	Explanation
0-cell		 contains no curve, an empty cell
2-cell		 a single curve enters and exits the cell at two particular boundary segments
1-cell		 a curve ends in this cell it enters the cell at a particular boundary segments
*-cell		 a cell where curves meet
1-edge		 curve segment intersects edge
0-edge		 no curve segment intersects edge

Figure 6. Curve labels for vertices and edges

of its 4-adjacent pixel cells as in Fig. 7a.

We assume that the pixel grid overlays a set of curves where the cell classes are consistent (i.e., if a curve crosses a boundary segment then both cells adjacent to this segment are in the correct class) and that all curves are distinguishable in the base (i.e., there is no more than one curve in each cell of the base except in *-cells). We initialize the algorithm by assigning all cells in the base (i.e., the squares in Fig. 7a) one of the four cell-classes using the following simple algorithm. If a curve crosses an edge, then the edge receives an attribute 1 otherwise 0 and all pixel cells sum the attributes of their incident edges. Sums of 0, 1, and 2 correspond to 0-cell, 1-cell, and 2-cells respectively, while cells with a sum greater than 2 are *-cells. In the dual irregular pyramids the cells where the curves are represented are contracted and not the dual graph as was done in [7].

3.2. Selecting the Contraction Kernels

As introduced by Kropatsch [4] the graph transformation starts at a pixel based grid graph. In developing real world document image analysis methods we can not use a pixel based grid graph as the base since it would require four times more space than the already $O(N^2)$ iconic representation and would lead to $O(N^2)$ time algorithms. By using the run graph representation directly as the base for the transformation we reduce our memory requirements and allow arbitrarily thick curves and objects while still obtaining the advantages of the transform.

A 0-cell is inserted for each face of the run graph and an edge between it and each vertex of the face is created. All other cell values of a vertex can be computed directly from their degree using Equation 1 before applying the rules of Tables 1a to select the contraction kernel.

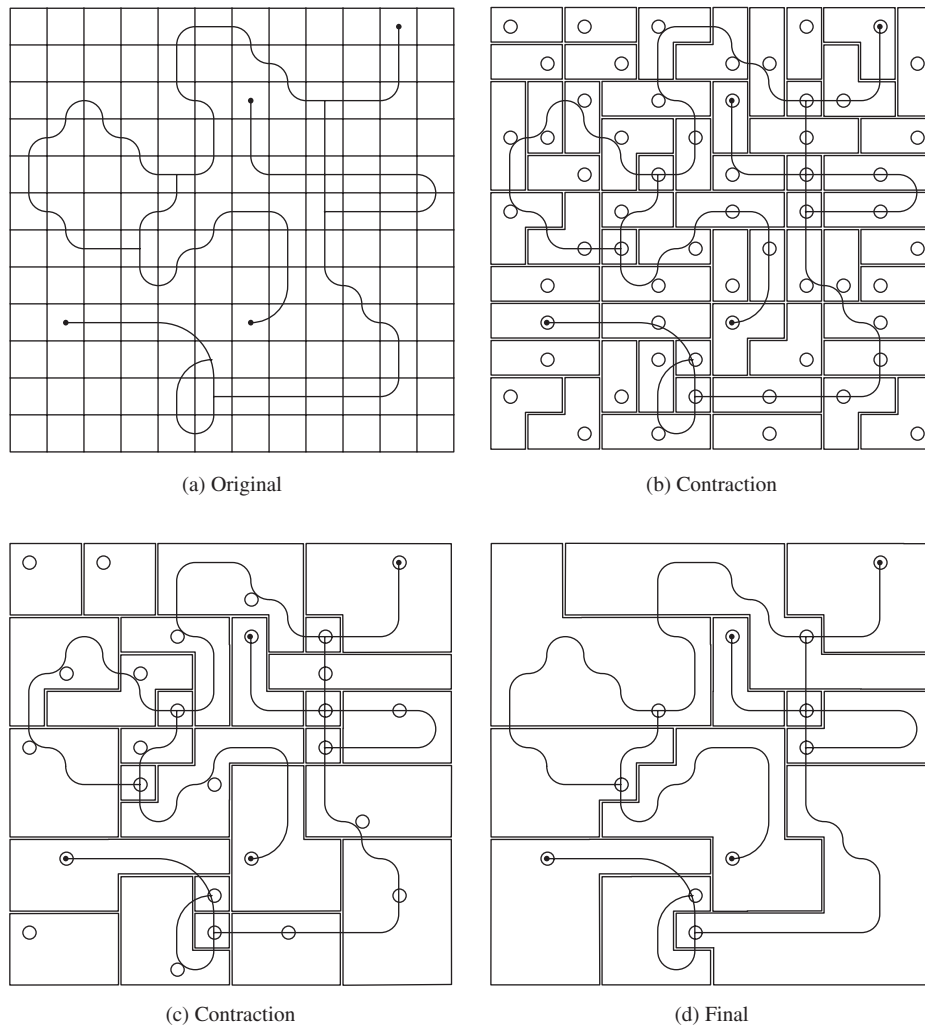


Figure 7. Three step contraction preserving line connectivity

$$\text{cell value} = \begin{cases} d(v) & \text{if } d(v) < 3 \\ * & \text{otherwise} \end{cases} \tag{1}$$

$$d(v) = \text{degree}(v) - |\text{adjacent 0-cells}|$$

The rules are derived from those in [4] but differ in that we now allow *-cells to merge with 0-cells and 2-cells since the geometrical position of the junction is inherited from the base graph. Due to this change faces in the dual graph surrounded by either one (i.e., a self-loop) or two parallel marked edges may appear and must

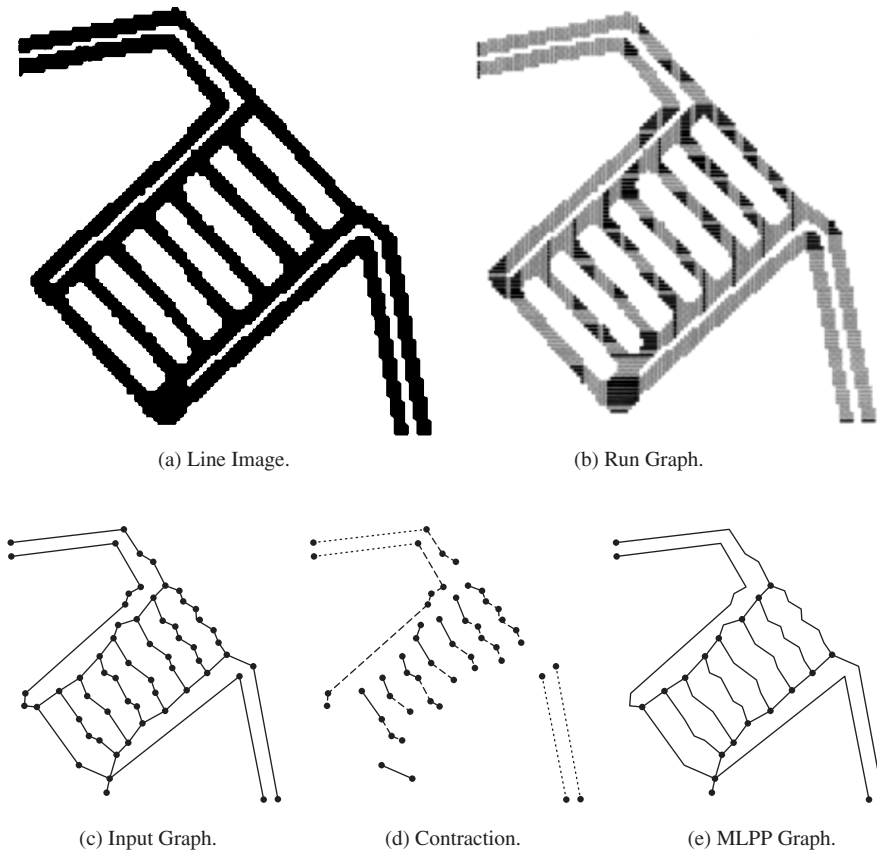


Figure 8. DCG starting from a run graph base

Table 1. Selecting the contraction kernel for (S, N)

Rule	(S, N)	Becomes
R12	$\boxed{1} \xrightarrow{1} 2$	1
R10	$\boxed{1} \rightarrow 0$	1
R22	$\boxed{2} \xrightarrow{1} 2$	2
R20	$\boxed{2} \rightarrow 0$	2
R00	$\boxed{0} \rightarrow 0$	0
R*2	$\boxed{*} \xrightarrow{1} 2$	*
R*0	$\boxed{*} \rightarrow 0$	*

(a) Selection rules

	0	1	2	*
0	R00	R10	R20	R*0
1	R10	C2	R12	C1
2	R20	R12	R22	R*2
*	R*0	C1	R*2	C3

(b) Recursive Application

be excluded from dual face contraction so that no line segments are lost in the

final graph. 1-cells and *-cells must always survive and the bridges of connecting paths inherit their attributes to the new edge. Random selection, as in adaptive pyramids [3], applies whenever the given rules do not determine the contraction kernels completely. The rules (Table 1a) are selected in the order presented below:

1. A 1-cell can merge with an adjacent 2-cell (R12) or 0-cell (R10).
2. A *-cell can merge with an adjacent (connected by a 1-edge) 2-cell (R*2) or with any adjacent 0-cell (R*0).
3. A 2-cell can merge with two adjacent (connected by 1-edges) 2-cells (R22) or with any adjacent 0-cell (R20).
4. A 0-cell can merge with any adjacent 0-cell and remains a 0-cell (R00).

3.3. Properties Preserved

The rules are applied recursively, as shown in Table 1, to dually contract the graphs until no further contraction is possible. The resulting graph has the following properties: there are no 0-cells and no 2-cells present, the number of 1-cells and the number of *-cells is the same as in the base graph. This results in a topologically correct and minimal description for all possible planar configurations of curves regardless of how complicated their layout is. All curves have been contracted to minimal length and those which were separate in the base remain distinct. The connectivity information of the base is preserved and all empty space has been removed. Any further deletions would remove either a line end point (state C2, Table 1b) or crossing point (states C1 and C3, Table 1b) and hence destroy the line topology.

3.4. Computational Complexity

One iteration of dual graph contraction reduces the length of the curves in terms of the number of edges by at least a factor of two since surviving vertices on the curves are not allowed to be neighbors. After n iterations the curve has been reduced by a factor of 2^n . No further contraction is possible when all curves between curve ends and junctions have become a single edge. Hence the number of iterations needed until convergence is $O(\log(\text{longest curve}))$.

4. Results and Conclusions

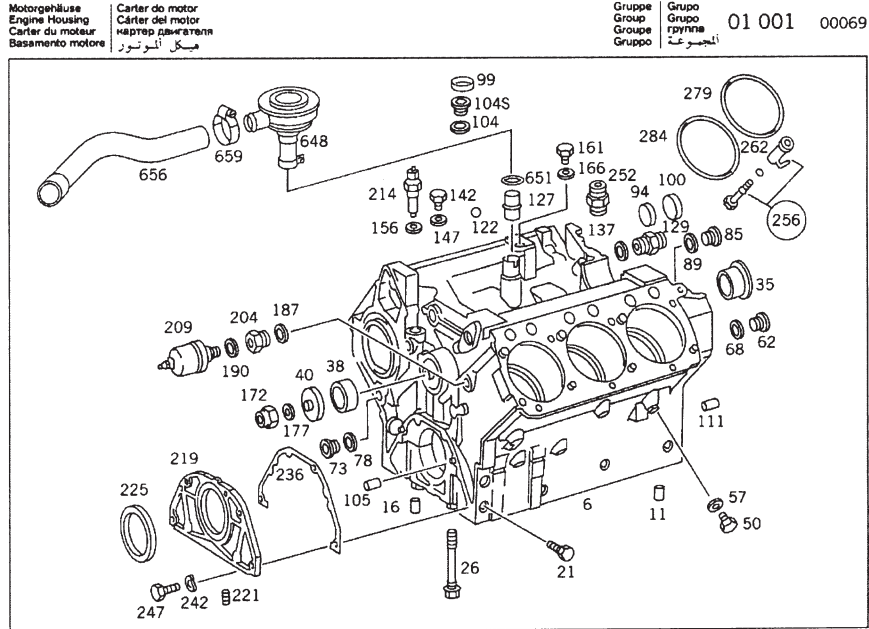


Table 2. Statistics of the MLPP graph for the above image

Contr.	V	R12	R22	R*2	E
Input	1953				2172
1	1953	92	358	329	2172
2	1174	3	10	117	1391
3	1044	0	0	1	1261
Result	1043				1260

The MLPP graph, Fig. 8e, of a staircase section, Fig. 8a, of a cadastral map is computed from its run graph representation, Fig. 8b, which exhibits many topologically extraneous nodes and edges. In the contraction kernel, Fig. 8d, edges arising from rule R12 are shown as dotted lines, rule R22 as dashed lines, and rule R*2 as thick solid lines. The run graph and dual graph contraction have both been implemented in C++ using the LEDA [9] library, see [5] for details of the implementation and the IAPR Technical Committee 15 software page www.prip.tuwien.ac.at/TC15/software.html for dual graph contraction software.

Tables 2 and 3 give an example of computing the MLPP on two real world line images. Computing the MLPP graph for the 6251 x 4416 pixel line image shown in Table 2 on a Sparc 20 with 64 Megabytes of memory took 28.88 seconds for the run graph and 12.35 seconds for the selection and dual graph contraction.

IS03-006 08/29/95

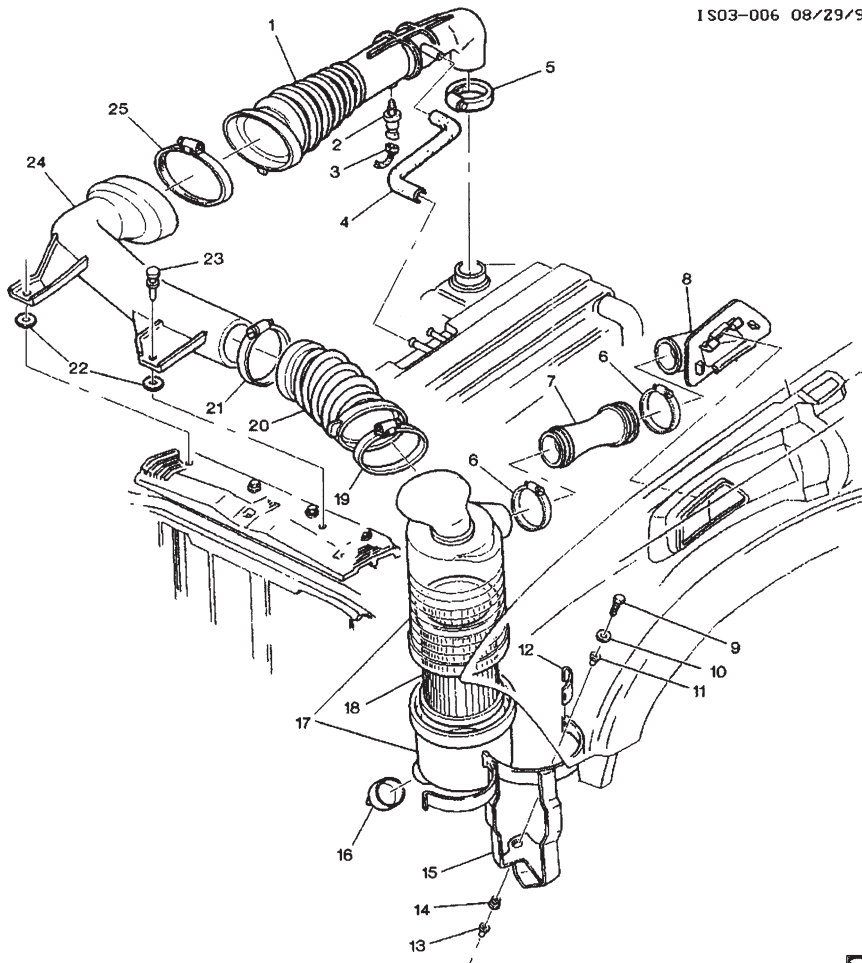


Table 3. Statistics of the MLPP graph for on the above image

Contr.	V	R12	R22	R*2	E
Input	2522				2672
1	2522	214	501	325	2672
2	1482	7	22	83	1482
3	1370	1	9	15	1370
4	1345	0	2	3	1345
5	1340	0	1	1	1340
Result	1338				1412

The run graph provides a compact, structural representation for line image understanding, but because of its geometric nature, it does not succinctly describe the topology of a line drawing. Our new algorithm based on DGC transforms the run

graph into its MLPP topological form and when implemented in parallel requires $O(\log(\text{longest curve}))$ time.

References

- [1] Aggarwal, A. K., Kulkarni, A. V.: A sequential approach to the extraction of shape features. *Comp. Graph. Image Proc.* 6, 538–557 (1977).
- [2] Boatto, L., Consorti, V., Del Buono, M., Di Zenzo, S., Eramo, V., Esposito, A., Melcarne, F., Meucci, M., Morelli, A., Mosciatti, M., Scarci, S., Tucci, M.: An Interpretation System for Land Register Maps. *IEEE Comput.* 25, 25–34 (1992).
- [3] Jolion, J., Montanvert, A.: The adaptive pyramid, a framework for 2D image analysis. *Comput. Vision Graph. Image Proc.* 55, 339–348 (1992).
- [4] Kropatsch, W. G.: Property preserving hierarchical graph transformations. In: *Advances in visual form analysis* (Arcelli, C., Cordella, L. P., Sanniti di Baja, G., eds.), pp. 340–349. Singapore: World Scientific, 1998.
- [5] Kropatsch, W. G., Burge, M., Ben-Yacoub, S., Selmaoui, N.: Dual graph contraction with Leda. *Computing [Suppl]* 12, 101–110 (1998).
- [6] Kropatsch, W. G.: Curve representations in multiple resolutions. *Pattern Rec. Lett.* 6, 179–184 (1987).
- [7] Kropatsch, W. G., Willersinn, D.: Parallel line grouping in irregular curve pyramids. In: *Proceedings Computer Vision and Pattern Recognition - CVPR'93*, pp. 784–785. IEEE Comp. Soc. Press, 1993.
- [8] Meer, P., Sher, C. A., Rosenfeld, A.: The chain pyramid: Hierarchical contour processing. *PAMI* 12, 363–376 (1990).
- [9] Mehlhorn, K., Naher, S.: Leda, a platform for combinatorial and geometric computing. *Comm. ACM* 38, 96–102 (1995).
- [10] Monagan, G., Rössli, M.: Appropriate base representation using a run graph. In: *Proceedings of the Second International Conf. on Document Analysis and Recognition*, pp. 623–626, Tsukuba, Japan, October 20–22 1993. IAPR, IEEE Computer Society Press.
- [11] Pavlidis, T.: A minimum storage boundary tracing algorithm and its application to automatic inspection. *IEEE Trans. Systems Man Cybern.* 8, 66–69 (1978).
- [12] Solina, F., Kropatsch, W., Klette, R. (eds.): *Theoretical foundations of computer vision*. Computing [Suppl] 11 (1996).
- [13] Willersinn, D., Kropatsch, W. G.: Dual graph contraction for irregular pyramids. In: *12th IAPR International Conference on Pattern Recognition volume III* (Peleg, S., Ullman, S., Yeshurun, Y., eds.), pp. 251–256. IEEE Comp.Soc., 1994.

M. Burge
 Computer Science Department
 Armstrong Atlantic University
 Savannah, Georgia 31419
 U.S.A.
 e-mail: mburge@acm.org

W. G. Kropatsch
 Vienna University of Technology
 Institute of Automation 183/2
 Pattern Recg. & Image Proc. Group
 A-1040 Vienna, Austria
 e-mail: krw@prip.tuwien.ac.at