Minimizing the Topological Structure of Line Images*

Walter G. Kropatsch[1] and Mark Burge[2]

[1] Vienna University of Technology, Institute of Automation 183/2, Pattern Recognition and Image Processing Group, Treitlstr.3, A-1040 Wien, Austria.
[2] Johannes Kepler University, Institute of Systems Science, Computer Vision Laboratory, A-4040 Linz, Austria. burge@cast.uni-linz.ac.at

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

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.

2 Line Image Representations

A natural representation for an image is an array where each element represents the intensity or color in a spatially corresponding area of the image. This representation maps itself very simply and elegantly into most programming languages and enables a natural raster style of processing. A disadvantage is that the representation does not reflect the structure of the image because regardless of whether the image contains a few scattered on pixels or a complex scene the

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

amount of memory allocated is constant and dependent only on the dimensions of the image (i.e., $w \cdot h \cdot s$ where w and h are the width and height of the image and s is the storage size of a pixel).

When processing line images we need to use high scanning resolutions to ensure that details are not merged (e.g., the touching character problem) and that they contain enough information to distinguish among them (e.g., the thickness of different lines). The memory requirements for the array representation is $O(n^2)$ in terms of the number of pixels, leading to $O(n^2)$ time algorithms, to avoid this we will not use an iconic representation but instead a symbolic one in which the primitives are groups of pixels.

2.1 Skeleton and Contour Based

Many skeleton based representations of images have been derived from the Medial Axis Transform (MAT) [1]. The MAT consists of the set of points in which each point is equidistant from its two closest boundary points. Skeletons are often computed using local operators making them sensitive to noise and causing many artifacts including unwanted short segments, called spurs, which jut off of a skeleton edge. Many algorithms [2] and heuristics have been proposed to rectify the above problems, but in addition skeletons are an inherently lossy or non-information preserving representation. Skeletons are often computed using local operators making them sensitive to noise and causing many artifacts including unwanted short segments, called spurs, which jut off of a skeleton edge.

While the skeleton representation does encode the local topology of each image element (i.e., connected component consisting of four connected on pixels) topological relations for the entire image like containment are not preserved. Skeletons are not ideal for encoding the spatial extent of an image element since most algorithms perform heuristics to remove artifacts from the skeleton, after which it is no longer possible to recreate the original contours of the image element using the radius function. Contour based representations on the other hand are especially suited for encoding this type of information.

Contour based representations consider only the border pixels of an object (e.g., the symbol “8” contains three contours: the outside double loop and the two inner loops). In order to extract some of the structure of the contour and to compress the representation, contours are often encoded. The most common encoding is Freeman [3] or chain coding in which the contour is followed from a given starting point by encoding each pixel’s location in terms of the relative direction (e.g., up, down, left, right) from its neighbor. The Freeman code is compact because the absolute value for each pixel is not stored but instead for each contour a single absolute value and the shorter relative values are stored.

Contour representations are information preserving and do compress the image but it is difficult to derive the topology of an image from its contour representation. More sophisticated versions of Freeman coding, like the Primitives Chain Coding (PCC) [4], do encode the complete line structure and topology by indicating the end points and branching points of lines.

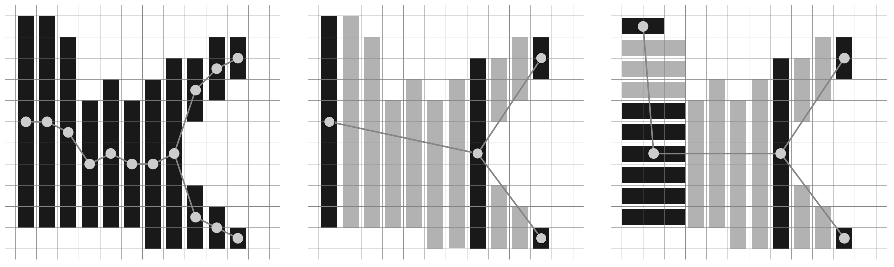

(a) LAG. (b) VSG. (c) (M)RG.

Fig. 1. Line image graph encodings, dark and light runs indicate nodes and edge respectively.

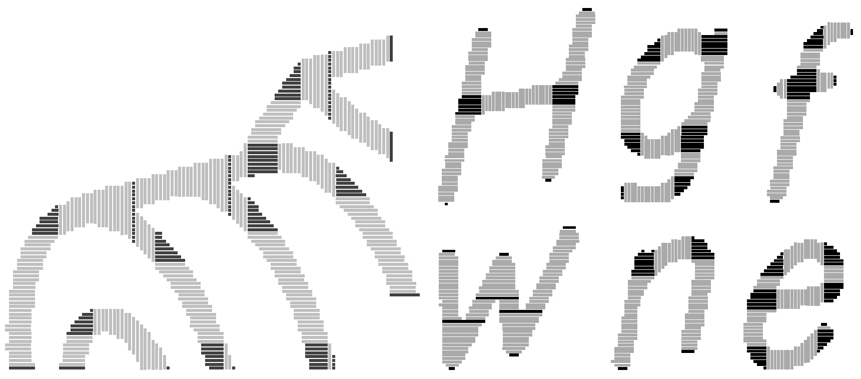
Contour representations are especially sensitive to noise along object borders, such noise, in the form of spur pixels, is unfortunately often an artifact of scanning. These noise pixels result in spurious features when encoded with chain codes like PCC. Extensive progress [5] in detecting and eliminating such noise has been made, but unlike standard chain coding which can be done in a single pass through the image such methods are heuristic based, computationally expensive and are not information preserving.

One of the drawbacks of contour representations is that they do not encode the topology of objects, whereas skeleton representations do. In addition a disadvantage of skeleton representations is there inability to encode the spatial extent of objects in contrast to contour representations. A representation combining contours and there corresponding skeletons [6] provides the local topology of skeletons and the shape information of contours, this representation is however computationally expensive to create and requires sophisticated data structures to use efficiently.

2.2 Graph Based

In a *run length* or *interval encoding* [7] 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.

A simple re-encoding of the run length representation is the *Line Adjacency Graph* (LAG) [8] in which vertical columns of pixels are encoded into runs, each run is considered a node, and adjacent runs are connected by edges. This



(a) Curves. (b) Lines.

Fig. 2. Line images encoded as run graphs.

simple graph re-encoding as shown in Figure 1(a) 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.

In the LAG every vertical run is a node of the graph and graph edges serve only to connect adjacent runs. The LAG can be reformulated [9] to obtain the Vertical Simple Graph (VSG) representation of Figure 1(b). As the VSG is built up entirely of vertical runs, when a line is near vertical it will be encoded in a non-optimal fashion, instead graphs built of mixed horizontal and vertical runs can be constructed.

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 Figure 1(c).

The run graph representation for an image is constructed by first finding maximal vertical and horizontal runs and then using simple rules (i.e., see [10] where we extend those of Monagan [11] and [9]) to encode them into node and edge areas. The run graph is information preserving; the runs underlying the nodes and edges completely encode the shape of the image. The run graph provides the local topology of each image element and we can compute its dual graph to obtain the topology of the entire image.

Often in document image analysis we are interested in the line properties (i.e., the end and crossing points of lines) of the line image. Since the run graph is built from simple, local definitions [10] it contains many nodes and edges which are extraneous in a minimum line property preserving representation. We now present a new general method for transforming run graphs into their minimum line property preserving (MLPP) graph representations.

3 Dual Graph Contraction (DGC)

Dual graph contraction is the basic process [12] 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.

$$\begin{tikzcd}
\text{neighborhood graph} & & \text{face graph} \\
\boxed{G_i\langle V_i, E_i\rangle} \arrow[rr, leftrightarrow, "\text{dual}"] \arrow[d, Rightarrow, "\text{edge contracted}"'] & \textbf{dual edge contraction} \arrow[r] & \boxed{\overline{G_i}\langle \overline{V_i}, \overline{E_i}\rangle} \arrow[d] \\
\boxed{G^*\langle S_i, E_i \setminus N_{i,i+1}\rangle} \arrow[rr, leftrightarrow, "\text{dual}"] \arrow[d, "\text{dually contracted}"'] & \textbf{dual face contraction} \arrow[l] & \boxed{\overline{G^*}\langle \overline{V_i}, \overline{E_i} \setminus \overline{N_{i,i+1}}\rangle} \arrow[d, Rightarrow, "\text{eliminates } \deg(\overline{v}) < 3"] \\
\boxed{G_{i+1}\langle V_{i+1}, E_{i+1}\rangle} \arrow[rr, leftrightarrow, "\text{dual}"] & & \boxed{\overline{G_{i+1}}\langle \overline{V_{i+1}}, \overline{E_{i+1}}\rangle}
\end{tikzcd}$$

Fig. 3. Dual Graph Contraction $(G_{i+1}, \overline{G_{i+1}}) = C[(G_i, \overline{G_i}), \langle S_i, N_{i,i+1}\rangle]$.

Figure 3 summarizes dual graph contraction, It proceeds in two major steps: dual edge contraction and dual face contraction. The first step contracts all edges of graph $G_i = \langle V_i, E_i\rangle$ that are selected by the contraction kernel $N_{i,i+1} \subset E_i$ into the surviving vertices S_i. The decimation is controlled by the subgraph $\langle S_i, N_{i,i+1}\rangle$ which must be a spanning forest of G_i. Duality of the pair of input graphs G_i and $\overline{G_i}$ is preserved by simply removing all those dual edges $\overline{N_{i,i+1}}$ that correspond to a contracted edge in $N_{i,i+1}$. The second step removes unnecessary multi-edges and self-loops by dually contracting faces of degree less than 3.

Decimation parameters can be selected in a number of different ways (e.g., data dependent selection criteria or by interactive selection). In this paper it is controlled by the selection criteria derived from the topological relations between the lines in the image. The criteria are formulated as a set of formal rules. They allow to define those important entities of the represented data that should survive the contraction process. Independent of the chosen parameters, the process preserves several properties of the surviving parts: their connectivity, planarity, and topology.

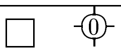
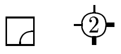
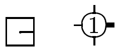
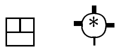
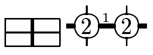
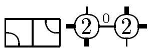
Label	Examples	Explanation
0-cell		contains no curve, an empty cell
2-cell		a single curve enters and exits the cell at two particular boundary segments
1-cell		a curve ends in this cell it enters the cell at a particular boundary segment
*-cell		a cell where curves meet
1-edge		curve segment intersects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

Fig. 4. Curve labels for vertices and edges.

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. Curves and cells are related by several cell classes (Fig. 4) and to simplify comparison with previous methods [13] we use a grid graph base where each pixel cell is a node with edges between each of its 4-adjacent pixel cells.

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 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 [14].

3.2 Selecting the Contraction Kernels

The rules in Table 1(a) differ slightly from those in [15] 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 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 [16], applies whenever the given rules do not determine the contraction kernels completely. The rules of Table 1(a) 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).

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

(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) Application.

Table 1. Selecting the contraction kernel for $\langle S, N \rangle$.

We apply these rules recursively, as shown in Table 1(b), 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 is the same as in the base graph.

Our method constructs 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 or crossing point and hence destroy the line topology.

3.3 Computational Complexity

One iteration of dual graph contraction reduces the length of the curves in terms of edges by at least a factor of two since surviving vertices of 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 $\mathcal{O}(\log(\text{max-curve-length}))$.

4 Implementation and Conclusions

In presenting the theory of the graph transformation we used a pixel based grid graph as the base and made the assumption that curves were always only a single pixel wide. In developing real world line image understanding methods we can not use a pixel based grid graph as the base since it would require roughly four times more storage then the already too large n^2 iconic representation and would lead to $O(n^2)$ 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 ensuring the properties of the transform.

When using the run graph as the base only the 0-cells need to be labeled and all other cell values can be computed from their degree as follows:

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

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

0-cells encoded the space between curves in the grid graph and were necessary to ensure that the topology of the dual graph was correct. When using the run graph as base far fewer are necessary, a single 0-cell is inserted for each face of the run graph and an edge between it and each vertex of the face is created. By using the run graph as a base we eliminate the need for preprocessing the graph to set node labels and can immediately apply the rules of Tables 1 to select the contraction kernel and dually contract the graph until no more contractions are possible. The run graph and dual graph contraction have both been implemented in C++ see [17] for details of the implementation. The MLPP graph, Figure 5(d), of a staircase section of a cadastral map is computed from its run graph representation, Figure 5(a), which exhibits many topologically extraneous nodes and edges. In the contraction kernel, figure 5(c), edges arising from rule R12 are shown as dotted lines, rule R22 as dashed lines, and rule R*2 as thick solid lines.

Using this implementation, computation of the MLPP graph from a the 6251 x 4416 pixel line image of an engine 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, results are summarized in Table 2. 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

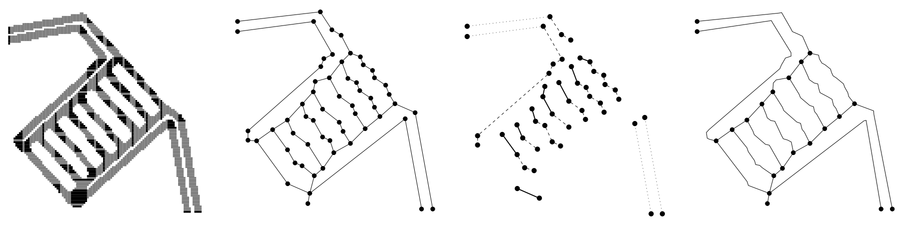


Fig. 5. DGC starting from a run graph base.

into its MLPP topological form, and when implemented in parallel, requires $O(\log(\text{longestcurve}))$ steps.

Software for dual graph contraction is available from the IAPR Technical Committee 15 software page http://www.prip.tuwien.ac.at/TC15/software.html

References

[1] H. Blum and R. N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10(3):167–180, 1978.

[2] L. Lam, S. W. Lee, and C. Y. Suen. Thinning methodologies: A comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(9):869–885, September 1992.

[3] H. Freeman. Computer processing of line drawing images. *Surveys*, 6(1):57–97, March 1974.

[4] L. O'Gorman. Primitives chain code. In *Computer Vision and Image Processing*, pages 167–183, 1992.

[5] P. Zhu and P. M. Chirlian. On critical-point detection of digital shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(8):737–748, August 1995.

[6] O. Hori and D. S. Doermann. Quantitative measurement of the performance of raster-to-vector conversion algorithms. *GRMA95*, pages 57–68, 1996.

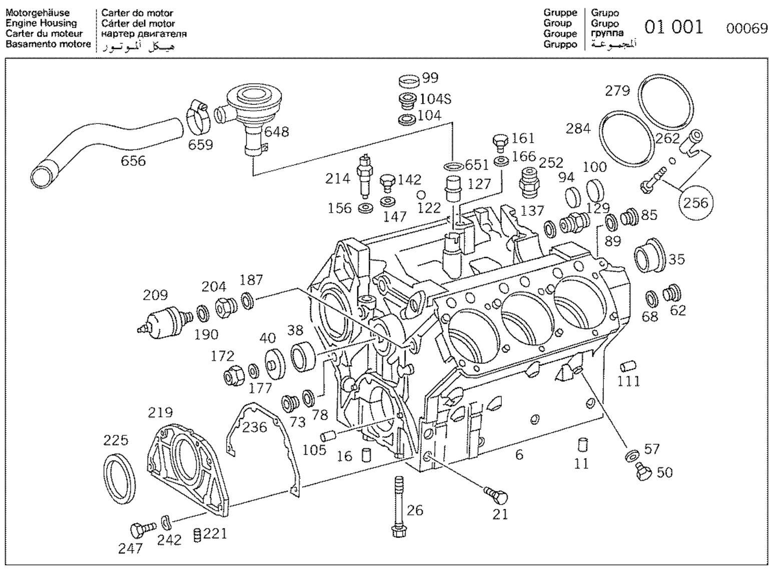
[7] A. K. Aggarwal and A. V. Kulkarni. A sequential approach to the extraction of shape features. *Comp. Graph. and Image Proc.*, 6(6):538–557, December 1977.

[8] T. Pavlidis. A minimum storage boundary tracing algorithm and its application to automatic inspection. *IEEE Trans. Systems, Man, and Cybernetics*, 8(1):66–69, January 1978.

[9] L. Boatto, V. Consorti, M. C. Buono, S. DiZenzo, V. Eramo, A. Melcarne, M. Meucci, A. Morelli, M. Mosciatti, S. Scarci, and M. Tucci. An interpretation system for land register maps. *Computer*, 25(7):25–33, July 1992.

[10] M. Burge and W. Kropatsch. Contracting line images using run graphs. In *22th Workshop of the Austrian Association for Pattern Recognition.* ÖAGM, R. Oldenbourg Verlag, 1998.

[11] G. Monagan and M. Röösli. Appropriate Base Representation Using a Run Graph. In *Proceedings of the Second International Conf. on Document Analysis and Recognition*, pages 623–626, Tsukuba, Japan, October 20-22 1993. IAPR, IEEE Computer Society Press.



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

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

[12] Dieter Willersinn and Walter G. Kropatsch. Dual graph contraction for irregular pyramids. In *12th IAPR*, volume III, pages 251–256. IEEE, 1994.

[13] P. Meer, C.A. Sher, and A. Rosenfeld. The chain pyramid: Hierarchical contour processing. *PAMI*, 12(4):363–376, April 1990.

[14] Walter G. Kropatsch and Dieter Willersinn. Parallel line grouping in irregular curve pyramids. In *Proceedings Computer Vision and Pattern Recognition - CVPR'93*, pages 784–785. IEEE Comp.Soc.Press, 1993.

[15] W. G. Kropatsch. Property Preserving Hierarchical Graph Transformations. In Carlo Arcelli, Luigi P. Cordella, and Gabriella Sanniti di Baja, editors, *Advances in Visual Form Analysis*, pages 340–349. World Scientific Publishing Company, 1998.

[16] Jean-Michel Jolion and Annick Montanvert. The adaptive pyramid, a framework for 2D image analysis. *Computer Vision, Graphics and Image Processing*, 55(3):pp.339–348, May 1992.

[17] W. G. Kropatsch, M. Burge, S. Ben Yacoub, and N. Selmaoui. Dual graph contraction with leda. *Computing, Supplementum: Graph Based Representations in Pattern Recognition*, To Appear, 1998. Online at http://www.cast.uni-linz.ac.at/Vision/papers/tc15-97/