Proceedings of the 5th WSEAS International Conference on Telecommunications and Informatics, Istanbul, Turkey, May 27-29, 2006 (pp133-138)

A Rapid Algorithm for Topology Construction from a Set of Line Segments

SEBASTIAN KRIVOGRAD, MLADEN TRLEP, BORUT ŽALIK
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova ulica 17, SI - 2000 Maribor
SLOVENIA

Abstract: - An efficient algorithm for topology construction from a set of line segments is considered in the paper. The algorithm works in two steps: input analyzing, and topology construction. In the first step, inconsistencies in input are solved. The second step consists of two parts: the envelope construction, and determination of neighbouring relations between the obtained neighbouring The time complexity of the first part highly depends on the input data distribution, and the second part, which presents our original work, runs in *O(n)*, what is confirmed by experiments using artificial input and real data from geographic database.

Key-Words: - Geographic Information System (GIS), Topology Construction, Uniform Plane

1 Introduction

In the past, land-maps were drawn manually on large sheets of papers. These traditional maps have been gradually replaced by digital formats as computers took-over the storage and maintenance of the data with the graphical content. The traditional maps have started to be digitalised, filtered, and vectorized [1, 2]. In this way, more and less complete and correct set of line segments is obtained. However, although these maps could look perfectly correct, the obtained set of line segments does not contain topological information, which is necessary for various spatial analyses and a geometric reasoning of land-maps [3, 4]. However, the lack of the topology information is not related only with the scanned old cadastre maps, but it also arises during actual activities. For example, spatial data are frequently obtained in a pure digital form (i.e. satellite images, orthophoto), surveyors measure visible points and not closed parcels. Therefore, solutions, which automatically construct the topological information, are more than welcome [3]. Some actual GIS systems include program solutions for the problem being considered here, but we have not found any hints for implementation. Recently, Žalik [5] proposed an algorithm for topology construction. We have analyzed his solution, identified bottlenecks, optimized the algorithm, and partially employed it in our solution. As a result, a very fast, numerically stable, and easy to implement solution has been obtained, and it is presented in the paper.

The paper is organized in five sections. After the introductory one, detection and removal of inconsistencies in the input data are briefly considered in Section 2. Section 3 presents the algorithm for topology construction. In Section 4, analysis of the algorithm time complexity, and practical results are given. At the end, the results are emphasized and the work is evaluated.

2 Detection and Removal of Inconsistencies

Input set of line segments frequently contains inconsistencies, which have to be detected and removed before the topology construction algorithm is used. Fig. 1 shows some of the characteristic inconsistencies:

- Line segments intersect but the intersection point is not marked as a vertex (case (a) in Fig. 1).
- line segment crosses for a small distance another line segment instead of ending on it (case (b) in Fig. 1).
- Similarly to the previous case, a line segment should touch another line segment in a vertex (case (c) in Fig. 1).
- A vertex is isolated (case (d) in Fig. 1).
- Vertices should be joined (case (e) in Fig. 1).
- In a vertex, at least two line segments should meet each other (case (f) in Fig. 1).
- A line segment is isolated (case (g) in Fig. 1).

A detailed description how the inconsistencies are detected and removed is described by Žalik [5]. We give just a brief summary here. The following tasks are implemented:

1. All input polylines are split into line segments.
2. The presented line segments are checked for intersections. The approach described by Žalik [5] uses a uniform plane subdivision for this task. This approach includes computationally expensive rasterisation of the line segments, and relatively complicated way of marking the line segments being already tested for intersection. It turns out as the most critical part concerning the spent CPU time. Instead of the uniform plane subdivision, the sweep-line approach can be applied to accelerate

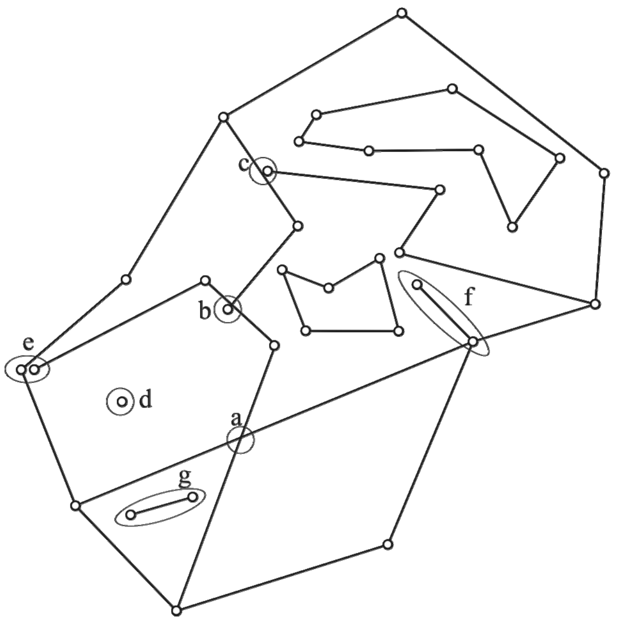


Fig. 1: An inconsistent set of geometric data

calculations of the intersection points [6]. This approach requires sorting of the presented line segments, but this task is much easier and faster than rasterisation of the line segments.

3. Vertices are examined to remove duplicates, to join vertices being closer than prescribed tolerance, and to remove isolated vertices. All vertex problems are reduced to the closest point problem, one of the most fundamental problems of computational geometry [7]. For that, a uniform plane subdivision (UPS) turns out as a very efficient solution as far a distribution of vertices is uniform enough [8]. Our experience presented Section 4 confirms that UPS can be efficiently used in GIS environment.

The steps described above automatically remove the majority of existing inconsistencies. Unfortunately, some ambiguous cases require a user intervention. Case (a) in Fig. 2, for example, shows two vertices being very close (within prescribed tolerance). Each of them defines a pair of line segments (this is the reason why they are not merged automatically). The algorithm does not know whether these vertices should be joined or a tiny corridor exists between them. Case (b) in Fig. 2 shows similar situation. Such cases are visualized and the user is asked for advice.

During the process of removing inconsistencies, the data structure is build. It consists of a list of vertices and a list of line segments. Each vertex contains information about the line segments terminating in it.

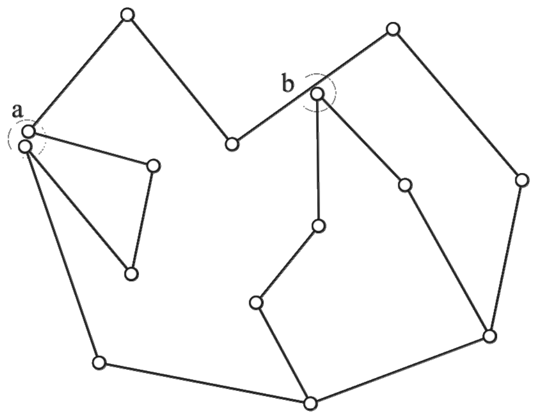


Fig. 2: Possible errors in data

3 Problem Solution

The algorithm for constructing polygons from inconsistencies-free set of line segment consists from three steps:

- determination of a set of line segments forming the outer border of the area,
- creation of polygons (parcels), and
- establishing spatial relationships between obtained polygons (finding the parents of holes).

The algorithm proposed by Žalik [5] omits the last step and therefore, the topology is not complete.

3.1 Construction of the outer border of the area

The set of line segments forming the outer border of the area will be considered as an envelope in our case. The construction begins with selecting the vertex, which for sure belongs to the envelope. This is, for example, the vertex with the minimal *x*-coordinate (vertex 1 in Fig. 3). If there are more vertices with the same *x*-coordinate, the one with the smallest *y*-coordinate is chosen. For the first line segment, the line segment with the smallest angle regarding the negative *x*-axis is chosen (line segment (1, 2) from Fig. 3). After this, the algorithm moves to the end vertex of the current line segment (vertex 2 in Fig. 3). The next line segment is selected among the line segments ending in the observed vertex (except the one employed already). The line segment forming the smallest angle with the last line segment in the envelope is selected (line segment (2, 7) in Fig. 3). The process continues until the start vertex is reached. In this way, the line segments of the envelope are oriented in counter-clockwise direction. In our case, the envelope consists from the following sequence of vertices: 1, 2, 7, 8, 9, 10, 11, 12, 13, and 14.

3.2 Construction of the polygons

Construction of polygons always starts with the line segment being the member of the envelope. Such line segment limits a single polygon. Let us suppose the line

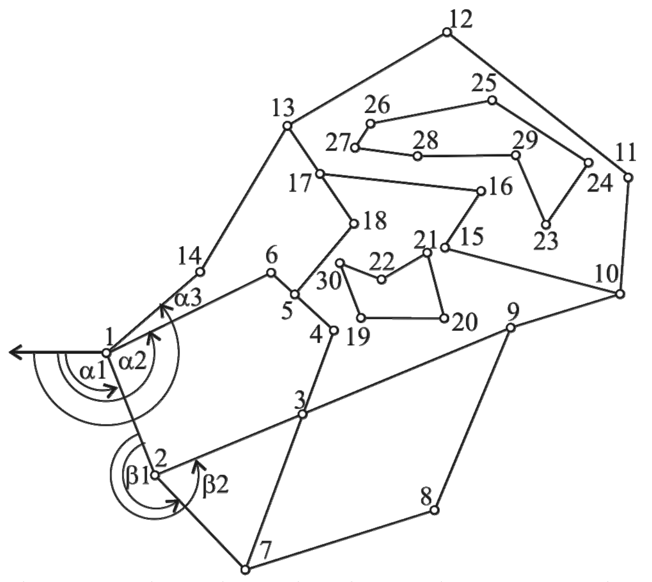


Fig. 3: Consistent input data for topology construction algorithm

segment (1, 2) from Fig. 3 has been chosen. For the next line segment, the algorithm chooses the line segment with the largest angle to the current one. In Fig. 3, the selection is made between the line segments (2, 7) and (2, 3). Because $\beta_2 > \beta_1$, the line segment (2, 3) is the right one. The algorithm continues until the starting line segment is reached, and in this way, the polygon is closed. In our case, the polygon defined by the vertices 1, 2, 3, 4, 5, 6 is obtained. The employed line segments belonging to the envelope (the line segment (1, 2)) are removed from the data structure, and the rest of the employed line segments are inserted into the envelope. They are already oriented adequately.

In this way, the algorithm continues until the envelope is not empty. However, in some cases some line segments remain in the data structure. They belong to the polygon holes or polygons inside the holes. In Fig. 3, two holes are shown defined by vertices 19, 20, 21, 22, 30, and 23, 24, 25, 26, 27, 28, 29. The described algorithm is applied again.

3.3 Establishing spatial relationships

To establish spatial relationships between the created polygons and holes, the parents of the holes have to be determined. For each hole, one of the hole vertices could be selected, and then tested regarding the containment against all the polygons. One of the known point-in-polygon algorithms could be applied for this [9, 10]. However, this would require testing all the polygons in the worst case for each hole. However, the uniform plane subdivision, which has been used during inconsistency detection and removing, can be applied here efficiently. The algorithm is based on assumption that the closest vertex of the presented polygons is surrounded by the polygons, which contain the tested point. For this reason, each vertex stores a list of polygons, which surround it. If none of the surrounding polygons contains the tested point, the neighbouring parcels are added. Fig. 4 shows an example. The vertex 2 is the closest to the tested vertex 1. All polygons marked by a are examined at first. As seen, none of them contains the vertex 1. The neighbours marked by b are checked in the next step and one of them contains the vertex.

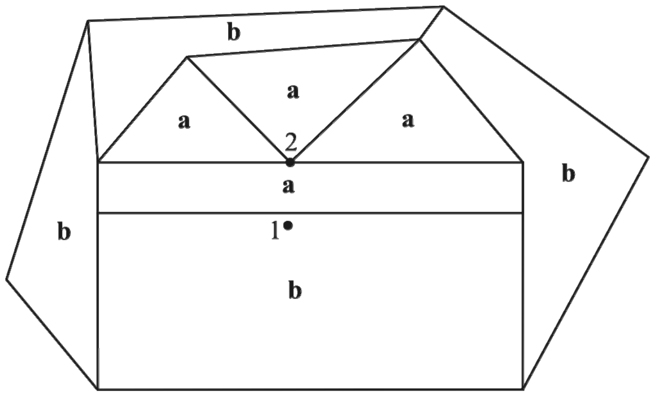


Fig. 4: Location of the polygons for the containment test

4 Results

As described, the algorithm for topology construction consists of two steps:

- detecting and removing inconsistencies and
- topology construction.

In this section, we consider theoretical time complexity for the topology construction with inconsistencies already removed. Namely, the detection and removal of them highly depends on the input data. The algorithm for the topology construction is implemented in four steps:

1. At first, all line segments are added to their ending vertices. Two vertices define each line segment, and thus this part is finished in linear time *O(n)*, where *n* is the total number of the line segments.
2. The angle of each line segment regarding the negative *x*-axis is calculated. This task is also terminated in *O(n)*.
3. The line segments are sorted around their vertices regarding the angles against *x*-axis. At each vertex, there are m_i line segments (and m_i angles). We can expect

$$\sum_{i=0}^{n-1} m_i = 2n \Rightarrow m_i << n, \quad 0 \le i < n$$

Therefore, the sort in each vertex does not depend on *n*; i.e. it is performed in *O(1)* time. As we have *n* vertices, the common time complexity is *O(n)*.

4. The construction of the topology requires that each line segment is passed exactly twice giving us time complexity *O(n)*. If holes are detected, the parent polygons have to be found. If we assume distribution of vertices which is not extremely non-uniform, the closest point to tested point is found in constant time *O(1)*, as proved by Bentley [8]. After this, the point-in polygon problem has to be solved. Experimentally, we have shown that only $t << n$ polygons have to be checked, and therefore, the expected time complexity remains *O(n)*. It is worth to mention that in real examples, very frequently, holes do not exist at all. However, in highly non-uniformly distributed input data with a large number of holes, this part consumes a lot of time, and the time complexity grows up to $O(n^2)$.

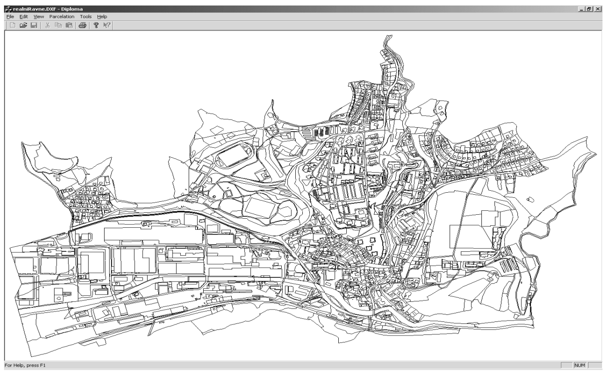
As seen, the expected time complexity of the algorithm for topology construction is *O(n)*.

4.1 Measurements of Spent CPU Time

In this subsection, the measurements of the algorithm's run-time are given. A computer PC Pentium Celeron 600 MHz with 128K Cash and 384MB RAM running under Windows 2000 was employed. The algorithm is implemented in C++. The measurements have been done on five real examples (one of them is shown in Fig. 5) and three artificial examples (Fig. 6). Table 1 summarises the results, where the first 5 entries correspond to the real examples.

The diagram in Fig. 7 shows spent CPU time for topology construction regarding the number of the output line segments without determination of neighbouring relations. The spent CPU time for topology construction with the determination of the neighbouring relations regarding the number of holes is shown by the diagram in Fig. 8. In this case, only the real examples containing holes are considered.

Table 2 shows the number of vertices searched to find the nearest vertex during the determination of the neighbouring relations.


a)

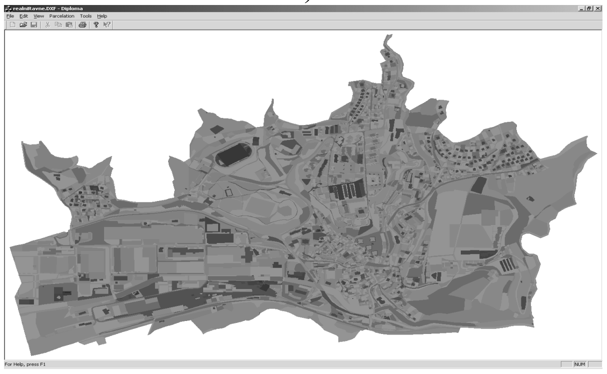
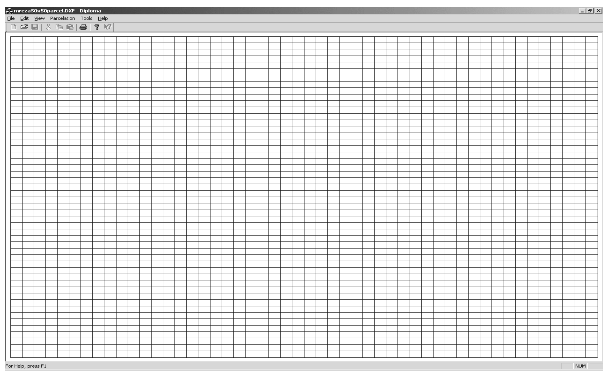

b)

Fig. 5: A real example: a) 48672 line segments on input, b) 19964 on output

	Ex. 1	Ex. 2	Ex. 3	Ex. 4	Ex. 5	Ex. 6	Ex. 7	Ex. 8
No. of input edges	2,467	11,464	17,413	36,297	48,672	102	202	402
No. of input vertices	2,395	4,897	14,990	15,820	20,014	200	400	800
No. of output edges	2,467	5,941	17,413	18,305	19,964	5,100	20,200	80,400
No. of output vertices	2,395	4,897	14,990	15,820	24,399	2,601	10,201	40,401
No. of polygons	73	1,078	2,583	2,560	4,764	2,500	10,000	40,000
No. of holes	0	35	193	205	542	0	0	0
Sort of line segments	0.004	0.025	0.045	0.106	0.144	0.000	0.000	0.000
Inconsistencies removal	0.038	0.483	0.990	1,379	3.017	0.223	1.306	12.514
Association of line segments to vertices	0.006	0.014	0.043	0.044	0.059	0.013	0.042	0.191
Angle calculation	0.002	0.006	0.015	0.014	0.022	0.000	0.011	0.044
Angle sort	0.006	0.012	0.034	0.036	0.047	0.008	0.034	0.139
Walk-through without father search	0.017	0.050	0.150	0.150	0.210	0.043	0.187	0.754
Walk-through with father search	0.017	0.069	0.331	0.358	0.938	0.043	0.187	0.754
Total time for topology construction without father search	0.031	0.082	0.242	0.244	0.338	0.064	0.274	1.128
Total time for topology construction with father search	0.031	0.101	0.423	0.452	1,066	0.064	0.274	1.128

Table 1: CPU times (in sec) for five real examples and three artificial examples



a)

b)

Fig. 6: An artificial example: a) 102 line segments on input, b) 5100 on output

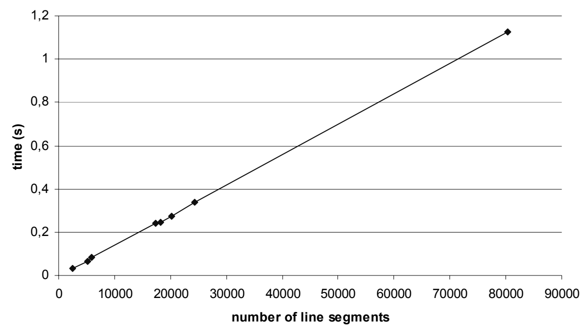


Fig. 7: Topology construction without determination of neighbouring relations

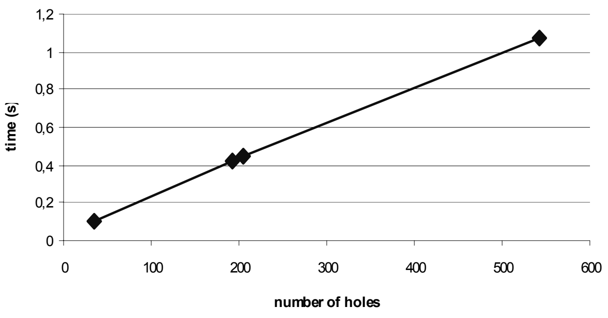


Fig. 8: Topology construction with determination of neighbouring relations

	Ex. 2	Ex. 3	Ex. 4	Ex. 5
Min no. of points	2	7	7	5
Max no. of points	59	257	134	137
Average no. of points	21.4	62.8	40.2	50.0
Min no. of polygons	1	1	1	1
Max no. of polygons	63	44	34	370
Average no. of polygons	4.2	2.6	3.1	5.0

Table 2: Number of searched near vertices and polygons for father search

At the end, we give the comparison of spent CPU time with the algorithm developed by Žalik [5] for one of the real examples and an artificial one (Table 3). The time comparison is made on sum of time needed for removal of inconsistencies and time needed for topology construction without father search.

	real data containing 11464 input edges	**artificial data containing 202 input edges**
Žalik's algorithm	7.16 sec	438.51 sec
Our algorithm	0.59 sec	1.58 sec

Table 2: A comparison with Žalik's algorithm

As seen from Table 3, the proposed algorithm is much faster than the referenced algorithm. In the real case, it is quicker for about 12 times, and in the artificial case, when a lot of intersections exist between the input line segments, for approximately 275 times.

5 Conclusions

In this work, an efficient algorithm for constructing topology from a set of line segments is given. The presented approach upgrades the approach given by Žalik [5], and consists from two main parts:

- detecting and removing inconsistencies from input data,
- topology construction.

The problem of determining the parent polygons to holes has been solved efficiently by applying the uniform plane subdivision. Theoretical expected time complexity has been estimated to *O(n)*, where *n* represents the number of input line segments. This time complexity has been also confirmed by experiments using real and artificial (almost the worse case) input data.

Acknowledgements:
We are grateful to the Slovenian Research Agency for supporting this research under the project Z2-6661-

0769-04/2.12 - Compression of elements appearing in the final elements methods (FEM)

References:

[1] M. Sonka, V. Hlavac and R. Boyle, *Image Processing, Analysis and Machine Vision*, Chapman & Hall Computing, 1993

[2] N. R. Chrisman, Efficient digitizing through the combination of appropriate hardware and software for error detection and editing, *Geographical Information Systems*, Vol. 1, No. 3, 1987, pp. 265-277

[3] R. Laurini and F. Milleret-Raffort, Topological Reorganisation of Inconsistent Geographical Databases: A Step Towards Their Certification, *Computer & Graphics*, Vol. 18, No. 6, 1994, pp. 803-813

[4] T. Ubeda and M. Egenhofer, Correcting Topological Errors, *In: Scholl V, editor. Advances in Spatial Databases - Fith International Symposium on Large Spatial Databases, SSD'97, Lecture Notes in Computer Sciences, 1262, Berlin: Springer-Verlag*, 1994, pp. 283-297

[5] B. Žalik, A Topology Construction From Line Drawings Using A Uniform Plane Subdivision Technique, Computer-Aided Design, Vol. 31, No. 5, 1999, pp. 335-348

[6] M. Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry, Algorithms and Applications.* Springer-Verlag, 1997

[7] F. P. Preparata and M. I. Shamos, *Computational Geometry: an Introduction, 2nd ed.* Springer-Verlag, 1998

[8] J. L. Bentley, B. W. Weide and C. Y. Andrew, Optimal Expected-Time Algorithms for the Closest Point Problems. *ACM Transaction on Mathematical Software*, Vol. 6, No. 4, 1980, pp. 563-580

[9] M. Chen and P. Townsend, Effcient and consistent algorithms for determining the containment of points in polygons and polyhedra, *In Marechal, G. (Ed.), Proceedings of Eurographics'87, Elsevier Science, Amsterdam*, 1987, pp. 423-437

[10] C-. W. Huang and T-.Y. Shih, On the complexity of point-in-polygon algorithms, *Computers & Geosciences*, Vol. 23, No. 1, 1997, pp. 109-118