The importance of computational geometry for digital cartography

Tomáš Bayer
Faculty of Science, Charles University in Prague
bayertom@natur.cuni.cz

Keywords: computational geometry, digital cartography, open source, GIS, automated generalization, convex hull

Abstract

This paper describes the use of computational geometry concepts in the digital cartography. It presents an importance of 2D geometric structures, geometric operations and procedures for automated or semi automated simplification process. This article is focused on automated building simplification procedures, some techniques are illustrated and discussed. Concrete examples with the requirements to the lowest time complexity, emphasis on the smallest area enclosing rectangle, convex hull or self intersection procedures, are given. Presented results illustrate the relationship of digital cartography and computational geometry.

Introduction

Needs of human to capture and represent surrounding landscape are very old. The first evidences can be found on the walls of caves or animal horns; they are associated with the beginnings of the cartography. Cartography is over two milenia old science, but during this period has been radically changed. Adding mathematical fundamental and analytical methods to the process of data acquisition and mapping resulted in the birth of earth sciences. Due to new knowledge in mathematics, physics, computational geometry, statistics and informatics the methods of creating maps have been rapidly modified and enforced (Kolar at al, 2008). The transformation process of analogue maps to digital maps incurred as a result of cartographic representation of the Earth based on planar structures (eg. points, lines, polygons) brought some new problems that can be effectively solved using computational geometry. In digital cartography are some new geometric structures like topological skeleton, Voronoi diagrams, Delaunay triangulation has been started to use.

From computational geometry to natural sciences

The beginnings of the computational geometry arose as a response to data acquisition and data processing techniques changes at the 60th of 20 century. Their transformation into digital form brought a new data representation of the landscape, based on its decomposition to 0D, 1D, 2D, 3D entities. The process of creating maps was associated with the lack of digital data analysis and synthesis. It led to the need of their processing with the least amount of manual interventions by an operator. A number of new techniques aimed to planar or spatial data analysis and their relationships has been created. Those exact methods were based on linear algebra, geometry, cartography, statistic or adjustment calculus.

Based on synthesis of these findings, a new field "computational geometry" has been established. The computational geometry studies features of geometry algorithms in 2D or 3D and tries to find an optimal solution for geometry problems due to the time complexity. Whereas there is a bigger amount of data we are able to process, it is necessary to solve problems effectively. Due to the difference of the cartographic and informatic look to problems, this article tries to find unifying perspective emphasing importance of the computational geometry for the cartography education. In order to the Czech Republic does not become passive consumers of information technologies, it is necessary to invest in development of own geoinformatic problems solutions. This fact plays and important role and can not be underestimated in the long term perspective.

The educational process must be adapted to those facts. It is not sufficient to focus only on practical solving of problems. Based on an analysis, the student should be able to find optimal solution for the problem. In general terms, it is necessary to strengthen the teaching of natural sciences. In today´s highly over-technized world plays the ability of exact assessment of the problem an important role. It allows to reduce an inwardness of human decision-making and a dependency on ideologies. But this concept is in the discrepancy with the requirements to practical focus of higher education. It is possible to illustrate those problems on the educational process of computational geometry with the focus on interrelationships and interdisciplinary links. Students would be able to feel the problem comprehensively and solve it much more effectively.

Computational geometry and building generalization

A map represents an abstract expression of the reality. To maintain the basic characteristics of the cartographic products (dis passionateness, clearness, lucidity...), a controlled reduction of information must be performed. This process is called "generalization" and results in the simplification of the map content. Generalization takes an important role in computer graphics, it allows to reduce the amount of information and shorten the visualization process. Generalization is a subjective process with an accent to knowledge and experiences of the cartographer. Computational geometry makes the process of simplification less dependent on a subjective view of a cartographer. An algoritmization of the simplification process is not unambiguous. It is not an easy task to find and set a geometric criterion, that is supposed to be satisfied by a simplified element. The simplification represents a process of more interdependent steps, an implementation of one step causes the next step. This

part of the article uses information and mathematical background from a new simplification algorithm proposed by the author.

Generalization factors. The are several important factors of the generalization that affect the results. They could be divided into four categories: map scale, the purpose of the map, characteristic of the territory, used cartographic symbols.

Geometric generalization of the building. The geometric generalization carries out a controlled reduction of the map content based on analysis of the geometric properties of the elements. It tries to remove those elements, that are not significant in the map context. Some geometric structures like Voronoi tessellation or Delaunay triangulation can be used. Automated or semi automated generalization of the building represents a problem solved in many ways. Commonly used simplifying algorithms can not be applied, they do not maintain internal angles of the polygon edges ($\pm\frac{\pi}{2}$) representing the building, see Fig. 1. Building simplification has the constrain that makes this process more difficult.

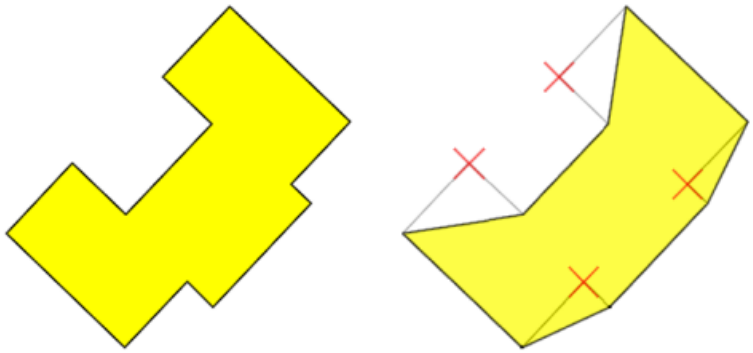

Figure 1: Building generalization without internal angles maintaining.

Requirements for the algorithm. A design of the algorithm with the reasonable time complexity (quadratic or better) providing appropriate cartographic results with minimizing the needs of manual corrections seems to be a hard problem. In addition, we have the following requirements for the simplification algorithm:

- ability of building detection and simplification in any position,
- self intersections removing,
- ability to keep the area (equal area or near o equal area algorithm),
- regulation of the simplification factor by user,
- ability to simplify complex and non-convex shapes.

In terms of computational geometry we explain more detailed the first and second points.

Scheme of the simplification process

An automated or semi automated simplification of buildings based on the least square method is currently being solved in many ways. From the cartographic perspective it provides relatively good results. The simplification process can be shortly described using the following

steps:

1. Detection of the angle of rotation φ of the building:
   - construction of the convex hull of a set of points,
   - construction of the smallest area enclosing rectangle of a set of points.
2. Set rotation of the building: the angle of rotation $-\varphi$.
3. Detection of the vertices and edges of the building based on the recursion:
   - calculation of the splitting criterion σ,
   - recursive decomposition of the edge to the set of new edges.
4. Set rotation of the building: angle of rotation φ.

In order to simplify mathematical calculations, generalized building is rotated by the angle of $-\varphi$. The building is rotated so that its edges are parallel to the axes of x, y.

Detection of the building rotation

We will consider a non convex rectangular polygon in the plane to be a building. The building usually does not have to be oriented in the *basic position*, when all edges are parallel to axis x, y of the coordinate system. In general position the building is rotated, the rotation angle φ must be detected as a first step of the simplification algorithm.

An accuracy of determining the angle of rotation φ significantly affects an effectiveness of the algorithm. The most common method of detecting the angle of rotation φ formed by x axis and the longer edge of the rectangle, is based on construction of the minimum bounding box (rectangle enclosing all points with the minimum area), and follows with the detection of the angle formed by the x axis and the longer edge of the rectangle, see Fig. 2.

Whereas, the calculation is carried out over a large set of points, it is necessary to choose the procedure with the lowest time complexity. The procedure runs over non-convex polygon, this feature makes the process more complex. Commonly available algorithms achieve quadratic time complexity $O(N^2)$ for this operation. Using rotating calipers method published in [2] we can perform this step in linear time. This procedure is usable only for convex polygons, first step represents transformation of the non-convex polygon to convex hull. Which method for the convex hull construction is the best to choose: Jarvis scan, Graham scan or QuickHull? Given the time complexity requirements as the best variant appears the Graham scan.

An interesting fact may be a comparison of the detected angle φ to street line angle constructed using the topological skeleton (eg straight skeleton). This technique is currently at the research stage.

Graham scan

Graham scan enables constructioning the convex hull in sub quadratic time with $O(N \cdot \lg N)$ complexity. It assumes, there are no three collinear points in the set. Algorithm is based on the idea of *right turn*. For each triplet P_i, P_{i+1}, P_{i+2}, $i \in 1, .., n-2$, we analyze relative

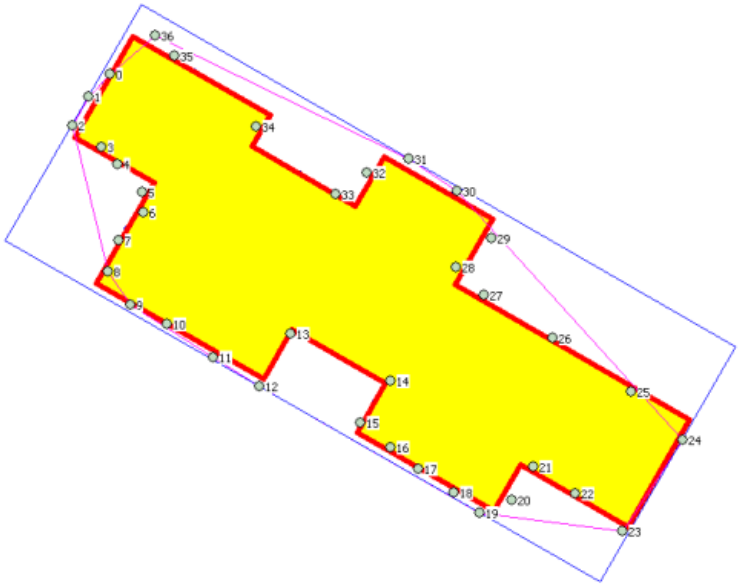

Figure 2: A detection of the building rotation using convex hull and the smallest area enclosing rectangle.

position of P_{i+2} and the segment consisting of P_i, P_{i+1} (left or right turn). Let us denote $\overrightarrow{u} = P_i - P_{i+1}$ and $\overrightarrow{v} = P_{i+1} - P_{i+2}$. Right turn criterion we can write as

$$\begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} v \leqq 0.$$

The first step consists of finding a point Q with extreme x coordinate (x_{max}). It follows with sorting of points according to the angle ω measured between $\| x$ and Q, P. When calculating the angle, it is necessary to determine ω at interval $(0, 2\pi)$. Notify, that computing angle ω from

$$\omega = \arccos\left(\frac{(x_2-x_1)(x_3-x_2)+(y_2-y_1)(y_3-y_2)}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}}\right)$$

brings numerical troubles.

Sorting algorithm. The relationship computational geometry and informatics can be illustrated by a sorting algorithm. What algorithm seems to be appropriate for sorting the set of points because of the time complexity? Given the fact, that the set of points forming a building is not too large, the choice of sorting algorithm does not play an important role. Given the fact that sorting procedure could be repeated for the data made of thousands of buildings, it is efficient, in terms of overall approach to the problem, to use QuickSort. The QuickSort implementation is available in many programming languages as a standard sorting procedure.

Data structure and implementation. A concept of the data representation also plays an important role. One possible solution using the stack can be found in [3]. Every point is represented by its unique identifying number, coordinates x, y, and flag illustrates the deletion of point from the hull. A correct definition of copy constructors and casting operators is important. Look to the following source code sample:

```
class Point
{
    private:
        int num;
        bool del;
         double x,y;
        ...
    public:
        Point::Point (const Point &point)
         {
            num=point.num;
             del=point.del;
             x=point.x;
            y=point.y;
         }
        bool Point::operator < (const Point &point)
        {
            return (y<point.y)||(x>point.x)&&(y==point.y);
        };
        bool Point::operator == (const Point &point)
        {
            return (x==point.x)&&(y==point.y);

        };
        Point Point:: operator = (const Point &point)
        {
            num=point.num;
            del=point.del;
            x=point.x;
            y=point.y;
            return *this;
        }
    ...
}
```

Collinerity problem. The collinearity problem negatively affects the process of convex hull construction. Collinear points have the same angle, how to sort those points? Let us denote two colinear points P_i, P_j and s_i, s_j euclidean distances from those points to the Q. We define a new sorting rule: if $(\omega_i = \omega_j)$ than closer point $\min(s_i, s_j)$ is considered as earlier. Coincident points represent a special case of the collinearity problem. For GIS data this problem is not so important, they are topologically valid (it means also without duplicated points).

Smallest area enclosing rectangle

Problem of the smallest area enclosing rectangle construction was solved in many ways. Presented solution described in [5] solves the problem in linear time using two calipers orthogonal to each other. The following procedure is called *Rotating calipers*. The idea of construction is based on the repeated rotation of rectangle, the rectangle is gradually improved and becomes an approximation of smallest enclosing area in the next step. One edge of the smallest enclosing box must be collinear with one segment of the convex hull.

Let us denote $\varphi_j, j \in \langle 1, 4\rangle$, four angles formed by the four smallest area enclosing box edges and four edges of the convex hull in points of contact V_j. Let V_j' represents a point, that is a successor of the point V_j, and M_j represents a vertex of the smallest area enclosing box. Vertices of the smallest area (and thus edges) are clockwise oriented.

We find the minimum angle $\varphi_{min} = \min(\varphi_j)$ and rotate the rectangle by an angle φ_{min}. Another edge of the rectangle becomes collinear with some segment of the convex hull. Three points of contacts will not change. However one point V_j, represented by the start point of the collinear segments, changes to its successor V_j'. We calculate an area S of the rectangle, compare it with a minimum area S_{min} initialized during the first step to ∞. If $S < S_{min}$, we store $S_{min} = S$. Repeat those steps until $\sum \varphi_{min} < \frac{\pi}{2}$ leads to result $\sum \varphi_{min} = \varphi$. Due to the fact, that buildings are represented by rectangular polygons, more than one edge of rectangle with more segments of the convex hull. Because of errors cumulation the numerical inaccuracy is the problem of presented algorithm.

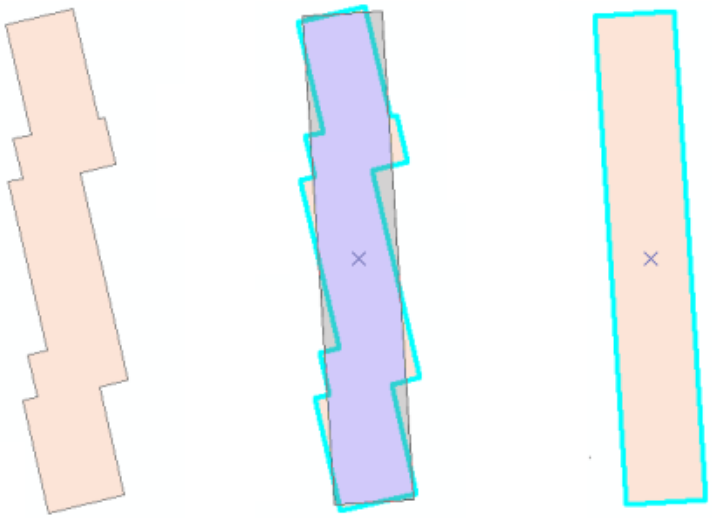

Figure 3: Problem of smallest area enclosing rectangle construction lead to inappropriate simplification.

For our purposes it is sufficient to determine the angle φ with the accuracy of one degree, therefore we do not have to deal with this problem in more detail. For some specific shapes the smallest area enclosing rectangle do not have to be the best way how to detect the rotation of the building. This situation is typical for Z or L segment, when the deviation between calculated angle φ and true value of the angle may be of up to several tens of degrees, see Fig. 3. It is important to note, that the steps above represent only auxiliary geometric construction with a certain percentage of errors.

The detection of self intersections

During the process of the cartographic generalization we can be encountered with the problem of self intersections. They represent such situations, in which some undesirable forms as a results of the generalization process have been created. Due to the topological incorrectness of such data, this error is very dangerous. Closed "pseudoregion" is the result of crossing of two or more line segments. In the locus of the intersection there is no vertex inserted. Using GIS software this pseudoregion will be considered as topologically incorrect, see Fig. 4.

One of the possible solution may be a test, which verifies an existence of self intersections. Before an edge removing or edge splitting procedure it is verified, whether this edge does not intersect any other edge of the building. If so, a procedure for the edge simplification will be

canceled. Unfortunately, this step will contribute to a significant slowdown of the algorithm. How to perform the effective detection of self intersections with better than quadratic time complexity? Bentley&Ottman algorithm brings one of possible solutions.

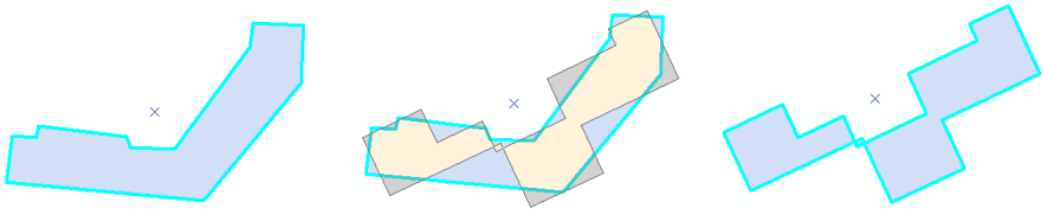

Figure 4: Problem of self intersection after the splitting procedure.

Bentley&Ottman algorithm

Bentley&Ottman algorithm, published in 1979, is able to find intersections of sets of lines with $O(N \lg N)$ time complexity. A brute force algorithm, based on checking of all possible intersection, is working only with the quadratic time complexity. Bentley&Ottman algorithm represents an application of the sweep line, moving over the lines from left to right. The sweep line parallel to y axis divides the set into processed part and unprocessed part. It calculates intersections only with those lines, that are cut by the sweep line.

Data structures. The proposed algorithm is an example of the use of the priority queue. The proposal of data structures plays an important role. The first data structure is represented by the priority queue, points are sorted according to x coordinate. Information whether this point is a start point, an end point or an intersection, are stored for each point. If sweep line moves to point, an event is called. The second data structure, often represented by the tree, stores lines in the order in which they intersect the sweep line.

Lines intersection. Finding the intersection of two lines is possible from parametric equations. Using general equation for lines parallel to x bring problems. Let us denote the first line l_1 given by two points $P_1 = [x_1, y_1]$, $P_2 = [x_2, y_2]$, the second line l_2 given by two pints $P_3 = [x_3, y_3]$, $P_4 = [x_4, y_4]$, and intersection $Q = [x_q, y_q]$. Parametric equation for the line we can write

$$\begin{pmatrix} x_q \\ y_q \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + s \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} = \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} + t \begin{pmatrix} x_4 - x_3 \\ y_4 - y_3 \end{pmatrix},$$

where

$$s = \frac{y_1(x_3-x_4)+y_3(x_4-x_1)+y_4(x_1-x_3)}{(x_2-x_1)(y_3-y_4)-(y_2-y_1)(x_3-x_4)}, t = \frac{y_1(x_3-x_2)+y_2(x_1-x_3)+y_3(x_2-x_1)}{(x_2-x_1)(y_3-y_4)-(y_2-y_1)(x_3-x_4)}.$$

For $s \in (0,1) \cap t \in (0,1)$ the intersection could be found from previous formulas.

Intersection of segments. The sweep line moves over the segments and stops at events of three types: (1) start point of the segment, (2) intersection point between two segments, (3) end point of the segment, see Fig. 5. If the event point represents start point, we test segment against two neighbors along the sweep line. If the event point represents end point, point is removed from the list. If we found an intersection of those segments, it becomes a new event point. If the event point represents an intersection of two lines, we change their

order. Each of both segments has adjacent segments along the sweep line, that must be tested for intersections. If the point represents an end point, adjacent segments are tested for intersection and event point is removed. Bentley&Ottman algorithm is based on assumption, that no segment is parallel to sweep line and no three segments pass through one point.

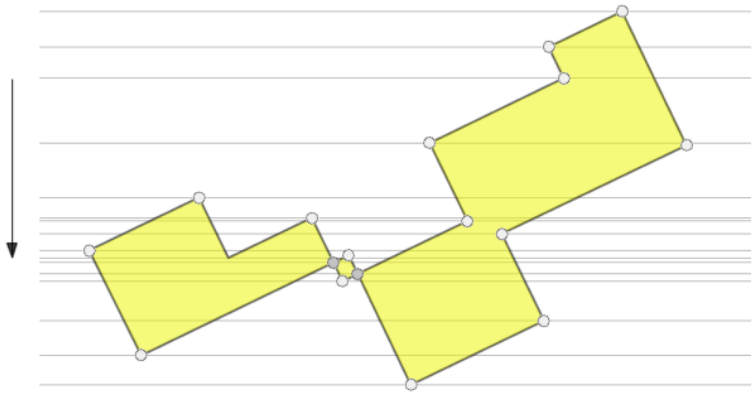

Figure 5: Bentley&Ottman algorithm with positions of sweep line.

History of segments intersecting the sweep line is stored in balanced binary tree. This data structure is very efficient and enables update operations in $O(\lg(N))$ time. So, it is apparent, that the implementation of Bentley&Ottman algorithm looks quite difficult, and uses a combination of several dynamic data structures.

Conclusion

This paper presents the use of computational geometry in digital cartography. As an illustrative example the process of automated or semi automated building simplification was chosen, several examples were given and discussed. It was focused on the idea of possibility of more intensive computational geometry teaching. This article tries to find unifying perspective emphasis importance of the computational geometry for cartography education. Not to become only passive consumers of information technologies, it is necessary to invest in the development of own geoinformatic solutions. This fact plays and important role and can not be, as mentioned above, underestimated in the long term perspective.

References

1. DE BERG M., SCHWARZKOPF O., KREVELD M., OVERMARS M.: Computational geometry: Algorithms and applications, 2000, Springer-Verlag.
2. DUTTER M.: Generalization of buildings derived from high resolution remote sensing data, 2007.
3. ROURKE O. J.: Computational geometry in C, 2005, Cambridge University Press.
4. SESTER M.: Generalization based on least square adjustment, International Archieves of Photogrammetry and Remote Sensing, 2000.

5. TOUSSAND G., Solving Geometric Problems with the Rotating Calipers, McGill University Montreal, 1983