Automated building simplification using a recursive approach

Tomas Bayer | bayertom@natur.cuni.cz

Department of Applied Geoinformatics and Cartography, Faculty of Science, Charles University in Prague, Albertov 6, 128 48, Praha 2, Czech Republic.

Abstract

This article presents a new generalization algorithm for automated or semi automated buildings simplification based on the recursive approach. From the data set with a higher level of detail another data set that containing a lower level of detail can be derived. The proposed algorithm has an ability to detect and simplify buildings rotated in random position and does not depend on ordering of points. To determine the angle of rotation φ of the building, a smallest area enclosing rectangle is constructed. The splitting procedure is performed on the basis of splitting criterion σ calculated for each edge of the building. The simplified edges are replaced with regression lines, whose parameters are determined using the least squares method. The algorithm was implemented in C++. With the use of ArcObjects libraries, a new DLL application designed for ArcGIS 9.x was created. This tool is applicable to the polygonal shape files.

Keywords: digital cartography, automated simplification, buildings generalization, recursive algorithm

1 Introduction

A geometric generalization carries out a controlled reduction of the map content based on the analysis of the geometric properties of elements. It tries to remove those elements that are not significant in the map context. The generalization is a subjective process with an accent to knowledge and experience of the cartographer. The computational geometry makes a process of the simplification less dependent on a subjective view of the cartographer. But the algoritmization of the simplification process is ambiguous. It is not an easy task to find and set a geometric criterion that should be satisfied by a simplified element.

Automated or semi automated generalization is currently being solved in many ways. Commonly used simplifying algorithms can not be applied, they do not maintain internal angles ($\pm\frac{\pi}{2}$) formed by the adjacent polygon edges representing the building and parallelism. Thus, the building simplification has some constrains that make this process more difficult.

2 Related work

The cartographic generalization represents a process of deriving a cartographic product with a lower level of detail from a source (and more detailed) cartographic product. The resulting map should be easy readable, clearly decodable with aesthetic pleasing (Weibel et al. 1998). The simplification represents a process of more interdependent elementary geometric and graphic operations, an implementation of one step conditionally causes the next step. This idea was mentioned by McMaster (1987) and further developed. A decomposition of the generalization process into sub-processes using generalization operators was introduced by McMaster & Shea (1992), Weibel (1997), Weibel & Dutton (1999). The concept of Elementary Generalization Operations (EGO) described by Sester & Brenner (2004) defines a set of Simple Operations (SO) and rules that can be applied to the geometric structures. Some of these elementary operations (i.e. insert/remove vertex) and concepts are used by the proposed algorithm.

A simplification of the polygon boundary represents one of the generalization operators used in categorical generalization (Galanda 2003). Douglas-Peucker Algorithm (Douglas & Peucker 1973), one of the most frequently used methods for line generalization, has been repeatedly modified: non self intersect version (Wu T. S. et al. 2003) or polygon version (Guibas et al. 1993).

A concept of the automatic building generalization based on removing or resolving short edges was published by Staufenbiel (1973). Kanani (2000) presented modified Douglas-Peucker algorithm for buildings. Sester (2000) published a building simplification method based on the least squares adjustment. A recursive algorithm for the building approximation was developed by Gross et al. (2005), a recursive algorithm based on modified Douglas-Peucker algorithm using the least squares

adjustment was implemented by Dutter (2008). The proposed algorithm tries to improve the ability to simplify rectangular buildings of complex geometric forms. The process of the simplification will be significantly modified due to a request for the recursive solution of the problem. From the cartographic perspective it provides relatively good results, see Chapter 6. Haunert & Wolf (2008) brought an interesting solution using graph algorithms, their implementation is based on the heuristic solution. Yan et al. (2008) introduced an approach to automated building grouping and generalization based on Voronoi diagram and topological adjacency.

3 Basic concept

A map represents an abstract expression of the reality. To maintain the basic characteristics of the cartographic outputs, a controlled reduction of information must be performed. This process results in a simplification of the map content. The generalization takes an important role not only in cartography but also in computer graphics. It allows to reduce the amount of information and to shorten the visualization process. Some basic concepts can be defined as follows.

Building definition. Let us consider a non convex rectangular polygon in the plane to be a building. This polygon is bounded by a finite collection of n line segments. Each segment e_i has two vertices $P_i, P_{i+1}, i \in \langle 1, n-1 \rangle$. Let $P_1, P_2, ..., P_n$ be n points representing vertices of the polygon and e_i, e_{i+1} two adjacent and perpendicular line segments. 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, the position of the building is rotated. The angle of rotation φ must be detected as a first step of the simplification algorithm. An initial boundary edge consists of all points of the building.

Building simplification. The aim of the simplification process is to find another building representation based on a rectangular polygon of the simpler geometric form (in general non convex). The geometric form of the resulted polygon will be affected by the location of set of vertices of the simplified. During the simplification process such edges, that are not significant in the map context, are removed. Simplified edges are replaced with the set of new edges of a simpler form, whose parameters are determined using the least squares method.

Splitting criterion σ. For an assessment of the geometric complexity of the edge a criterion based on the standard deviation σ is used. This criterion is calculated repeatedly for each detected edge of the building. It is compared with a maximum value of the criterion σ_{max}. Value of the criterion depends on the geometric complexity of the edge, for more complex forms becomes of greater values. Until $\sigma > \sigma_{max}$, each edge is recursively decomposed to the sets of new edges. However σ_{max} represents a simplification criterion regulated by a user.

3.1 Scheme of the simplification process

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

- the ability to simplify a rotated building,
- self intersections removing,
- the ability to keep an area of the building,
- the regulation of the simplification factor by a user,
- ability to simplify non-convex shapes with more complex geometry.

In terms of computational geometry these points will be explained in more detail.

Scheme of the algorithm. The simplification process can be shortly described using the following steps:

Algorithm 1 Scheme of the simplification process.

1: Detection of points orientation, see Chapter 4.1.
2: Detection of the angle of rotation φ of the building:
3: Construction of the convex hull of the set of points, see Chapter 4.2.1.
4: Construction of the global smallest area enclosing rectangle of the set of points, see Chapter 4.2.2.
5: Counterclockwise rotation of the building (the angle of rotation $-\varphi$), see Chapter 4.3.
6: Detection of the vertices and edges of the "no-rotated" building using the recursion:
7: Insert initial boundary edge into stack S, see Chapter 4.4.
8: Processing all edges, see Chapter 4.4.
9: Reconstruction of the simplified building, see Chapter 4.5.
10: Clockwise rotation of the building (the angle of rotation φ), see Chapter 4.6.

In order to simplify mathematical calculations, a generalized building is firstly rotated by the angle of $-\varphi$ (the building is rotated so that its edges are parallel to the axes of x and y). In this position, all steps of the simplification process are carried out. Finally, the simplified building is rotated to the starting position by the angle of φ.

3.2 Hierarchical detection of vertices

The presented algorithm is based on the idea of hierarchical detection of vertices. Vertices are detected on the basis of their geometric significance. New vertices are gradually added to the building, among them edges of the building are geometrically reconstructed. The similar approach uses Douglas & Peucker algorithm and for buildings also algorithm presented by Dutter (2008).

The idea of the algorithm is based on repeated search for such pairs or two pairs of points, that are located *closest* to the vertices of the smallest area enclosing rectangle, constructed over every edge. Those points will represent new vertices of the building in the next step. All points P_i, which are located "between" two adjacent vertices, are matched all in all to the new edge. This step is similar to Dutter (2008) , but the process of the detection and results are different.

A replacement of detected edges. Each detected edge of the buildings will be replaced by the set of new edges of simpler geometric forms. Edges of such simple geometric forms, that can not be further split, will replaced by regression lines. Neighboring detected edges are perpendicular to each other. Splitting edges will be processed using recursive approach in the same way.

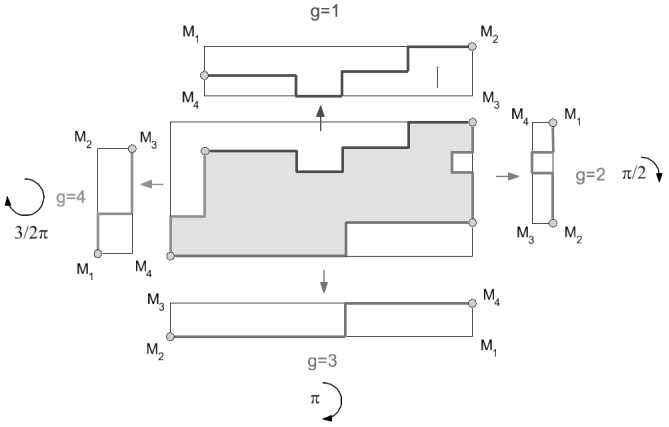

Figure 1: The global and four local smallest area enclosing rectangles with the rearrangement of the vertices.

Detection of the vertices. As mentioned above, we will consider a non convex rectangular polygon in the plane to be a building. The process of detection of the vertices is based on repeated searching for such pair or two pairs of vertices N_j, $j \in \langle 1,4 \rangle$, $N \subset P$, that are closest to one pair or two pairs of vertices M_j of the smallest area enclosing box. Let M_1 represents a point with coordinates $[\min(x_i), \max(y_i)]$.We suppose, that vertices M_1, M_2, M_3, M_4 of the smallest area enclosing rectangle (and thus edges e_1, e_2, e_3, e_4) are *clockwise oriented*. The process of detection over each edge represents analogy to the situation over the first edge rotated by the angle of $\frac{\pi}{2}, \pi, \frac{3}{2}\pi$, see Figure 1. This problem is, instead of the

rotation of the constructed smallest area enclosing box, also easily solvable by changing an ordering of formal parameters (eg vertices). Let us use this idea for a further development of the algorithm.

Change ordering of formal parameters. If some detected edge of the building is *parallel* and has the *same direction* (e.g. it is same orientation) to other edge of the smallest area enclosing rectangle than the first edge (formed by vertices M_1, M_2), we formally rearrange ordering of the smallest area enclosing box vertices (and thus detected vertices of the building).

Depending on the orientation of the detected edge of the building to the orientation of the edge of the smallest area enclosing box constructed over the whole building (i.e. *global enclosing rectangle*) a set of the vertices $\{M_1, M_2, M_3, M_4\}$ of the smallest area enclosing rectangle over the detected edge (i.e. *local enclosing rectangle*) is rearranged to another set, see Table 4.

Table 1: An rearrangement of vertices of the local enclosing rectangle depending on the orientation of the detected edge e to the global enclosing box.

Detected edge e, orientation same as	$g^{(1)}(e)$	Rotation	Set of vertices	Vertices of the building
1. edge	1	0	$\{M_1, M_2, M_3, M_4\}$	$\{N_1, N_2, N_3, N_4\}$
2. edge	2	$\frac{\pi}{2}$	$\{M_4, M_1, M_2, M_3\}$	$\{N_4, N_1, N_2, N_3\}$
3. edge	3	π	$\{M_3, M_4, M_1, M_2\}$	$\{N_3, N_4, N_1, N_2\}$
4. edge	4	$\frac{3}{2}\pi$	$\{M_2, M_3, M_4, M_1\}$	$\{N_2, N_3, N_4, N_1\}$

So, we are working with a local enclosing rectangle and rearranging its vertices in accordance with the orientation of the corresponding edge of the global enclosing rectangle.

Vertices of the buildings. Let N_j, $N_j \subset P$, represent four (possible) vertices of the building closest to points M_j. Then point N_1 is the closest to the point M_1, etc., see the Table 4. For the global enclosing rectangle all four vertices (2 pairs) will be searched, for the local enclosing rectangle only one vertex or 2 vertices will be searched.

Remark: For Euclidean distance d and index k, $k = 1, 2, 3, 4$, where $k \neq j$, we suppose $d(M_j, N_j) < d(M_j, N_k)$ and $d(M_j, N_j) < d(M_k, N_j)$. In some cases the conditions may not be satisfied. Case one, $d(M_j, N_j) = d(M_k, N_j)$, represents one point N_j closest to the two different vertices M_j, M_k of the enclosing box. Case two, $d(M_j, N_j) = d(M_j, N_k)$, represents two vertices N_j, N_k closest to one point M_j of the enclosing box.

3.3 Orientation of vertices and edges

This algorithm keeps information about the orientation of each edge of the building to the edge of the global smallest area enclosing rectangle at any moment. Let g,$g = 1, 2, 3, 4$, represents an orientation of the building segment e to corresponding edge of the enclosing rectangle. As mentioned above, edges of the enclosing rectangle are *clockwise* arranged. Orientation g of the segment e depends on the orientation of its first point N denoted as $g(N)$. This orientation is subsequently set for all points P_i lying within the segment e. After finding four vertices M_j of the smallest area enclosing rectangle and four closest vertices N_j of the building the simplified building is split into four edges e_1, e_2, e_3, e_4. We set $g(e_1) = 1$, $g(e_2) = 2$, $g(e_3) = 3$, $g(e_4) = 4$. For details see Chapter 3.3.1.

Depth of recursion. Let us denote r the depth of the recursion. For any depth of recursion, we are able to determine an orientation of the segment of the building e to corresponding edge of the smallest area enclosing rectangle constructed in this recursion depth.

The relationship $g^{(r)}(N)$ to $g^{(r-1)}(N)$. Let $g^{(r-1)}$ represents a cumulated orientation of the "parent" edge in the last recursion depth $r-1$, let $g^{(r)}$ represents a cumulated orientation of any "child" edge (i.e. actual edge resulting from a split of the parent edge) in the actual depth of recursion r, and let g^{act} represents an orientation of the actual edge in the actual depth of recursion r to the corresponding edge of the local smallest area enclosing rectangle. The relationship $g^{(r)}(N)$ to $g^{(r-1)}(N)$

$$g^{(r)}(N) = g^{(r-1)}(N) + g^{(act)}. \tag{1}$$

How to determine orientation of the segment in any recursion depth to corresponding edge of the (global) smallest area enclosing rectangle, constructed over all points P of the building (it means in the first recursion depth $r = 1$)? We can write

$$g^{(1)}(N) = g^{(r)}(N) mod(4) - (r-1) mod(4). \tag{2}$$

If $g^{(1)}(N) = 0$, then $g^{(1)}(N) = 4$.

$$g^{(1)}(N) \begin{cases} = 0, & g^{(1)}(N) = 4, \\ \neq 0, & g^{(1)}(N) = g^{(1)}(N). \end{cases}$$

Formula (2) is calculated repeatedly for each detected edge. For the sample of values $g^{(r)}$ in recursion depths $r = 1, 2, 3, 4$, see Table 2.

Table 2: Values $g^{(r)}$ *for recursion depths* $r = 1, 2, 3, 4$.

$g^{(1)}(N)$	$g^{(2)}(N)$		$g^{(3)}(N)$			$g^{(4)}(N)$			
1	2	6	3	7	11	4	8	12	16
2	3	7	4	8	12	5	9	13	
3	4	8	5	9		6	10	14	
4	5		6	10		7	11	15	

Remark: If two orientations $g_1^{(r)}$ and $g_2^{(r)}$ satisfy the following condition $g_1^{(r)} = g_2^{(r)}$, values $g_1^{(1)}$ and $g_2^{(1)}$ may not be equal, providing the condition $r_1 \neq r_2$. This feature shows the need and importance of maintaining the depth of recursion r as a separate variable. During the process of the proposal of the class structure, it is necessary to take this fact into account.

3.4 Splitting criterion σ

The proposed algorithm performs recursive splitting of the edge depending on its geometric complexity. The splitting criterion σ is calculated for each detected edge of the building. It is based on the calculation of standard deviation σ, that minimizes the sum of the squares of the points distances from the regression line; the variance of the residuals is the minimum possible. The regression line is oriented in accordance with one edge of the smallest area enclosing rectangle, it is parallel to this edge (and depends on the value of $g^{(1)}$ of this edge). In this case, there is no need to determine the angle of the regression line. The regression line also passes through the center of gravity of the set of points. For a construction of the regression line, it is necessary to know the value of $g^{(1)}$ given by (2). Depending on the value of $g^{(1)}$ the splitting criterion could be rewritten as

$$g^{(1)} = \begin{cases} 1, 3: & \sigma = \sqrt{\frac{\sum_{i=1}^{n}(y-y_T)^2}{n}}, \\ 2, 4: & \sigma = \sqrt{\frac{\sum_{i=1}^{n}(x-x_T)^2}{n}}, \end{cases} \tag{3}$$

where n represents number of points of the simplified edge e and $T = [x_T, y_T]$ is the center of gravity of this edge. In the first case, σ is only a function of the variable y, in the second case only a function of the variable x. For each edge the coordinates x_T, y_T of the center of gravity usable to its further reconstruction are stored. An interesting opportunity for further research brings adding the dependency of σ on the map scale.

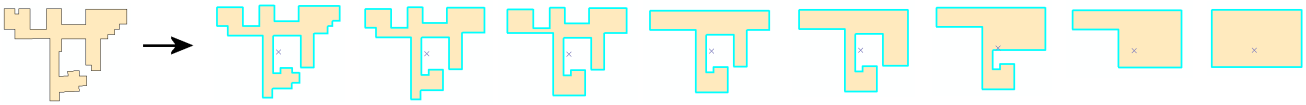

Figure 2: A building with complex geometric form, results of the building simplification depending on σ*: source building,* $\sigma = 4m$*,* $\sigma = 6m$*,* $\sigma = 8m$*,* $\sigma = 12m$*,* $\sigma = 16m$*,* $\sigma = 20m$*,* $\sigma = 24m$*,* $\sigma = 28m$*.*

Because of the reduction of the time complexity, a procedure with linear time complexity $O(N)$ is used. Coordinates $[x_T^{(i)}, y_T^{(i)}]$ of the center of gravity for i points can be determined using coordinates $[x_T^{(i-1)}, y_T^{(i-1)}]$ of the center of gravity for $i - 1$ points using

$$x_T^{(i)} = \frac{1}{i}[(i-1)x_T^{(i-1)} + x_i], \tag{4}$$

$$y_T^{(i)} = \frac{1}{i}[(i-1)y_T^{(i-1)} + y_i]. \tag{5}$$

Rearranging (3) with respect to (4) brings

$$\sigma = \sqrt{\frac{k_1 + k_2 + k_3}{n}}. \tag{6}$$

For the orientation $g^{(1)} = 2, 4$

$$\begin{aligned} k_1 &= n(x_t^{(n)})^2, \\ k_2 &= -2(x_T^{(n)}) \sum_{i=1}^{n} x_i, \\ k_3 &= \sum_{i=1}^{n} x_i x_i. \end{aligned} \tag{7}$$

Coefficients for y coordinates could be derived analogously using a substitution of x for y. Coefficient k_1 is determined only once after entering the last point. From coefficient k_2 we calculate repeatedly only the sum, coefficient k_3 must be determined continuously.

Results of the simplification process depending on splitting criterion σ are given in Figure 2.

4 Simplification algorithm

In this chapter some important steps of the simplification process will be described.

4.1 Points orientation

We assume clockwise and cycling ordering of the points P_i representing vertices of the polygon. The first step of the algorithm represents an ordering test, based on the formula calculating the area A of the polygon

$$A = \frac{1}{2} \sum_{i=1}^{n} x_i (y_{i+1} - y_{i-1}). \tag{8}$$

If

$$A \begin{cases} > 0, & \text{points are sorted clockwise,} \\ < 0, & \text{points are sorted counterclockwise.} \end{cases}$$

Ordering of points affects the way, in which points are added to the initial edge. If points are sorted clockwise, they are added at the end of the list using the `push_back()` method. Otherwise they are added to the top of the list using the `push_front()` method. An estimation of the time complexity is $O(N)$.

4.2 Building rotation

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), see Figure 3. An approach based on statistical weighting introduced Bader (2000), a solution using the wall average presented Duchene et al. (2003). Whereas, the calculation is carried out over a large set of points, it is necessary to choose the procedure with low time complexity.

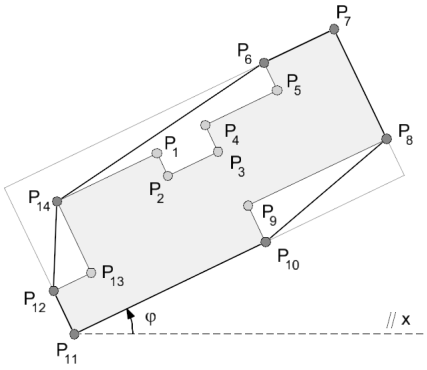

Figure 3: A determination of the building rotation using the smallest area enclosing rectangle.

The procedure runs in general over a non-convex polygon and makes the process more difficult. Commonly available algorithms achieve quadratic time complexity $O(N^2)$ for this operation. Using rotating calipers published in Toussand (1983) we can perform this step in sub quadratic time. However this procedure is usable only for convex polygons, it will use a convex hull of the building.

4.2.1 Convex hull

Graham scan enables the construction of the convex hull in sub quadratic time of $O(N \lg(N))$. It assumes, there are no three collinear points in the set. This algorithm is based on the idea of the "right turn". For each triplet P_i, P_{i+1}, P_{i+2}, $i \in 1, .., n-2$, we analyze relative position of the point P_{i+2} and the segment formed by points P_i, P_{i+1} (left or right turn). Let us denote $\overrightarrow{u} = P_i - P_{i+1}$ and $\overrightarrow{v} = P_{i+1} - P_{i+2}$. The right turn criterion we can write as follows

$$\begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} < 0. \tag{9}$$

A collinearity problem negatively affects the process of construction of the convex hull. This problem is discussed in detail in Rourke (2005), pp. 75-85. However a problem of coincident points (a special case of collinearity) is not significant for GIS data, they are in general topologically valid (without duplicated or coincident points).

4.2.2 Smallest area enclosing rectangle

The presented solution described in Toussand (1983) solves the problem using two calipers orthogonal to each other in linear time. An idea of the construction is based on the repeated rotation of the rectangle. This rectangle is gradually improved and becomes an approximation of the smallest area enclosing rectangle in the next step. At least one edge of the smallest area enclosing rectangle must be collinear with at least one segment of the convex hull. Let us denote φ_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. Each point of contact V_j lies on edge formed by points M_j, M_{j+1}. Let $P_j^s = suc(V_j)$ represents a point, that is a successor of the point V_j, and M_j represents a vertex of the smallest area enclosing box.

A modified version of the rotated calipers algorithm can be formally described as follows:

Algorithm 2 RotatingCalipers (input: points of contact V_j and vertices M_j).

1: Initialize $\varphi = 0$, $A_{min} = \infty$, $W_j = M_j$.
2: Find 4 successors P_j^s of points V_j, $P_j^s = suc(V_j)$.
3: while $(\varphi < \frac{\pi}{2})$:
4: Calculate 4 angles φ_j between line (W_j, W_{j+1}) and line (V_j, P_j^s).
5: Find $\varphi_{min} = \min(\varphi_j)$ and remeber corresponding point of contact and its successor as V_i, P_i^s.
6: Using V_i, P_i^s and other three V_j, P_j^s calculate vector equations of edges of the new enclosing rectangle.
7: Using vector equations of edges calculate 4 new vertices W_j (i.e. intersections) of the enclosing rectangle.
8: If $(\varphi_j = 0)$ or $(\varphi_j = \varphi_{min})$ find new point/points of contact and successor/successors. Repeat:
9: increment: $V_j = P_j^s$, $P_j^s = suc(P_j^s)$,
10: until line (M_j, M_{j+1}) and line (V_j, P_j^s) are colinear.
11: Area A of the enclosing rectangle formed by 4 vertices V_j.
12: Increment angle: $\varphi = \varphi + \varphi_{min}$.
13: if $(A < A_{min})$:
14: Assign: new vertices of the enclosing box $M_j = W_j$, $A_{min} = A$.
15: Remeber building rotation $\varphi_{res} = \varphi$.

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 P_j^s. We calculate an area A of the rectangle, compare it with a minimum area A_{min} initialized during the first step to ∞. If $A < A_{min}$, we store $A_{min} = A$. Repeat those steps until $\sum \varphi_{min} < \frac{\pi}{2}$ leads to result $\varphi = \sum \varphi_{min}$, see Figure 4.

Due to the fact, that buildings are represented by rectangular polygons, in general more than one edge of the rectangle is collinear with more segments of the convex hull. Angles φ_j formed by those segments are equal to zero. In this case more than one point of contact V_j changes to its successor V_j^s. It implies the following conclusion: if a tested segment of the convex hull is collinear with some edge of the rectangle, we assign $V_j = P_j^s$ and continue to the first non collinear segment. We repeat this step for all collinear segments (see points 8-10). However there may be such situation, when all four segments are collinear.

Due to the knowledge of points M_j for the last rectangle and points of contact V_j for the new rectangle, we are able to determine coordinates M_j of the new rectangle using analytical geometry. Because of cumulative errors cumulation a problem of the presented algorithm is the numerical inaccuracy.

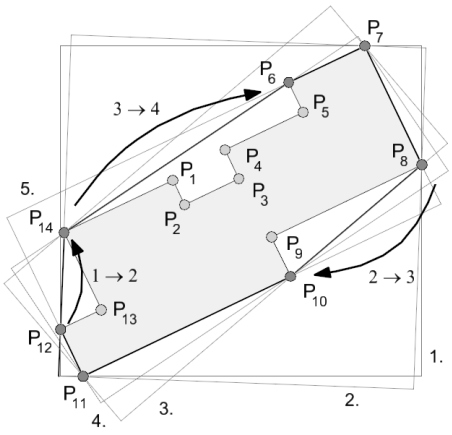

Figure 4: A construction of the smallest area enclosing rectangle using rotating calipers (phases are numbered). Changes of the point V_j to its successor V_j' are marked with arrows.

4.3 Rotation of the building to the basic position

The building is rotated by the angle of -φ so that its edges are parallel to the axes of x and y (counterclockwise rotation about the origin). Let the point P has coordinates $[x, y]$ in the unrotated coordinate system and coordinates $[x', y']$ in the rotated coordinate system. Then

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{10}$$

4.4 Processing edges

The proposed algorithm has the ability to simplify buildings of any geometric form with the order of hundreds of vertices, while the algorithm presented by Dutter (2008) some buildings with more complex geometry leaves unprocessed (see the p. 58). The simplification process represents an application of the recursive approach, a recursive processing is realised using the stack S. The first step of the splitting procedure represents insertion of an initial edge e containing all points of the building into stack S.

The splitting procedure can be described as follows:

Algorithm 3 Processing edges.

1: Set a depth of recursion: $r = 1$.
2: while the stack S not empty:
3: Pop an edge e from the stack S.
4: Get orientation $g^{(r)}(N)$ of an edge e and a recursion depth r.
5: Calculate orientation $g^{(1)}(N)$ of an edge e, see Chapter 3.3.
6: Calculate σ for the edge e, see Chapter 3.4.
7: If e represents initial edge:
8: Split e into four new edges e_1, e_2, e_3, e_4, see Chapter 4.4.1.
9: Else if $\sigma > \sigma_{max}$ and e has at least 3 vertices:
10: Recursive splitting of the edge e, see Chapter 4.4.2.
11: Else:
12: Send edge e to the output data structure (list L).

The recursive splitting runs until the stack is empty. It consists of several steps, that are repeated while edge e satisfies the following condition: $\sigma > \sigma_{max}$ and has at least 3 vertices. So the procedure performing the recursive splitting of each edge

has two parameters. The first one, σ_{max}, represents the simplification *sensitivity* and could be set by a user. The second one, enables or disables splitting of the edge formed by two points.

This step can be formally described as follows: An edge on the top is removed from the stack and split into several new edges, if its geometric form is considered as too complex. So, one edge is replaced by more new edges perpendicular to each other. However edges of the simple form are not split but sent to the output. Due to the recursion it is quite difficult to determine a time complexity of the proposed algorithm, but having regard to divide and conquer algorithm it can be estimated as $O(N \cdot \log(N))$.

4.4.1 Recursive splitting of the initial edge

The first step of the simplification algorithm replaces the polygon with the rectangle formed by four edges. These four edges are subsequently pushed into a stack S. Splitting of the initial edge e formed by the polygon, into four new edges e_j (i.e. e_1, e_2, e_3, e_4) could be decribed as follows:

Algorithm 4 Recursive splitting of the initial edge e.

1: Find 4 vertices N_j closest to 4 vertices M_j of the global smallest area enclosing rectangle over e, see Chapter 3.2.
2: If vertices N_j clockwise ordered in edge e:
3: Set orientation $g^{(r)}(N) = g^{(r-1)}(N) + j$ for all 4 detected vertices N_j of the edge e.
4: Split e into four new edges e_j, each edge e_j contains all points between vertices N_j and N_{j+1}.
5: Set orientation $g^{(r)}(N)$ from 3) of the first vertex of the split edge for all points of this edge.
6: Store r for each edge e_j.
7: Push edges e_j into stack S.
8: Else replace a building by the smallest area enclosing rectangle:
9: Create edges e_j from detected vertices N_j.
10: Send edges e_j to the output data structure (list L).
11: Increment $r = r + 1$.

During this step four points N_j closest to four vertices M_j of the global smallest area enclosed rectangle, constructed over this edge, are found. Each point N_j (i.e. new vertex of the building) corresponds with some point P_i, so N_j represents a pointer to P_i. The algorithm also tests, if all points $N_j \Rightarrow P_i$ are clockwise ordered in the polygon. If there is no such ordered set of points, the splitting procedure is stopped and the building is replaced by the enclosing rectangle. Such polygons do not often appear (except of some modern buildings), however the algorithm must be prepared for their occurrence and stops the splitting procedure.

Orientation of new vertices and new edges. The algorithm keeps information about the orientation of the vertices $g^{(r)}$ in each recursion depth r calculated from (2). As mentioned above, an orientation $g^{(r)}$ of the first vertex of the split edge is subsequently set for all points P_i lying within the split edge e. Thus all points belonging to one edge have the same orientation (i.e. we set an orientation $g^{(r)}(N)$ for this edge).

During this step it is not necessary to determine the criterion σ, an initial edge is divided into four new edges in any case. Presented method has the advantage, that orientation of each edge can be determined from the orientation of its first point. Each vertex of an initial edge becomes a start point of one new edge, and at the same time an end point of one new edge adjacent to previous edge. Four new edges e_1, e_2, e_3, e_4 are subsequently added to the stack.

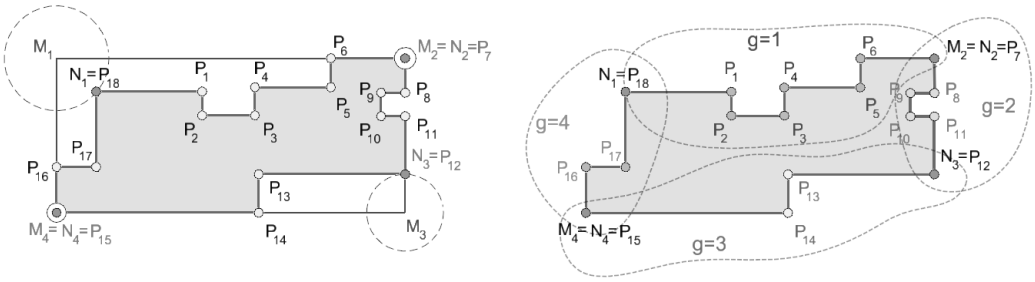

Figure 5: Recursive splitting of the initial edge: finding closest vertices N_j. All points between each pair of vertices are assigned to a new edge.

It should be noted, that the first point P_1 of the edge e may not be at the same time a vertex of the new edge. When setting an orientation, it is necessary to go through some parts of the points list repeatedly.

Orientations of four new edges e_1, e_2, e_3, e_4, generated as a result of the splitting process for initial edge e representing by the building in Figure 4, the recursion depth $r = 1$, are shown in Table 3.

4.4.2 Recursive splitting of other edges

The splitting procedure for other edges of the building is based on the same idea as the splitting procedure for an initial edge. However, it is accompanied by several steps. Recursive splitting of such edge is not always done but only in cases when the geometric form of the edge is more complex. For an assessment of the geometric complexity a criterion based on a standard deviation σ is used. This criterion is called as "*splitting criterion*". The splitting criterion σ is calculated repeatedly for each edge of the building, and assessing distances of the points to the regression line. As mentioned above, until $\sigma > \sigma_{max}$, such edge is recursively split into sets of new edges. Having regard to the brevity, only the most important steps and options of the splitting procedure will be presented.

Temporary edges. A split edge is intersected by the regression line at least in one point (except the case of collinearity). Such edge is decomposed to several new edges, whose end points are intersections of this simplified edge and the regression line. Those edges are called *temporary edges*. An idea of the splitting procedure is very simple. Each temporary edge is processed separately. All temporary edges are recursively split and replaced by several edges of the simpler form.

Table 3: Orientation of four new edges e_1, e_2, e_3, e_4 for $r = 1$.

e_1	P_{18}	P_1	P_2	P_3	P_4	P_5	P_6	P_7
$g^{(1)}(P_i)$	1	1	1	1	1	1	1	1
e_2	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}		
$g^{(1)}(P_i)$	2	2						
e_3	P_{12}	P_{13}	P_{14}	P_{15}				
$g^{(1)}(P_i)$	3	3	3	3				
e_4	P_{15}	P_{16}	P_{17}	P_{18}				
$g^{(1)}(P_i)$	4	4	4	4				

Splitting procedure. A splitting procedure for an edge e can be described as follows:

Algorithm 5 Recursive splitting of the edge e.

1: Intersection of the regression line with orientation $g^{(1)}(N)$ and the splitted edge e.
2: Create set of k temporary edges $\{t\}$.
3: For every edge t_k having at least 2 points, orientation $g^{(1)}(N)$ and recursion depth r split t_k :
4: Find a position of the temporary edge t_k to the regression line (left or right).
5: Construct the local smallest area enclosing rectangle over t_k depending on $g^{(1)}(N)$.
6: Find vertices N_j (one or two) closest to vertices M_j (one or two) of the local smallest area enclosing rectangle, see Chapter 3.2 and Chapter 4.4.3.
7: Set orientation $g^{(r)}(N)$ for all detected vertices N_j of the temporary edge t_k, see Chapter 3.2 and Chapter 4.4.3.
8: Split e into set of new edges $\{e_n\}$., the each new edge contains all points P_i lying between vertices N_j and N_{j+1} detected under each t_k at 6), see Chapter 4.4.3.
9: For every new edge e_n:
10: Set orientation $g^{(r)}(N)$ (from 7) of the first vertex of the new edge e_n for all points of this edge.
11: Store r for the new edge e_n.
12: Add the new edge e_n to the stack S.
13: Increment $r = r + 1$.

Each edge e having at least 3 points is tested whether it intersects the regression line. A list of its intersections is stored for each edge. Each intersection point stores a pointer to the first point of the edge behind the intersection. A temporary edge t_k is constructed from all points located between two adjacent intersections. However none of intersections is added to the temporary edge.

Remark: In some cases (e.g. temporary edge formed by only one point P_i), it is necessary to correct subsequently a form of the temporary edge. For example, such triangle, represented by point P_i, intersection of the segment P_i, P_{i+1} with the regression line and an orthogonal projection of P_i to the regression line, is replaced by the rectangle with the same area.

4.4.3 Splitting of temporary edges

This procedure represents an important step of the algorithm and significantly affect its effectiveness. One simplified edge may arise at least two new temporary edges. During the recursive subdivision of each temporary edge the local smallest area enclosing rectangle over this edge is constructed. As mentioned above, every split edge of the building has the same orientation $g^{(1)}$ as one edge of the local smallest area enclosing rectangle. Unlike the case of finding the global smallest area enclosing rectangle for an initial edge, only one point or one pair of points closest to one vertex or one pair of vertices, are searched.

First we determine the location of the temporary edge t_k with respect to the regression line passing through the center of gravity $[x_T, y_T]$ of the temporary edge. The temporary edge t_k can be found left or right to the regression line or it may be collinear. In the third case end points of such temporary edge represents new vertices of the building.

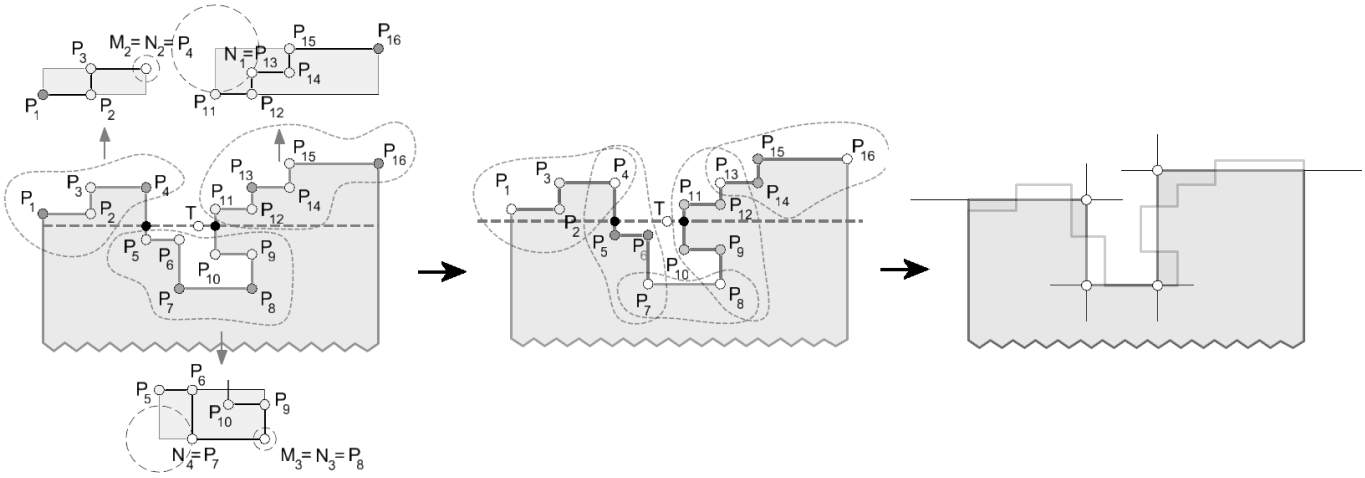

Figure 6: a) Splitting of the simplified edge e, $g^{(1)}(e) = 1$, into 3 new temporary edges, detection of the vertices N_j. b) Splitting of 3 temporary edges into 5 new edges, points between each detected adjacent vertices are assigned to new edge. c) The replacement of new edges satisfying the condition $\sigma < \sigma_{max}$ by the regression lines.

It is apparent, that a location of the temporary edge to the regression line depends on the location of its first point $P_{(1)} = [x_{(1)}, y_{(1)}]$. The temporary edge t having an orientation $g^{(1)}(N)$ on the left of the split edge e, if

$$g^{(1)}(N) = \begin{cases} 1, & y_{(1)} > y_T, \\ 2, & x_{(1)} > x_T, \\ 3, & y_{(1)} < y_T, \\ 4, & x_{(1)} < x_T. \end{cases}$$

The next temporary edge will be in the opposite position to the regression line than the previous one temporary edge. So positions of temporary edges to the regression line are alternately changing.

Results of the splitting procedure. It should be noted, that points of the building closest to vertices of the smallest area rectangle, represent possible vertices of such building. The first/last temporary edge will be replaced by a *pair* of new perpendicular edges, the second/penultimate new edge will be shared with the next/previous temporary edge. Other temporary edges will be replaced by a set of *three* new perpendicular edges. The first new edge will be shared with the previous temporary edge, the third new edge will be shared with the next temporary edge, see Figure 6 a). It is noticeable, that the first and last temporary edges will be split in another way than other temporary edges. Another important variable represents an *index* of the temporary edge.

New vertices of the building. On the basis of an index of the temporary edge and a position of the temporary edge to the split edge, corresponding points M_j and N_j (vertices) are searched. There are several rules $R_1 - R_6$ shown in Figure 7:

1. *first temporary edge t on the left of the regression line:* Search for point N_2 closest to point M_2. Point N_2 becomes a new vertex of the building.
2. *first temporary edge t on the right of the regression line*: Search for point N_3 closest to point M_3. Point N_3 becomes a new vertex of the building.
3. *last temporary edge t on the left of the regression line:* Search for point N_1 closest to point M_1. Point N_1 becomes a new vertex of the building.
4. *last temporary edge t on the right of the regression line:* Search point N_4 closest to point N_4. Point N_4 becomes a new vertex of the building.
5. *other temporary edge t on the left of the regression line:* Search two points N_1, N_2 closest to points M_1, M_2. Points N_1, N_2 become new vertices of the building.
6. *other temporary edge t on the right of the regression line:* Search for two points N_3, N_4 closest to points M_3, M_4. Points N_3, N_4 become new vertices of the building.

The splitting process for three temporary edges t_1, t_2, t_3 is shown in Figure 6 b). In some cases the geometric form of the edge may cause problems. As an example the following situation, where one point P_i is the closest point to more than one vertex of the smallest area enclosing rectangle, can be mentioned.

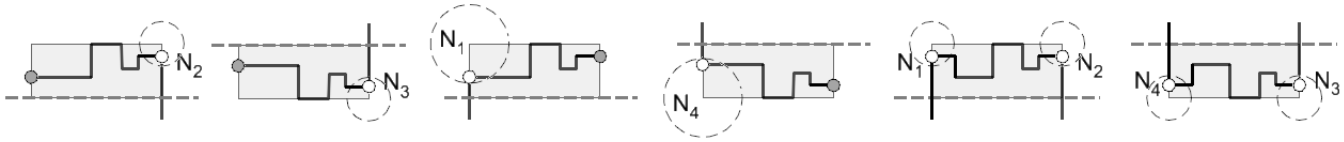

Figure 7: The detection of new vertices of the building using the smallest area enclosing rectangle over temporary edges t_k, $g^{(1)}(t_k) = 1$.

Modified splitting rules. If some split edge e has orientation $g^{(1)}(N) \neq 1$, a set of the vertices $\{M_1, M_2, M_3, M_4\}$ is rearranged to another set in accordance with Table 1. Let us give an example of some edge with the orientation of $g^{(1)}(N) = 2$. The call of the splitting procedure with rearranged ordering of formal parameters $\{M_4, M_1, M_2, M_3\}$ results in the following assignment: $M_1 = M_4$, $M_2 = M_1$, $M_3 = M_2$, $M_4 = M_3$. All new vertices $\{N_4, N_1, N_2, N_3\}$ will be rearranged the same way: $N_1 = N_4$, $N_2 = N_1$, $N_3 = N_2$, $N_4 = N_3$. For further information about splitting rules see Table 4.

Table 4: Modified splitting rules: an rearrangement of the set of new vertices depending on $g^{(1)}(N)$.

$g^{(1)}(e)$	R_1	R_2	R_3	R_4	R_5	R_6
1	N_2	N_3	N_1	N_4	N_1, N_2	N_3, N_4
2	N_1	N_2	N_4	N_3	N_4, N_1	N_2, N_3
3	N_4	N_1	N_3	N_2	N_3, N_4	N_1, N_2
4	N_3	N_4	N_2	N_1	N_2, N_3	N_4, N_1

An orientation of new vertices. All found vertices are given an orientation $g^{(r)}(N)$ depending on the position of the temporary edge to the regression line and on an index of the temporary edge. It ensures, that newly created adjacent edges will be perpendicular to each other. Otherwise, it would not be guaranteed, that adjacent edges have any intersection. Look to the following rules:

1. For the first point of the temporary edge $P_{(1)}$ we set $g^{(r)}(P_{(1)}) = g^{(r-1)}(P_{(1)}) + 1$, for the point N_2 we set $g^{(r)}(N_2) = g^{(r-1)}(P_{(1)}) + 2$.
2. For the first point of the temporary edge $P_{(1)}$ we set $g^{(r)}(P_{(1)}) = g^{(r-1)}(P_{(1)}) + 1$, for the point N_3 we set $g^{(r)}(N_3) = g^{(r-1)}(P_1) + 4$.

3. For the point N_1 we set $g^{(r)}(N_1) = g^{(r)}(P_{(1)})$.

4. For the point N_4 we set $g^{(r)}(N_4) = g^{(r)}(P_{(1)})$.

5. For the point N_1 we set $g^{(r)}(N_1) = g^{(r)}(P_{(1)})$, for the point N_2 we set $g^{(r)}(N_2) = g^{(r-1)}(P_{(1)}) + 2$.

6. For the point N_4 we set $g^{(r)}(N_4) = g^{(r)}(P_{(1)})$, for the point N_3 we set $g^{(r)}(N_3) = g^{(r-1)}(P_{(1)}) + 4$.

For edges with orientation $g^{(1)}(N) \neq 1$ all vertices (in previous relations) will be rearranged the same way, see Table 4.

An orientation of new edges. Each detected vertex of the temporary edge becomes a start point of the new edge and an end point of the previous edge. An edge with detected temporary edges is split into a set of new edges. The each new edge contains all points P_i lying between adjacent vertices N_j and N_{j+1}. We set orientation $g^{(r)}(N)$ of the first vertex of new (i.e. split) edge for all points of this edge (i.e. we set an orientation $g^{(r)}(N)$ for this edge). During this step a set of new edges is formed, all edges are subsequently added to the stack. Each edge can be split in the next recursion step equally, if necessary.

Figure 6 c) shows the results of the simplification process for three temporary edges. Split edges satisfying the condition $\sigma < \sigma_{max}$ are replaced by the regression lines. Orientations of newly created edges from the last example are shown in Table 5.

Table 5: The orientation of five new edges e_1, e_2, e_3, e_4, e_5 for $r = 2$.

e_1	P_1	P_2	P_3	P_4		
$g^{(2)}(P_i)$	2	2	2	2		
e_2	P_4	P_5	P_6	P_7		
$g^{(2)}(P_i)$	3	3	3	3		
e_3	P_7	P_8				
$g^{(2)}(P_i)$	2	2				
e_4	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}
$g^{(2)}(P_i)$	6	6	6	6	6	6
e_5	P_{13}	P_{14}	P_{15}	P_{16}		
$g^{(2)}(P_i)$	2	2	2	2		

Data structures. For the implementation of the algorithm sequential containers (list L) and container adapters (stack S) were used. All points are stored along with information about their coordinates x, y and orientation $g^{(1)}$ in the list. All edges of the building are stored with information about points forming the edge in the stack. New edges are pushed into the stack and popped in reversed order. As a result of the use of the list, after the completion of all recursive procedures adjacent edges will be perpendicular to each other.

4.5 Building reconstruction

The reconstruction of the simplified building is carried out from parameters of regression lines. Edges are stored in the output data structure represented by the list L, therefore two following edges popped from the list are always perpendicular. The reconstruction process runs in a very simple way:

1. Initialize $i = 1$. Get edge e from the top of the list L.

2. Store coordinates x_T, y_T of the edge e as $x_T^{old} = x_T$, $y_T^{old} = y$.

3. Using (2) determine its orientation $g^{(1)}$. Let B_i is a new edge point of the simplified building. If $i > 1$

$$g^{(1)}(N) = \begin{cases} 1,3: & B_i = [x_T^{old}, y_T]. \\ 2,4: & B_i = [x_T, y_T^{old}]. \end{cases}$$

4. Add B_i to the building.

5. Go to (1) until we reach end of the list L.

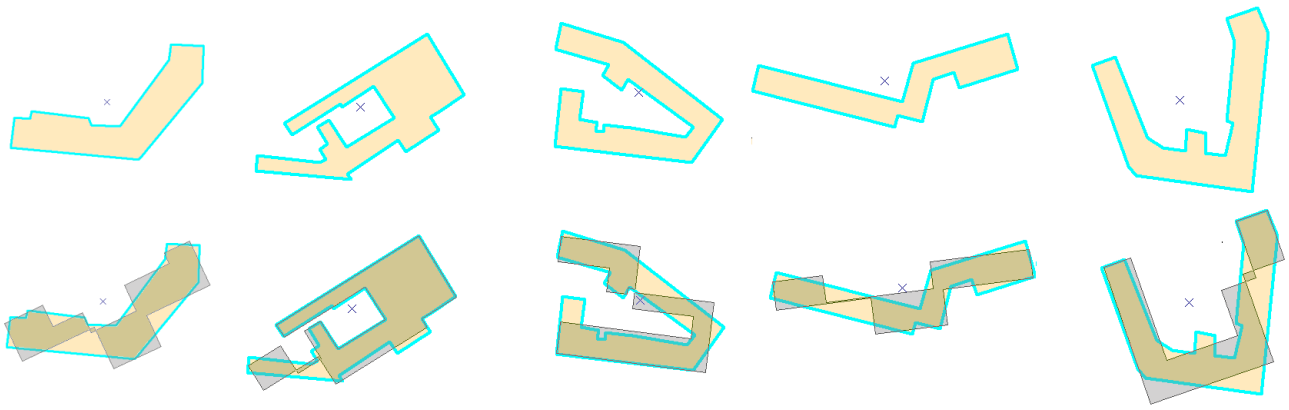

Figure 8: The problem of self intersections for non-rectangular, narrow and long buildings after the splitting procedure, visualization in ArcGIS 9.3.

4.6 Rotation of the building to the starting position

The reconstructed building is rotated by the angle of φ to the starting position (clockwise rotation about the origin). Then

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}. \tag{11}$$

4.7 The detection of self intersections

During the process of cartographic generalization we can be encountered with the problem of self intersections. They represent such situations, in which some undesirable geometric forms (long, narrow and non-rectangular buildings) as results of the generalization process have been created. Due to the topological incorrectness of such data, this error is very dangerous. Closed "pseudo region" 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 pseudo region will be considered as topologically incorrect, see Figure 8.

One of the possible solutions may be a test, which verifies an existence of self intersections. Before an edge removing or edge splitting procedure it is verified, whether actually simplified 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. The fast search (in sub-quadratic time) of possible line segments intersections is a fundamental step of the improved simplification algorithm. One possible solution brings Plane Sweep Algorithm well known as Bentley & Ottman algorithm (Berg et al., 2000, p. 26). Currently a version using Bentley & Ottman algorithm working with the balanced binary tree is being in development. But it shows the need for modification of the concept of the simplification algorithm, first of all some splitting rules.

5 Results and implementation

The algorithm was implemented in C++, the graphical user interface was created using WinAPI. With the use of ArcObjects libraries, a new DLL application designed for ArcGIS 9.x was created. This tool is applicable to the polygonal shape files. Author does not use any publicly available library of geometric algorithms.

Table 6: Running time (in sec) for the building simplification algorithm depending on the amount of buildings.

Buildings	100	200	500	1000	2000	5000	10 000	20 000	50 000
Running time [sec]	0.8	1,1	2.1	3.6	6,8	17,2	34,3	65,3	163,1

Performance test. Testing was performed on the DLL version of the presented algorithm using a 2.2 GHz CPU, 2 GB RAM, OS Windows Vista and ArcGIS 9.3. Sample data set consists of 51 757 buildings with 357 516 vertices (approx. 7 vertices per building). Table 4 shows the running time depending on the amount of buildings (including redrawing all buildings and

saving into feature class). Preliminary results for each set of buildings (100-50 000) represent the average of the ten runs of the algorithm under the data set. The performance is significantly affected by the overall slowness and hardware requirements of the ArcGIS (written in .NET). Nevertheless, the algorithm could be also used for the larger data sets.

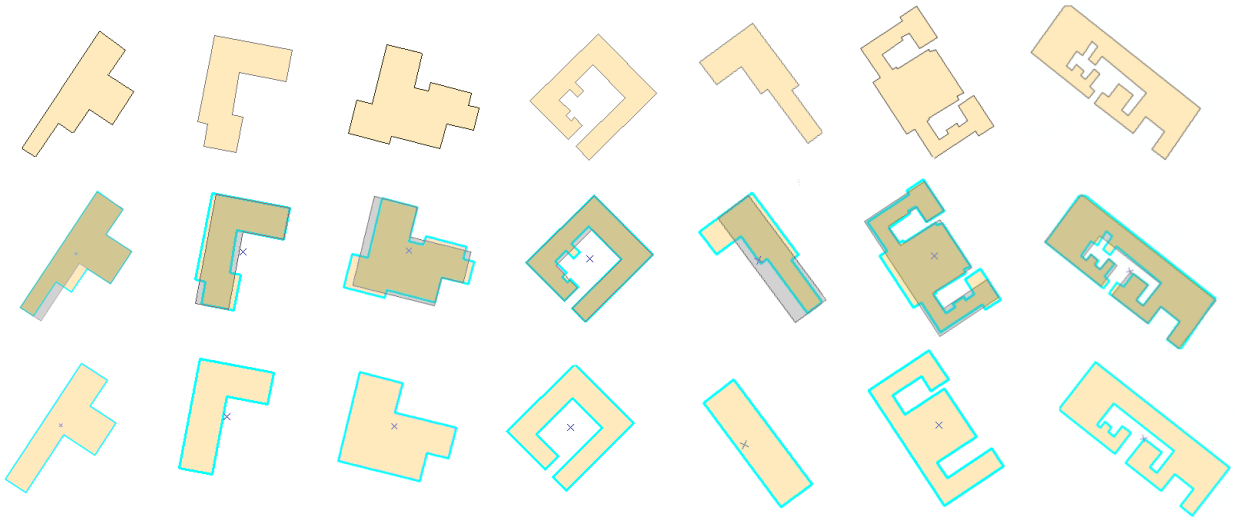

Figure 9: Results of the simplification process for various geometric forms of rectangular buildings, visualization in ArcGIS 9.3.

Cartographic assessment. The presented algorithm gives cartographic results. It was able to simplify buildings with more complex geometry, sample data contained artificial polygons with hundreds of randomly generated vertices. In most cases, such buildings represent only geometric constructs. However, they generally verify abilities of the simplification algorithm. It also appeared, that the need for manual corrections of the simplification process is relatively rare (for rectangular buildings).

This algorithm is suitable for stand-alone rectangular buildings. They represent buildings of "classical" geometric forms, see Figure 9. Modern architecture is typical of the greater form variability and more complex structures. Due to the fact, that original segments of the buildings are replaced by perpendicular segments, results may look artificially.

The accuracy of determining the angle of rotation φ affects the resulting quality of the simplification algorithm. For U-shaped and L-shaped buildings better results are achieved, L-shaped and Z-shaped buildings bring more problems. For these buildings the smallest area enclosing rectangle may not represent true rotation of the building. As a result of the recursive splitting new vertices in such positions, which are absent in the original building, can be created, see Figure 10. And so the resulting polygon meets geometric conditions but it is not similar to the building and it has no cartographic relevance (nor aesthetic pleasing). One possible solution of this problem represents the implementation of more reliable methodology for detecting the angle of rotation based on statistical weighting by Bader (2000) or wall average by Duchene et al. (2003) or its combination.

The process of cartographic generalization of the buildings depending on the value of σ_{max} can be considered as quite smooth and natural. Only for buildings, where one of the edges has more complex geometric form than others, the simplification process could be made "in jump". Such edge may be replaced by the regression line during one recursion step, which looks somewhat artificially. Possible solutions may constitute a modification of the splitting criterion or splitting rule.

The presented algorithm nearly keeps an area of the building and therefore it can be categorized in terms of the mathematical cartography as "equal area algorithm" (area distortion less than 5%).

6 Conclusion

This paper introduced an algorithm for automated cartographic generalization of buildings based on modified Douglas-Peucker algorithm using the least squares adjustment, that is a redesign of the solution published by Sester (2000), Sester (2004), Dutter (2008). The process of the simplification was significantly modified due to a request for the recursive solution of the problem. As mentioned above, this algorithm is suitable for stand-alone rectangular buildings. Non-rectangular shapes or buildings blocks (in historic urban centers) may cause problems. The proposed algorithm has the ability to simplify buildings of any geometric form, while the algorithm presented by Dutter (2008), some buildings with more complex geometry leaves unprocessed.

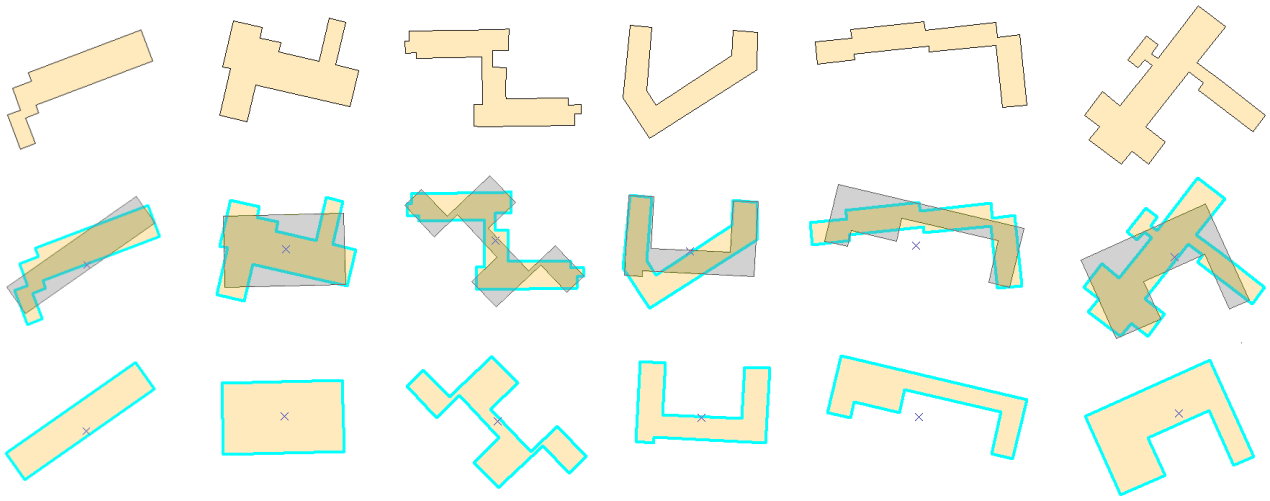

Figure 10: The problem of determining the angle of rotation φ for L-shaped and Z-shaped buildings and results of the simplification process, visualization in ArcGIS 9.3.

For future improvements in functionality, the author is going to modify the algorithm so that it will be suitable for common buildings, prevent self intersections and provide better cartographic results. The aim is to achieve more natural geometric form of simplified buildings and reduce the number of cases, where the algorithm gives cartographic wrong or unaesthetic results. Another possible improvement represents automatic removing of short offsets and extrusions (which may occur as an inappropriate consequence of the automatic simplification algorithm). An interesting opportunity for further research brings adding the dependency of σ on the map scale.

In general, this algorithm provides relatively good cartographic results and minimizes (for rectangular buildings) the needs of manual corrections. The presented algorithm was implemented as a stand-alone tool for ArcGIS 9.x and DLL library could be sent freely upon a request.

References

[1] Berg M, Kreveld M, Overmars M, Schwarzkopf O (2000) *Computational Geometry*, Springer

[2] Douglas D, Peucker T (1973) *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, The Canadian Cartographer 10(2), pp 112-122

[3] Dutter M. (2008) *Generalization of buildings derived from high resolution remote sensing data*, Wien

[4] Freeman H, Shapira R (1975) *Determining the minimum-area encasing rectangle for an arbitrary closed curve*. Communications of the ACM 18 (7), pp 409-413.

[5] Galanda M (2003) *Automated Polygon Generalization in a Multi Agent System*, Mathematisch-naturwissenschaftlichen Fakultät, Universität Zürich

[6] Haunert J H & Wolf A (2008) *Optimal Simplification of Building Ground Plans*, ISPRS Congress Beijing

[7] Rourke O J (2005) *Computational Geometry in C*, Cambridge University Press, 2005

[8] Sester M *(2000) Generalization based on least square adjustment*, International Archieves of Photogrammetry and Remote Sensing

[9] Sester M, Brenner C (2004) *Continuous Generalization for Visualization on Small Mobile Devices*, in: Peter Fisher (Edt.): Developments in Spatial Data Handling - 11th International Symposium on Spatial Data Handling, Springer Verlag, pp 469-480

[10] Sester M, Brenner C (2009) *A vocabulary for a multiscale process description for fast transmission and continuous visualization of spatial data*, Computers and Geosciences, 2009.

[11] Toussand G (1983) *Solving Geometric Problems with the Rotating Calipers*, McGill University Montreal