Merging a set of polygons with non-stable borders

Gregor Klajnsek
gregor.klajnsek@siol.net

Laboratory for Geometric Modeling and Multimedia Algorithms
Institute of Computer Science
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor / Slovenia

Abstract

The paper introduces an algorithm dealing with polygons with non-stable borders. Such polygons may appear because of measuring errors or during scanning and vectorisation of blueprints. The aim of the algorithm is to remove these errors within prescribed tolerance. The algorithm uses a sweep-line approach and works in two steps. In the first step, all non-stable areas are determined and in the second step fixed.

Keywords: algorithms, computational geometry, polygons, non-stable areas.

1. Introduction

Polygons present basic geometric structures in many 2D applications. Perhaps the most important are geographic information systems (GIS) where polygons represents piece of land considered as parcels. It is reasonable to expect that the parcels fit completely together because in reality, there is no overlapping or empty spaces between them. However, the borders of these parcels are obtained by measuring with given tolerance. In reality this tolerance is +/- 12 cm. Because of this, it could happen, that in GIS database the borders of the parcels do not fit exactly together, especially if different logical units are tried to be unified. The problem is even more difficult, if they have been stored in different databases. In such cases, the borders of the parcels do not fit together entirely but instead, they overlap or there are empty spaces among them. Such data represents real problems in further operations on geometric data and therefore, they have to be corrected. Indeed, the corrected polygons do not express the exact real data, but they are consistent within the given tolerance.

In this paper, an algorithm for detecting and eliminating described problem is considered. The algorithm works in two steps: At first, by the use of a sweep-line, the borders of the polygons that are not consistent are determined, and in the second step, they are corrected. Although practically important, we have found no solution to presented problem in the literature.

2. Description of the problem

In Figure 1 an example set of polygons is shown. Some of them are consistent regarding their neighbours, while others are not. At first, the polygons, which are consistent, are merged together and a smaller set of non-consistent polygons is obtained. Let us assume we already have a routine

performing merging. Here, just a brief idea is given. We have to eliminate all polygon edges, which are doubled in the database. With the proper organization of data structures, this task can be finished very fast. For example, Zalik reported that 100.000 polygons with 1.000.000 edges might be merged in 38 seconds measured on an ordinary PC [1]. The reduced set of polygons is then obtained and it is shown in Figure 1b. However, some vertices of those polygons are not consistent, this means, that they are not adjusted with the vertices in the neighbouring polygons. Therefore, the edges are also non-consistent and we considered such polygons as polygons with non-stable borders. Desired result of merging is shown in Figure 1c and can be obtained by the process of "stabilization" of the polygon borders. This process is described in the continuation. For clarity, the algorithm is explained when just two polygons are presented, although the generalization to more polygons is simple.

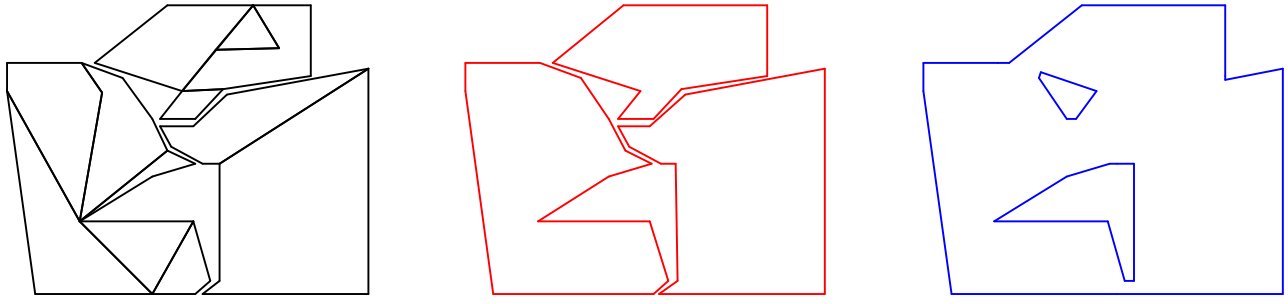


Figure 1: Input set of polygons (a) Result of merging (b) Expected result of merging (c)

Let us determine the possible cases of instability, which may appear.

- **Gap area.** If the borders of two polygons are partially far way for less than prescribed tolerance ε, but they do not intersect, a tiny gap appears between the polygons (see Figure2a). To get correct result gaps are filled by additional polygons that disappear during the merging process, which is rerun after stabilization is complete.
- **Intersection area**. Figure 2b shows an example of intersection area. Again, additional polygons are created at first, and then they are removed from the result.
- **Combined are.** This case includes gaps and intersection areas (Figure 2c). Again, additional polygons are created and classified as gap or intersection areas.

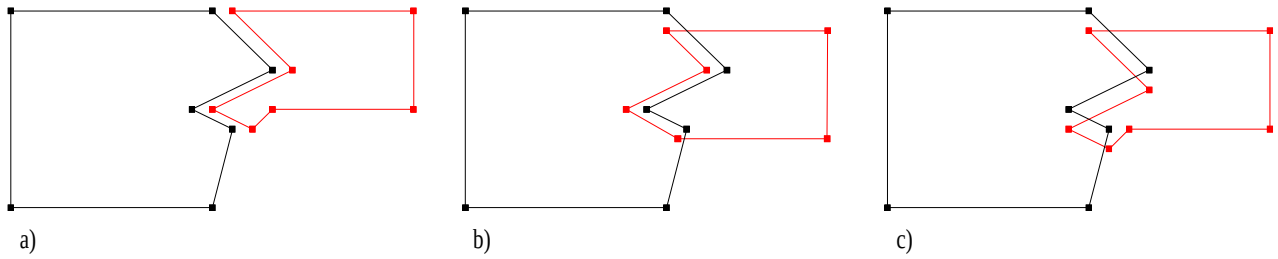


Figure 2: The cases of non-stable areas: gap (a) Intersection area (b) Combined area (c)

Obviously, the algorithm needs to know how wide the non-stable area could be. In reality, this is connected with the measurement error. Here, we will denote it by a parameter called the ε-distance,

or shortly ε. Each polygon, whose all edges are further apart than ε, is not considered in this algorithm. Indeed, such polygons must be eliminated from the process of non-stable area determination as soon as possible.

3. Algorithm

As mentioned, the algorithm for fixing the non-stable areas works in two steps:

- Determination of non-stable areas
- Removal of non-stable areas.

These steps are described in the continuation.

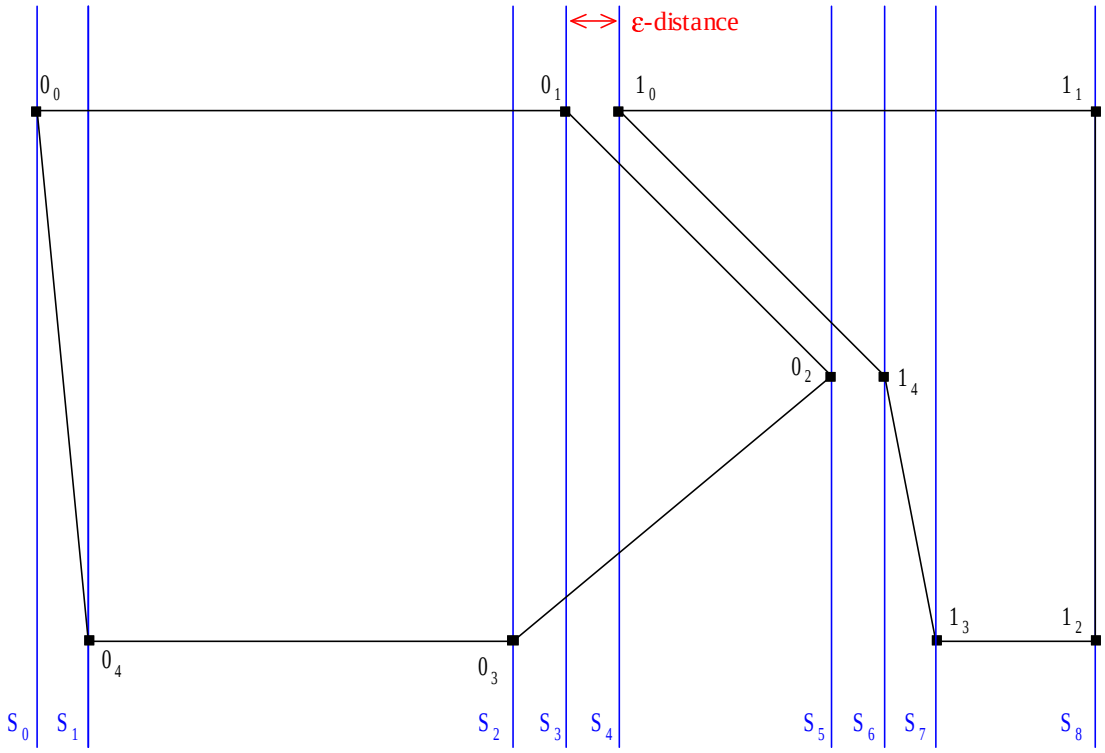
3.1 Determination of non-stable areas

A sweep-line algorithm has been used for detecting non-stable areas. The sweep-line algorithm introduces a sweep-line moving from $-\infty$ to ∞ across the plane [2]. Let us suppose it is vertical. It stops only at the event points - polygon vertices in our case. At the stop, the algorithm must update so-called sweep-line status (SLS) and perform some operations, if necessary. SLS contains all polygon edges, which are intersected by the current sweep-line. These edges are usually named active edges [3]. Because, we have to handle non-stable edges, each edge being inserted into SLS is prolonged for ε in x direction. When the sweep line hits a new event point the following actions are executed:

- For each edge in SLS, at first the horizontal distances between its points and the event point are determined. If both distances are greater ε, the edge is removed from SLS.
- The shortest Euclid distances between the event point and all remaining edges in SLS are calculated. If any distance is less then ε, the non-stable area is found.
- The edges which are connected with the actual event point and are not yet in SLS are inserted into it (i.e. the second vertex of these edges has x coordinate larger than x coordinate of the event point).

To ensure that the sweep-line algorithm works correctly, all polygon vertices have to be sorted regarding their x coordinate.

Let us highlight the algorithm by an example shown in Figure 3. For this purpose, let us observe what happened in individual event points.



line status.

Sweep line S_1: Sweep line stops at vertex 0_4. Horizontal distance between the right end of all edges in SLS and vertex 0_4 is calculated. As no horizontal distance is greater then ε all edges remain in SLS. As edges 0_00_4,0_00_1 and vertex 0_4 belong to the same polygon, calculation of Euclid distances between the vertex and the edges is skipped. Edge 0_30_4 is added into SLS.

Sweep line S_2: Horizontal distance between vertex 0_3 and the right end of edge 0_00_4 is greater then ε-distance. Therefore, edge 0_00_4 is erased from SLS. Other edges in SLS have smaller horizontal distances then ε and remain in SLS. All edges still belong to the same polygon and no other calculations have to be done. Edge 0_20_3 is added into SLS, which now consists of edges 0_00_1, 0_30_4, and 0_20_3.

Sweep line S_3: Sweep line stops in vertex 0_1. After calculation of horizontal distances, edge 0_40_3 is removed from SLS, after that edge 0_10_2 is added to SLS.

Sweep line S_4: Sweep line hits point 1_0. After calculation of horizontal distances, between edges in SLS and vertex 1_0, edge 0_30_4 is removed from SLS. Edges in SLS belong to polygon P_0, while vertex 1_0 belongs to polygon P_1. Because of this, we have to calculate Euclid distances between the vertex and the edges. Distances from edges 0_00_1 and 0_20_3 to vertex 1_0 are greater than ε and therefore, these edges are ignored. But, the distance between the edge 0_10_2 and the vertex 1_0 is smaller then ε indicating that the non-stable area has been entered. For the next step of the algorithm available information about non-stable area is stored. Up to now we know, that edge 0_10_2 and vertex 1_0 belong to this area. After that, edges 1_01_1 and 1_01_4, which derive from vertex 1_0 are added into SLS.

Sweep line S_5: Edge $0_0 0_1$ has horizontal distance to vertex 0_2 greater then ε and therefore is removed from the SLS. As vertex belongs to polygon P_0, Euclid distances between vertex 0_2 and edge $1_0 1_1$ and $1_0 1_4$ are determined. The distance between vertex 0_2 and edge $1_0 1_1$ is greater than ε, and therefore edge $1_0 1_1$ does not fall into non-stable area. On the other hand, because the distance between the vertex 0_2 and edge $1_0 1_4$ is smaller then ε it falls into non-stable area. As vertex 0_2 is already in the data structure describing non-stable areas, only additional information just found is added (edge $1_0 1_4$).

Sweep line S_6: Edges $0_1 0_2$ and $0_2 0_3$ are removed from SLS, as their horizontal distance to vertex 1_4 is greater then ε. All remaining edges in SLS and vertex 1_4 belong to polygon P_1 so determination of Euclid distances is skipped. Edge $1_3 1_4$ is added into SLS, which now consists of edges $1_0 1_1$,$1_0 1_4$, and $1_3 1_4$.

Sweep line S_7: Edge $1_0 1_4$ is deleted from SLS. Determination of Euclid distances is skipped and edge $1_2 1_3$ is added to SLS.

Sweep line S_8: Sweep line hits vertices 1_1 and 1_2. After horizontal distance is calculated all edges remain in SLS. Edge $1_1 1_2$ is added into SLS.

For described example, the sweep line algorithm produced following sets of data describing non-stable areas:

List of non-stable polygons: $\{P_0,P_1\}$

List of non-stable borders: $\{0_1 0_2, 1_0 1_4\}$

List of non-stable vertices: $\{0_1,0_2,1_0,1_4\}$

3.2 Removal of non-stable areas

In this step, the algorithm constructs the additional polygons obtained from the information about the non-stable areas. As we already know, three possible cases exist: gaps, intersections and combined non-stable areas. How they are constructed is described in the continuation.

3.2.1 Fixing gap areas

A polygon describing a gap consists of all the edges surrounding the gap and two additional edges, which must be determined to close the polygon. The creation of the new polygon follows the next algorithm:

- The set of non-stable-borders is split into two sets. In one set are all edges, which belong to polygon P_0 and in the second are edges belonging to polygon P_1.
- The left and the right chain have to be determined. After that, the end edges of the non-stable area are determined.
- Four perpendicular lines regarding to the end edges are constructed. From these, two, which actually intersect the edges on the opposite chain, are chosen. An intersection point between the perpendicular line and the corresponding edge is calculated. The intersected edge is then split and a new edge connecting the intersection point with the end vertex on the corresponding chain is generated.
- New polygon is constructed in this way and it fills the gap.

Let us consider the example in Figure 4a:

From the data structure filled in the first part of the algorithm, the sets of edges are obtained. Using the information from the input polygons, two chains are constructed. The left chain consists

of edges $0_1 0_2$, $0_2 0_3$, and $0_3 0_4$, and the right chain of edges $1_0 1_6$, $1_6 1_5$, and $1_5 1_4$. Let us consider now just the "upper" pair of edges of both chains – edges $0_1 0_2$ and $1_0 1_6$. From the ending points of both chains (in our case from the vertices 0_1 and 1_0), two perpendicular lines are calculated. Let us denote them as r_{left} and r_{right}. These lines are shown in Figure 4b. It happened, that r_{right} intersects left edge $0_1 0_2$. At the intersection point, edge $0_1 0_2$ is split and a new vertex V_0 is obtained. This vertex is then connected with the ending vertex of the right chain (vertex 0_1). Similarly, the algorithm solves the "bottom" part of the chains.

In Figure 4c the result of the merging operation, polygon P_m, after addition of the new polygon is shown.

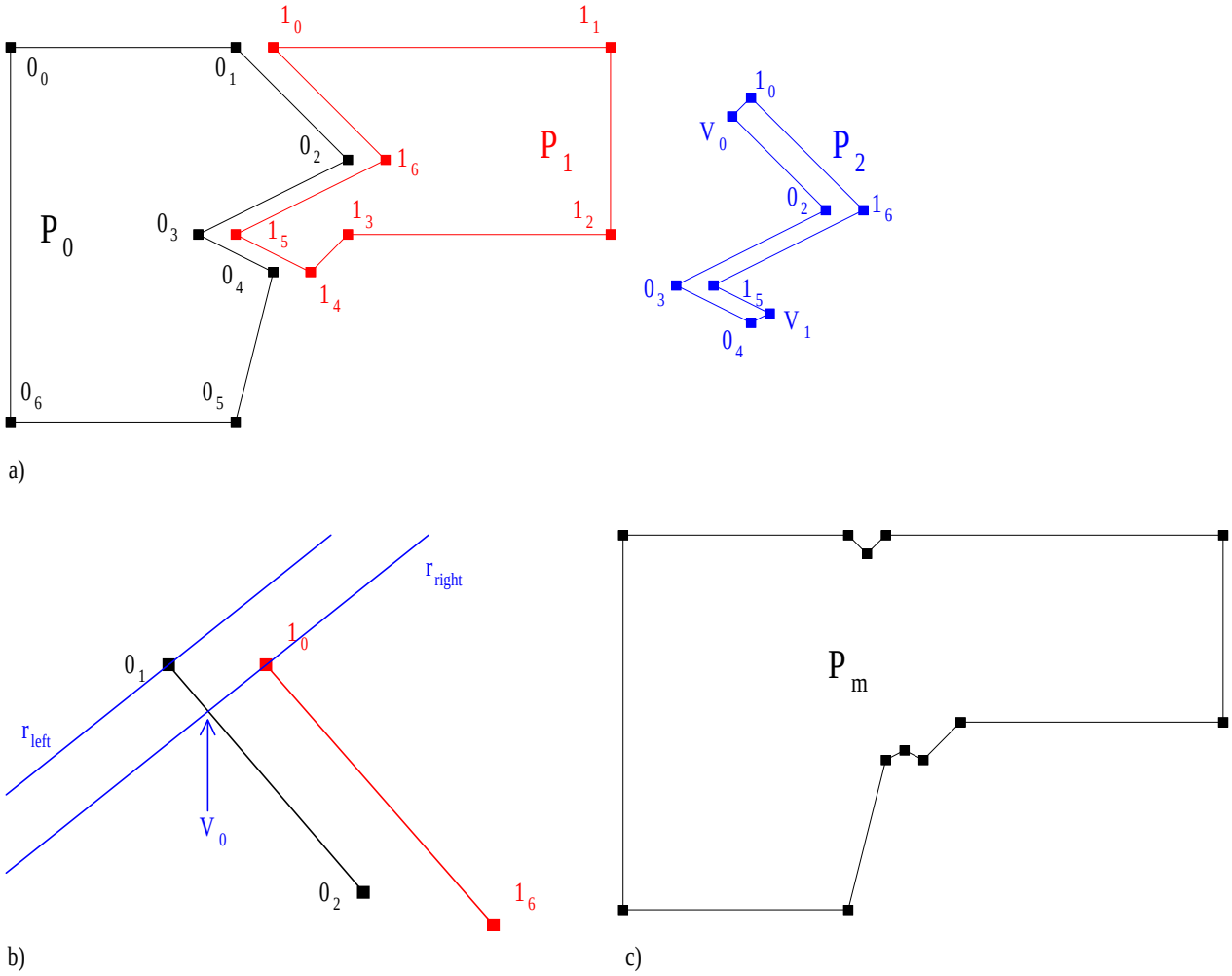


Figure 4: Set of polygons forming the gap area (a) Determination of additional edge (b) Final result of merging, after the "stabilization" (c)

3.2.2 Fixing polygon intersections

Two polygons, which intersect are transformed into three polygons. One polygon fits the intersection area. The intersection area is subtracted from the original polygons to obtain other two polygons. The creation of these three polygons follows the next algorithm:

- The set of non-stable-borders is split into two sets. In one set all edges belonging to polygon P_0 are located and in the second are edges belonging to polygon P_1.

- The left and the right chain have to be determined. After that, end edges of the non-stable area are determined.
- Intersection points between the end edges are calculated. These intersection points split end edges. All parts of end edges, which are not in the intersection area, are taken from the chains.
- All edges, which have remained in both chains, form the polygon, which fits the intersection area.
- The chains belonging to P_0 and P_1 are swapped and in this way polygons P_0 and P_1 are modified.

Let us highlight the algorithm using an example in Figure 5a:
From the data structure filled in the first part of the algorithm, two chains are constructed One chain consists of edges 0_10_2, 0_20_3, 0_30_4, 0_40_5 and 0_50_6 and the second chain from edges 1_01_5, 1_51_4, 1_41_3, 1_31_2 and 1_21_1. From the end edges of both chains, two intersection points are determined. Vertex V_0, represents intersection point between edges 1_01_5 and 0_10_2 and vertex V_1 is the in the intersection point of edges 0_50_6 and 1_21_1. Edge 0_10_2, is split into two edges 0_1V_0 and V_00_2, and edge 1_51_0 is split into edges 1_5V_0 and V_01_0. As new edges 0_1V_0 and V_01_0 are outside the intersection area, they are deleted from the chains. Similarly, removal of the edges is performed at the "bottom" of intersection area. Now the right chain consists of edges V_00_2, 0_20_3, 0_30_4, 0_40_5, and 0_5V_1 and the left chain consists of edges V_01_5, 1_51_4, 1_41_3, 1_31_2 and 1_2V_1. These chains now form the polygon P_2. Chains are swapped to modify the polygons P_0 and P_1. Polygons P_0, P_1 and P_2 are shown in Figure 5b.

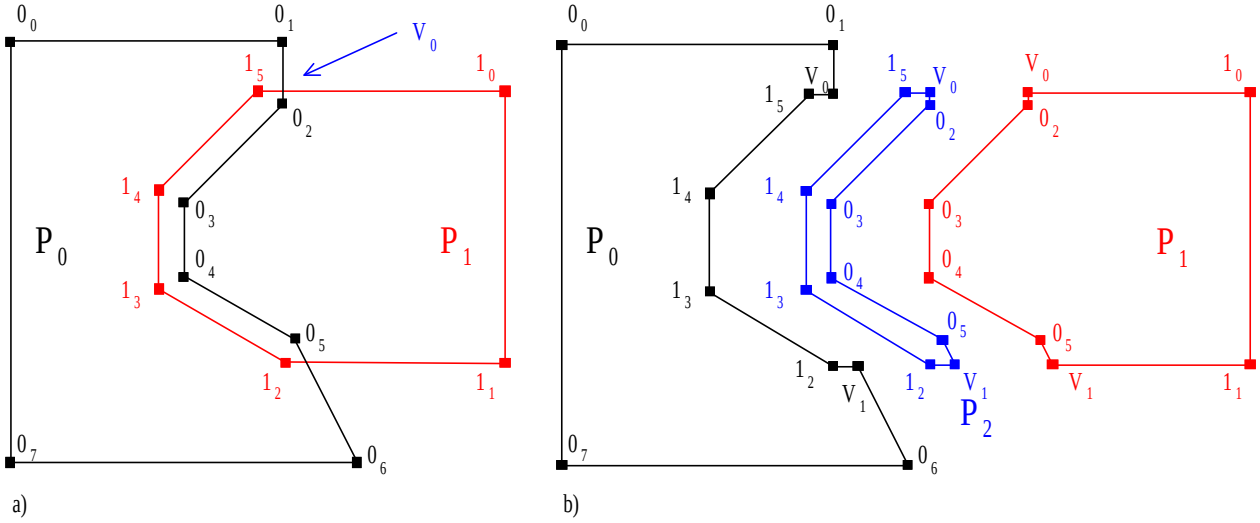


Figure 5: Set of intersecting polygons (a) Constructed set of consistent polygons (b)

3.2.3 Fixing combined areas

Combined areas consist of gap and intersection areas. All the gap areas must be covered with additional polygons, and the intersection areas have to be subtracted from the original polygons. New polygons fitting intersection areas must be constructed. The algorithm is described in continuation:

- The set of non-stable borders is split in two sets and the chains are determined.
- Two corresponding end edges are chosen as starting edges.
- If starting edges cross, the first area is intersection area, and the intersection point is determined. If end edges do not cross, first area is the gap area and additional edge is created.

- A walk-about is performed along the edges in chains, until the next intersecting edges are found or end of chains is reached. At each detected intersection point, the chains are split. If there are no intersecting edges, the algorithm jumps to the last step. Otherwise, the intersection point is determined, and the intersecting edges are split. New polygon is created from partial chains. If algorithm is fixing the intersection area partial chains are swapped in the original polygons, too. The partial chains are then removed. As gap area always follows the intersection area and vice-versa, the classification of next area is simple. The process is repeated until the algorithm deals with all edges.
- If the last two edges, do not intersect we have the gap at the "end" of intersection area. The additional edge is determined and a new polygon is formed from partial chains and the new edge.

Let us consider the example in Figure 6a:

From the data structure filled in the first part of the algorithm, two chains are constructed One chain consists of edges 0_10_2, 0_20_3, 0_30_4, 0_40_5 and 0_50_6, and the second chain contains edges 1_01_7, 1_71_6, 1_61_5 and 1_51_4. Edges 0_10_2 and 1_01_7 are chosen as starting edges. As these edges do not intersect, additional edge 0_1V_0 is formed.

Next intersecting edges are edges are edges 0_20_3 and 1_01_7. At their intersection, vertex V_1 is created, which splits both edges. "Upper" parts of edges are added to the partial chains. One partial chain now consists of edges 0_10_2 and 0_2V_1 and the second of edge V_0V_1. These partial chains and edge 0_1V_0 form the polygon P_2 which fills first gap area.

The next stop is intersection of edges 1_71_6 and 0_30_4. At their intersection new vertex V_2 is created which splits the edges. One partial chain consists now of edges V_10_3 and 0_3V_2 and the second from edges V_11_7 and 1_7V_2. The partial chains form polygon P_3, which covers the intersection area. In polygons P_0 and P_2 chains between vertices V_1 and V_2 are swapped and polygons are updated.

Polygon P_4 is formed from partial chains V_20_4, 0_4V_3 and V_21_6, 1_6V_3. As polygon P_4 fills the gap area, polygons P_0 and P_1 are not modified.

Polygon P_5 fits the last intersection area. It consist of chains V_31_5, 1_5V_4 and V_30_5, 0_5V_4. In polygons P_0 and P_1 chains between vertices V_3 and V_4 are swapped.

The final result of the algorithm are polygons P_0 (consisting of edges 0_00_1, 0_10_2, 0_2V_1, V_11_7, 1_7V_2, V_20_4, 0_4V_3, V_31_5, 1_5V_4, V_40_6, 0_60_7, and 0_70_0), P_1 (consisting of edges 1_01_1, 1_11_2, 1_21_3, 1_31_4, 1_4V_4, V_40_5, 0_5V_3, V_31_6, 1_6V_2, V_20_3, 0_3V_1, V_1V_0, and V_01_0), P_2, P_3, P_4, and P_5. All polygons are shown in Figure 6b.

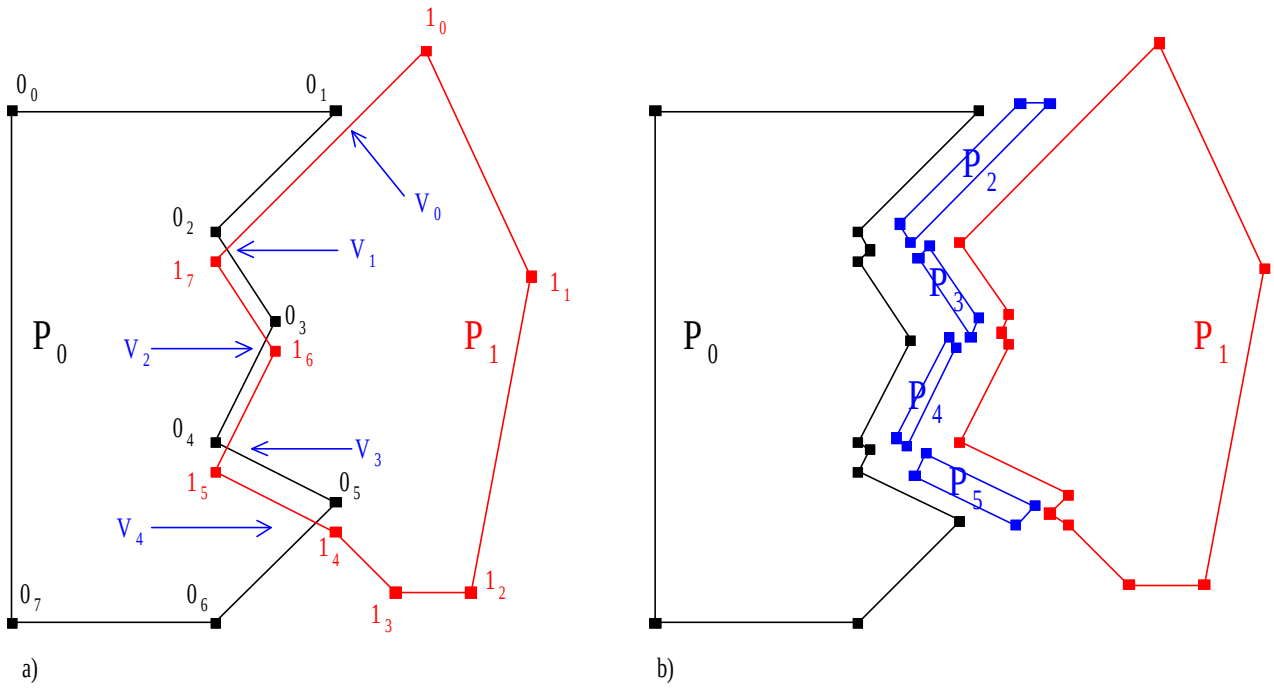


Figure 6: Set of polygons forming a combined area (a) Constructed set of consistent polygons (b)

4. Conclusions

The paper presents an algorithm for correcting the polygons with non-stable borders. As described, such po lygon easily appears at engineer practice, where polygons are obtained by some measurements subjected to measuring errors. We mention geographic information systems as an example. The algorithm presented here works in two steps: at first so-called non-stable areas are determined and in the second step, the additional artificial (blind) polygons are created. To accelerate the first step, a sweep-line algorithm is proposed. The second step handles three different cases how to construct these additional polygons.

The algorithm is being implemented at the time of writing. We hope that the implementation will confirm the correctness of the proposed algorithm.

5. References

[1] Zalik, B.: An Incremental Randomized Approach to the Envelope Determination of a Huge Set of Topologically Consistent Polygons, *paper sent to ACM ToG.*

[2] Preparata, F. P., Shamos, M. I.: Computational geometry - an introduction, Springer-Verlag, New York, 1985.

[3] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O.: Computational geometry – Algorithms and Applications, Springer, Berlin, 1997.