Delaunay-Based Adaptive Contour Reconstruction

Costin A. Boiangiu, and Mihai Zaharescu

***Abstract*—**The purpose of this paper is to offer a solution for building a controllable in refinement, adaptive contour, in a robust, intuitive manner, and generalized for any input point set. The presented algorithm is both mathematically elegant and general. Despite the fact that the problem and the solution are discussed for the two-dimensional case, the entire approach is valid for higher dimensions as well.

***Keywords*—**adaptive contour, alpha shape, beta shape, Delaunay triangulation

I. Introduction

THE original motivation for developing a tight fitting contour around a set of given data-points came when we needed to surround each element from a scanned newspaper page (paragraph, graphic, heading, etc.), so that they may be sent individually, in order, to an OCR system. The encapsulating contour must not contain fragments from neighboring elements, because the intersecting regions will be interpreted twice by the image to text conversion system (once for every element) and inserted throughout the text.

Newspapers have very nonstandard layouts, like random shaped images inserted between columns of text, with text flowing around them. This implies that a simple bounding box is not good enough; not even oriented bounding boxes or convex hulls are adequate, as the text contains concavities generated by inserted elements. Other methods that can be used are min-max shapes, alpha-shapes or conformal alpha shapes. Some of the drawbacks of each algorithm are mentioned here [1]. We will summarize them in Section II and explain why not even the more advanced ones are appropriate for the task.

Stepping away from the layout analysis domain, an algorithm for generating a tight fitting, connected, concave polygon, around a set of given data points is useful in domains like surface generation, collision detection optimization, mesh simplification and others that rely on computational geometry approaches. When a 3D scanner scans a single object, the sampled 3D vertexes must form a single connected geometry. The state of the art conformal alpha shapes fail to keep the geometry connected for all resolutions, while the others fail to recreate the finer details (alpha shapes) or concavities (convex hulls).

The paper is structured as follows: Section II discusses the drawbacks of similar methods, regarding scanned document segmentation; Section III describes the original proposed Beta-Shape algorithm; Section IV deals with the complexity of the original algorithm; Section V proposes a new way for generating the beta shape and; Section VI improves the output of the algorithm, generating a more naturally flowing mesh. At the end conclusions are drawn.

II. Drawbacks of Similar Methods

The history of contour and surface reconstruction, approximation is a long one. From simple, elegant, general mathematical concepts the theory evolved toward more complexes and more specifically application needs [2], [3].

Here we discuss some known methods for encapsulating objects, mainly from the point of view of how well suited are they for our document splitting problem [4].

The simplest method uses AABBs (Axis-Aligned Bounding Boxes). They are simple to calculate and intersections can be found easily, but that is about all they can do. For example, they can't even handle small rotations (which are very present in scanned documents because of imprecise paper alignment). Even small rotation angles cause neighboring columns to intersect on one of the axis. In our spitting problem they can be used as a first stage intersection detection.

The OBB (Oriented Bounding Box) [5] approach resolves the rotation problem but still assumes that all features are rectangular.

The CH (Convex Hull) [6] is the smallest convex set that contains all the data points. It can find more complex objects but magazine and newspaper articles usually include graphics that are inserted between text, and thus introducing concavities.

The Min-Max Shape [4] is formed by the intersection of two other shapes that can be calculated with little cost: the Min-Max onto X-axis shape and the Min-Max onto Y-axis shape. They are generated by taking the minimal and maximal point from every raster line, parallel to each axis. This can handle insertion of rectangular objects, but "C"-like shaped objects can't be handled by this approach.

The alpha-shape [7] is a filtration, or sub-graph, of the Delaunay triangulation and a generalization of the convex hull. Intuitively it is created by rolling a sphere around the input points; the contour left by the rolling ball is the alpha shape. The problem arises when the alpha parameter, the ball's radius, is small enough to fall between distant data points, effectively splitting the geometry. Increasing the radius is not a solution, as the ball won't be able to detect small cavities in the shape, thus lowering the shapes resolution.

Conformal alpha shapes [8] are an optimization of the alpha shape algorithm, aimed at changing the radius parameter adaptively with the local resolution of the mesh. This algorithm is thus able to generate a shape from non-uniform sampled datasets. However, the author mentioned that there are cases when the algorithm fails and generates holes in the constructed mesh. This happens because it selects a subset from the simplices generated by the Delaunay triangulation.

The largest spheres that fit in the empty spaces between data points are called *maximal open balls* and the union of their centers is the *medial axis*. The distance from a data point to the medial axis is the respective point's *feature size* (how dense the mesh is in that region). In regions where the sampling abruptly changes (ex: on a side of a data point the feature size is very small and on a different side it is very large) the sphere around the data point is too small to generate a triangle to the distant neighbor, thus generating a hole.

The mentioned algorithms are not specially built for page segmentation but rather for problems like intersection detection; pattern recognition; image processing; statistical analysis; global information system; reconstructing objects from a cloud of read data points, etc.

III. Beta Shape Algorithm

The Beta Shape algorithm was first proposed by Boiangiu [9] and it was developed just for this purpose: to generate a connected border, without holes, around all data points and as close as possible to the points, with the minimum number of edges, irrespective of the sampling.

Broadly described, the algorithm starts by computing the convex hull and then refines this border by iteratively adding data points to all segments larger than a given radius. At each iteration, for each border segment larger than the desired radius, the algorithm picks the closest point to the respective edge from a square region. This region is defined as the square, having an edge common to the border edge and oriented towards the center of the polygon. If an edge in the border is smaller than the radius, it will remain unchanged until the entire process stops and will finally be part of the resultant hull.

Using the square regions reduces the number of stolen best candidates from the neighboring regions and the search domain for the best point. However, when the border segments form sharp angles, they could possibly ask for the same point as best candidate. This means that an additional verification is needed in order to avoid stealing a neighbor's best candidate. This is also valuable for close-by edges that are inside each other's searching square.

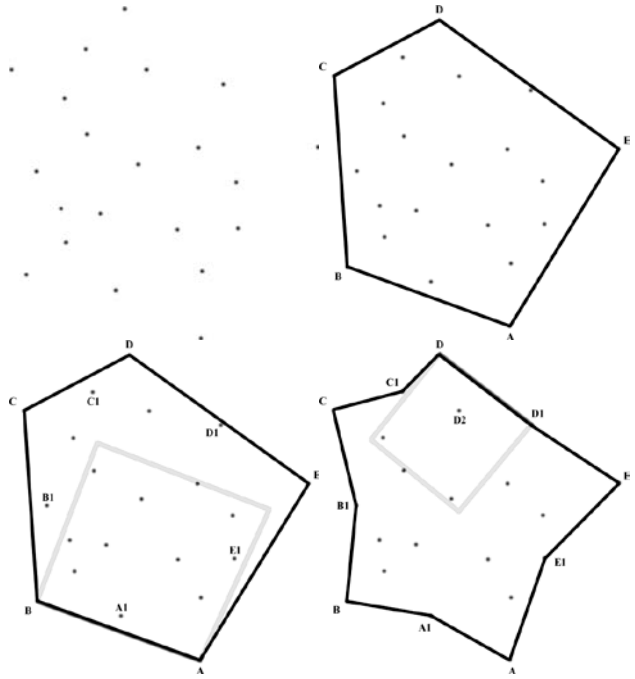

Fig. 1 Steps of the original Beta-Shape algorithm

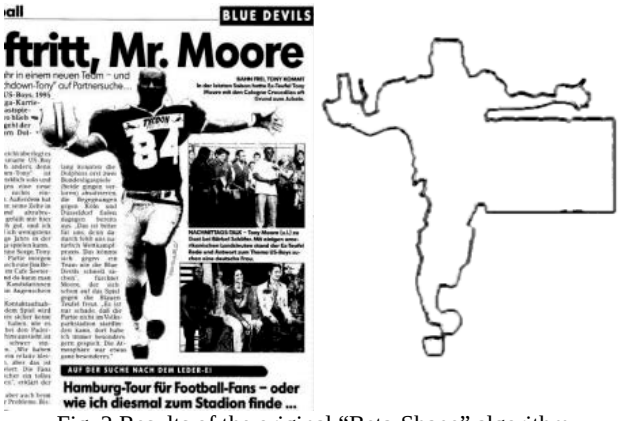

Fig. 2 Results of the original "Beta-Shape" algorithm

In order to verify if a point is inside the square, four inequalities are checked so that the following questions may be answered:

- is the point between the two perpendicular lines on the border segment?
- are they at the right depth?
- are they inside the shape?

It should be mentioned that a hash table can be used to fast-recover the search.

IV. Complexity

This chapter discuses the complexity of the original Beta-Shape algorithm starting from the author's mention [9] in the original paper that the overall complexity, deduced by

experimentation, is $O(n^{5/2})$ for random points and $O(n^{3/2})$ for real data.

In the case of random data this was determined using a Monte-Carlo approach and, as a result, observing an asymptotically decreasing convergence when dividing the execution time on the expected complexity for large batches of randomly-generated point sets. The results of the simulation are presented in Fig. 3.

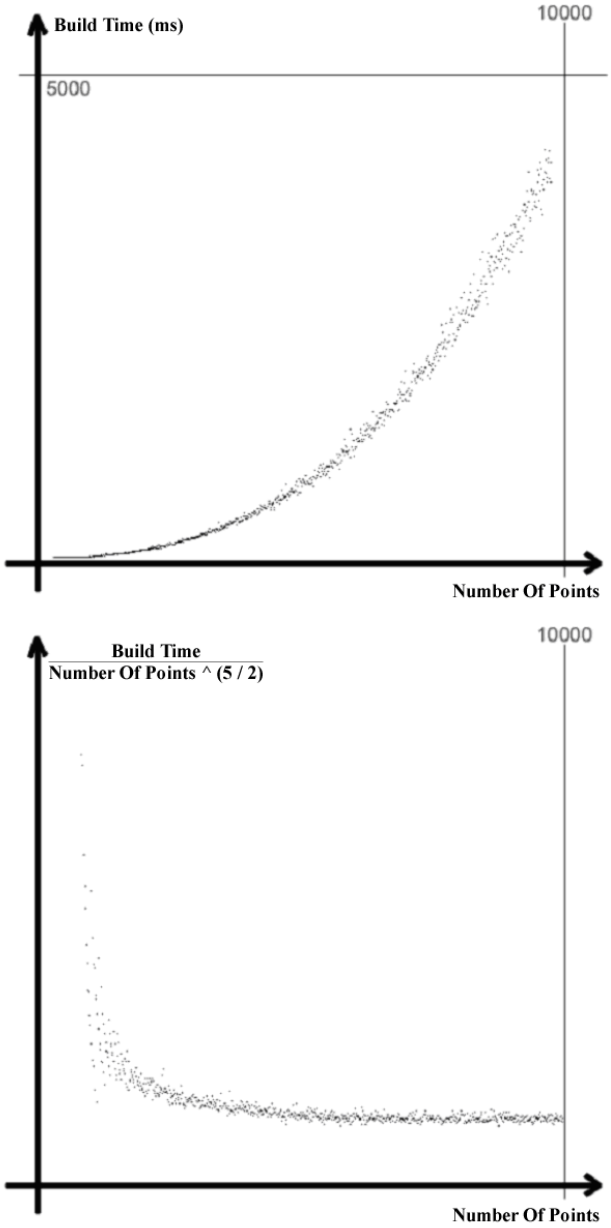

Fig. 3 The original "Beta-Shape" construction algorithm execution time vs. dataset input size for random point sets

To calculate the complexity, we sketch the program's steps:

1. Calculate the convex hull in $O(n \log h)$ time; where h is the number of points on the hull
2. Run $\log(n-h)$ iterations (because at each step, for each edge we include one more point in the border, doubling the number of edges and points on the border).
3. At each iteration, for every edge, find the best candidate. In order to find the best candidate, test if the point belongs to a square region, in $O(n-h)$ time and find the best candidate from the c points inside the square in $O(c)$ time.

Thus, the total complexity is about:

$$O(n \log h) + O((n-h)c \log(n-h)) \quad (1)$$

This explains the different complexity results for random and real data points: points in real binary images form clusters of white and black, giving a large h (number of points on the convex hull) for the complex hull. On the other hand, random numbers are spread, h being of $O(n^{1/3})$size, as calculated here [10].

Another attenuating factor for real images is that the edges are very small from the start. The data points are neighboring pixels - being the result of bitonal image conversion or wanted compact filtering of small white spaces - giving a small search region and small c.

V. Beta Shape Algorithm Improvement

In this chapter we propose a new method for obtaining the same results as the original Beta-Shape algorithm.

The proposed improved algorithm for calculating the beta-shape is still iterative. It erodes triangles one at a time from the Delaunay triangulation of the input points, verifying at each step the connectivity of the mesh [11].

The Delaunay triangulation generates the best connections that can be made between all data points, in order to minimize triangles that have sharp angles. The method avoids as much as possible the generation of skinny triangles. This is effectively achieved by testing that none of the points is inside the circumference formed by any other triangle.

Proof: if we have 4 points and a triangulation for them, the common edge of the two triangles formed is called *illegal,* if by flipping it we obtain a larger minimum angle. From Thales' theorem it can be noticed that if three of the points form a circle and the forth point is inside the circle, the value of the smallest corner will be smaller than that of the smallest corner had the point been outside.

By generalizing, in order to maximize the minimum angle from all the triangles, all the points must lie outside all circumscribed circles of all the triangles generated. The generalization is valid, because any four points can be validly triangulated just by flipping an edge and maintaining the outside border intact. Another explanation is that by flipping an illegal edge (the illegal edge that generates the smallest angle in the graph) the smallest angle will become greater. With every edge flip the smallest overall angle is increased and the algorithm finishes when no illegal edges remain.

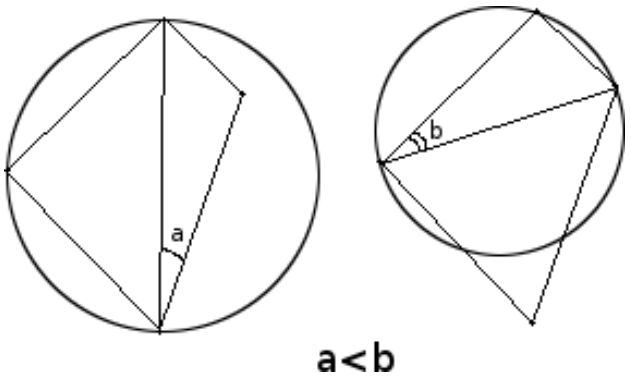

Fig. 4 Visual proof that an outside point generates a larger minimal angle

The Voronoi tessellation for a set of points consists of cells around each data point, such as if a random point lies inside one of the cells, it is closer to the cell's point than to any other data point. This means that the Voronoi vertexes (the point where at least three cells meet) are the centers of the circles formed by the three data points (or more if the points are co-circular) that form the neighboring cells. This generates just the condition for Delaunay triangulation, that all points are outside (further away) of the circle formed by any triangle.

The Voronoi graph and Delaunay triangulation are dual: the Delaunay triangulation can be calculated from the Voronoi graph by connecting all neighbors. The effect of this is that any edge will be connected to its closest points (because a Voronoi cell states that all points from the respective region are closest to the respective point).

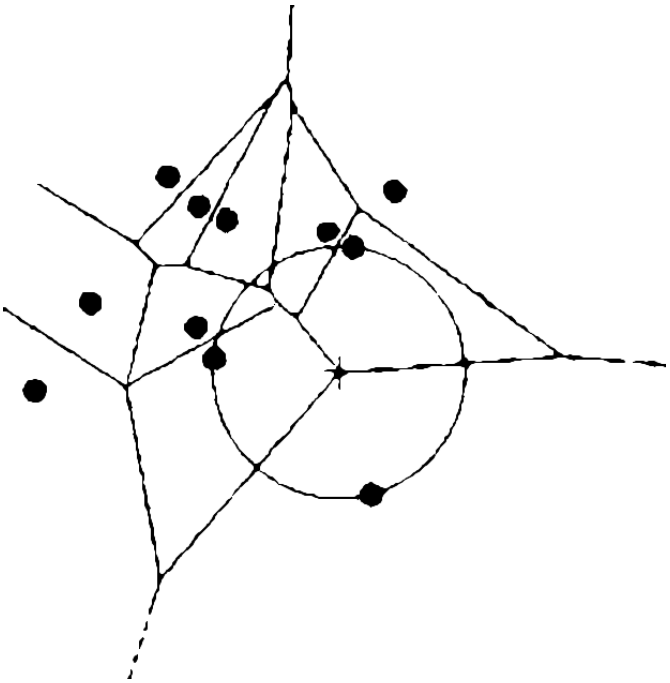

Fig. 5 Visual proof that a Voronoi vertex is the center of the circumscribed circle of neighboring data points

The conclusion is that if we know the Delaunay triangulation, we already know the best candidate for the outside edges.

Another intuitive proof is that if a different point, other than the one to which the border edge forms a triangle, would be closer to the edge that point would lay inside the Delaunay simplex, either isolated or causing edge intersections.

Knowing this, we generate an algorithm that selects the best candidates for the exterior edges in $O(1)$ time, and removes the exterior edges, appending to the hull the other two edges of the simplex triangle.

The unfiltered algorithm has flaws, as a single point can be best candidate for multiple edges, making the result dependent of the order of traversal. In order to eliminate this inconvenient we can remember the distance from a candidate to the edge and only erode the triangle with the minimal height.

The triangulation needs $O(n \log n)$ time, using a divide-et-impera method [12]. There are at most *n* points left to add to the border in $O(1)$ time, giving a total complexity of $O(n \log n) + O(n)$.

This way we obtain identical results to the originally proposed algorithm but with better complexity and more rigorous definition.

As the Delaunay triangulation is unique for non-co-circular points, the algorithm offers a single solution. In order to avoid multiple solutions for the case of co-circular data points, we can propose to select the top and left most point in case of a conflict.

VI. Natural Flow Beta Shape

However, the algorithm can be refined further, in order to give a result that encourages the erosion of large spaces. Instead of walking through all the border edges, it would intuitively be better to only refine the largest candidate, this means that only one point is added in a single iteration.

This ads an extra ordered list to the complexity for the exterior edges, forming the border, with insertion time of $O(n \log n)$ and popping in $O(1)$, thus maintaining the overall complexity.

The results are more natural than those of the original algorithm, following the shape generated by the points (flowing with the geometry).

Situations like the one depicted in Fig. 7 can appear if the document contains white space or separators [13] and the segments are easily interpretable by a classification algorithm (multicolumn, strange layout, etc.).

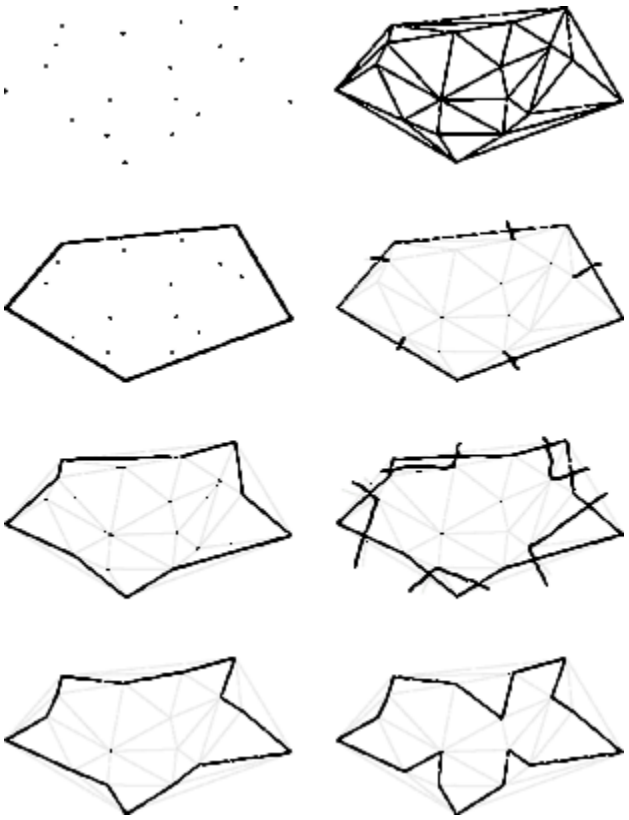

Fig. 6 Steps for generating the Beta-Shape using Triangle Erosion

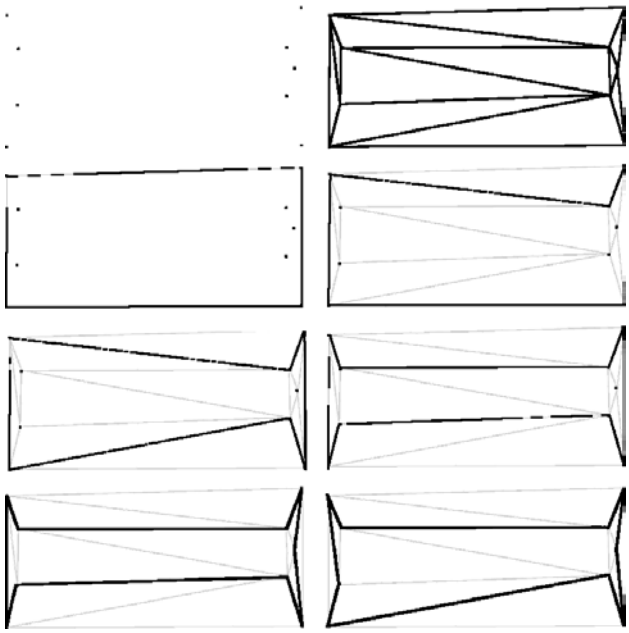

Fig. 7 The bottom images are the result of the proposed (left) and original (right) algorithms

VII. Conclusion

We propose an alternative way to compute the beta shape, reducing the overall complexity. Also, a more naturally flowing geometry can be generated with a small improvement.

The refinement of the Beta Shape may continue just until a number of "obstacle" points have been left outside the contour [14], thus skipping additional refinement steps when this is not needed. If the aforementioned situation occurs, the resulted Beta Shape may still contain edges whose length are greater than some expected thresholds, making this situation ideal not for obtaining a tight-fitting contour, or for shape-related statistics but instead very useful for geometrically separating input point sets defined as "inside" and "outside" (or "obstacle").

Fig. 8 Example of processed image

Fig. 8 displays the shapes returned by an overlapping removal phase on a layout analysis result [15] from a typical newspaper fragment. This is a real-life test scenario, where Beta-Shape has been integrated in an automatic content conversion system. The constructed shapes are:

- 1, 3, 4, 5, 10: Bounding Rectangles;
- 2: OBB;
- 6, 8, 9: Rectangular-Shape with some corners reconstructed up to the Bounding Rectangle;
- 7: Beta-Shape simplified by successive point removal until one intersection with other shapes occurred

The method can be generalized for the N-dimensional case, as the Voronoi can be calculated for any dimension, at the expense of complexity (but can be simplified at an initial stage), and an N-dimensional triangulation can result by connecting the neighbors through the Voronoi edges.

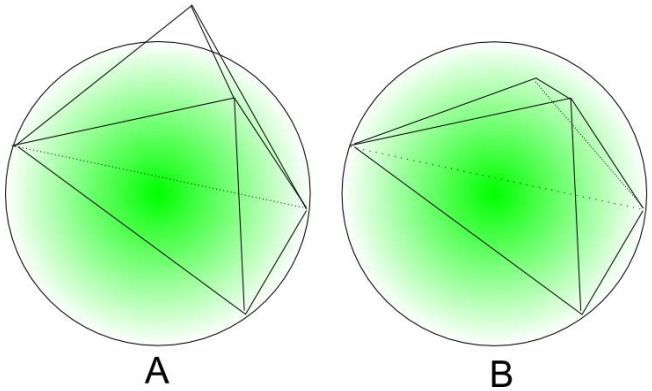

Fig. 8 Image A satisfies the Delaunay triangle, and Image B doesn't, because it includes a point from a different tetrahedron inside the bounding sphere

The Delaunay criteria can be ported to three dimensions by stating that each tetrahedron has to have an empty circumsphere (no other points inside the circumsphere).

In order to erode the tetrahedrons in 3D space, we will go line by line, not tetrahedron by tetrahedron, thus removing the largest exterior edge with every iteration (in 3D space a sorting of the points doesn't have any meaning).

The result is a single geometry, at high resolution, useful for containing spread data as close as possible, useful for many of the applications of computational geometry.

REFERENCES

[1] C. A. Boiangiu, M. Zaharescu, I. Bucur, "Building Non-Overlapping Polygons for Image Document Layout Analysis Results", *The Proceedings of Journal ISOM*, Vol. 6, No. 2, 2012, pp. 428-436.

[2] A. Hussein, "A hybrid system for three-dimensional objects reconstruction from point-clouds based on ball pivoting algorithm and radial basis functions," *WSEAS Transactions on Computers*, vol. 4, no. 7, July 2005, pp. 796-804.

[3] Y. Guang, J. Shiming and C. Shengyong, "An improved normal-free BPA algorithm for 3D surface", 7th *WSEAS Int. Conf. on Applied Computer & Applied Computational Science* (ACACOS '08), Hangzhou, China, April 6-8, 2008, pp. 477-481.

[4] C. A. Boiangiu, B. Raducanu, S. Petrescu, I. *Bucur, "Ensure non-overlapping in document layout analysis", 20th DAAAM World Symposium*, Austria Center Vienna (ACV), 25-28th of November 2009, pp.327-328.

[5] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Berlin, 1987.

[6] D. Avis, D. Bremner, "How good are convex hull algorithms?" in *Proc. 11th A.C.M. Symposium on Computational Geometry*, June 1995, pp. 20-28.

[7] N. Akkiraju, H. Edelsbrunner, M. Facello, P. Fu, E. P. Mucke, C. Varela, "Alpha shapes: definition and software," in *Proc. Internat. Comput. Geom. Software Workshop*, Minneapolis, 1995.

[8] F. Cazals, J. Giesen, M. Pauly, A. Zomorodian, "Conformal Alpha Shapes", *Eurographics Symposium on Point-Based Graphics*, 2005, pp. 55-61.

[9] C. A. Boiangiu, "The Beta-Shape Algorithm for Polygonal Contour Reconstruction", CSCS14, *The 14th International Conference on Control System and Computer Science*, 2003.

[10] S. Har-Peled, "*On the expected complexity of random convex hulls*". Technical Report 330, School of Math. Sciences, Tel-Aviv University, 1998.

[11] C.A. Boiangiu, M. Zaharescu, "Beta-shape using Delaunay-based triangle erosion", in *Proc. 18th WSEAS International Conference on Applied Mathematics* (AMATH '13), Budapest, Hungary, December 10-12, 2013, pp.97-101.

[12] S. Peterson, "Computing constrained Delaunay triangulations in the plane", *http://www.geom.uiuc.edu/~samuelp/del_project.html*, accessed on 23 December 2013.

[13] C. A. Boiangiu, D. C. Cananau, A. C. Spataru, "Modern approaches in detection of page separators for image clustering," *WSEAS Transactions on Computers*, vol. 7, no. 7, 2008, pp. 1071-1080.

[14] C. A. Boiangiu, I. Bucur, A. I. Dvornic, "The beta-star shape algorithm for document clipping", *Annals of DAAAM for 2008, Proceedings of the 19th International DAAAM Symposium, DAAAM Internationa*l (Vienna, Austria), 2008, pp. 0127–0128.

[15] C.A. Boiangiu, D.C. Cananau, B. Raducanu, I. Bucur, "A hierarchical clustering method aimed at document layout understanding and analysis", *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 2, no. 1, 2008, pp. 413-422.