SELF-REFINING MESH ALGORITHM TO DETERMINE THE VORONOI DIAGRAM

David M. Ochoa González

João Carlos Espíndola Ferreira

ochoadavid@gmail.com

j.c.ferreira@ufsc.br

Universidade Federal de Santa Catarina

Departamento de Engenharia Mecânica, Florianópolis, SC, 88040-900, Brazil

Abstract*. The Voronoi diagram and other skeleton or centerline representations of a shape (e.g. the medial axis) are commonly used as tools that allow their transformation and analysis. In those representations the geometric information of the boundaries, usually given as an input, is substituted or complemented by a shape's centerline geometry (centerline in the case of a 2D polygon). Many algorithms have been proposed to obtain the Voronoi diagram from a group of geometric elements, called sites, usually varying from the type of geometric elements that can be used (i.e. points, linear segments, curves, and patches). In this work a self-refining mesh-based algorithm for finding every element in a Voronoi region composed of 2D shapes is presented, which has the following stages: (a) a regular mesh of the region is created, and for every cell a possibly related list of influence sites is calculated; (b) the cells with two or more related sites are subdivided into smaller cells and the process is repeated. This process allows: (a) the determination of an approximated version of the Voronoi region and, consequently, the Voronoi diagram, (b) knowing which site-to-site relationships (and diagrams) need to be calculated in order to obtain the complete diagram (or a section of it). Along with the algorithm, the implementation using Python language and the obtained results are presented.*

Keywords: *Voronoi Diagram, Self-refining Mesh, Medial Axis, Computer-Aided Manufacturing.*

1 INTRODUCTION

Since proposed by Blum (1967), the centerline shape representations (Voronoi Diagram or Medial Axis Function) have been widely studied and used in many areas, including ecology, material sciences, computer animation, path planning, and computer-aided manufacturing. Many algorithms have been proposed to generate those diagrams in a computationally-efficient manner.

Historically, the first algorithms studied referring to the Voronoi diagram only dealt with diagrams generated by a number of points. Those points (usually referred to as "sites") are always completely surrounded by a Voronoi region, which contains all the points that are nearer to a given site than to any other. The Voronoi diagram is formed by combining all the Voronoi region boundaries and hence by all the points that are equally distant from two or more sites. In a simple case of point-only sites the resulting Voronoi diagram is formed only line segments. Obtaining this kind of diagram is usually accomplished by adding sites and modifying the diagram with every addition (Fortune, 1987). Similarly, when those sites are distributed in a 3D space, the resulting diagram will be formed by planar patches limited by line segments.

However, when other (more complex) elements are used as sites the resulting diagram description also becomes more complex. The difficulty in determining a shape diagram depends on two factors: the geometric complexity of the boundary elements, and the amount of elements. Therefore, using high order curves (or patches) in the boundary description will produce high order segments in the Voronoi diagram. On the other hand, a high amount of elements usually leads to a diagram formed by a higher number of distinguishable segments.

In any case, it is important to point out that the elements (i.e. points, line segments, and patches) that compose those boundaries are always produced by the relationship between two or more sites. The process of finding the relationships that will actually produce elements that will be part of the final diagram is an important part of the process, although it is time consuming. Consequently, many of the existing algorithms focus on finding those relationships, which can be accomplished in different ways: grouping the sites by regions and dealing with a lower number of elements, obtaining the Voronoi diagram for each group integrating them only in the end (Lee, 1982), adding the sites one by one in a random order (Devillers, 1992), or by a sweep line (Jin et al., 2006). Notice that those procedures are similar to the point-only diagram finding method, but they add the complexity resulting from the use of different kinds of sites.

After identifying the relationships of segments or patches that describe the corresponding geometries, they are added to the diagram. They are formed by bisectors, which are calculated from the related sites. For example, from two related straight line segments it is possible to describe the diagram using straight and parabolic segments (Lee 1982). Hence, the ability to describe the segment in an exact manner depends on the existence of this mathematical description of the related element's bisector. An example of this, with two line segments, is shown in Figure 1.

A different method, using pixels, have been proposed to deal with the shapes described. In this case, finding the minimum distance to the shape's boundary can be simplified by counting the number of pixels between a point and the boundary. Algorithms for obtaining the centerline diagrams (formed in this case by a group of pixels) have been proposed (di Baja and Thiel 1996, Hoff et al. 2000).

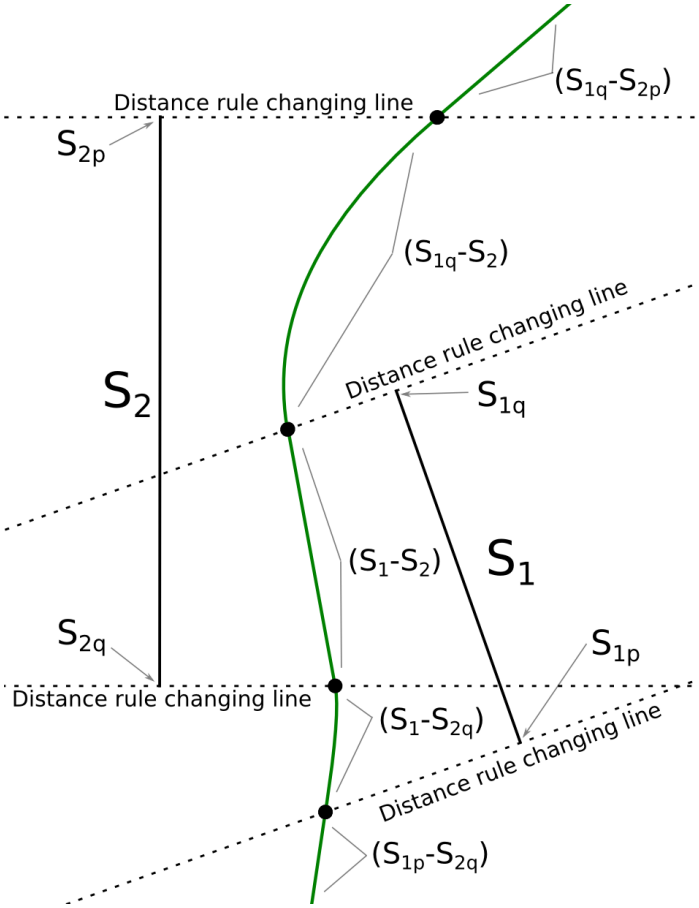

Figure 1. Example of a Voronoi diagram composed by two line segments S_1 and S_2

For its use in computer aided-manufacturing, specifically in trajectory generation (Held and Held and Spielberger, 2009; Hinduja et al., 2010; Ibaraki et al., 2010), such methods have some limitations. While the generation of the complete precise diagram is limited to certain boundary geometries, the computational cost of calculating an accurate diagram using a pixel-based method is high. With this purpose Held (2001) presented a specialized algorithm, but it can be applied only to line segments.

2 DESCRIPTION OF PROPOSED METHOD

The algorithm presented in this work uses a mixed approach to find an approximate Voronoi diagram. By dividing the domain in small rectangular areas, similar to the use of pixels, it is possible to filter the complete list of sites, shortening the time needed to calculate the diagram. The selective subdivision of those areas permits obtaining a detailed segment of the diagram only where it is necessary. A similar approach was used by Etzion and Rappoport (2002) for obtaining Voronoi skeletons of a 3D polyhedron.

2.1 General grid and parent cells setup

The first step in the proposed method is the creation of a group of parent cells, which are squared and aligned, forming a regular grid. The shape is analyzed and its extension on the

plane is calculated. With this information a number of desired cells are positioned along the longest side of the grid. Every cell includes a list of the related (or governing) segments, and initially such list is filled with the boundaries of the shapes. An example of this grid setup is presented in Figure 2.

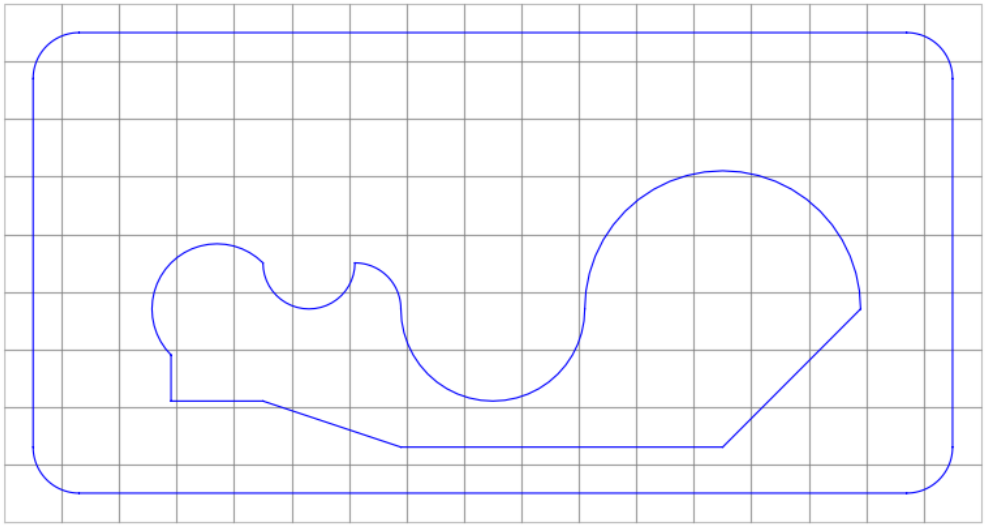

Figure 2. General grid setup (in gray) of a given example shape (in blue)

2.2 Site relationship fast filtering

Once having created the cell, the distance from its center to every one of the related segments is calculated. The segments are sorted using this distance and the smallest is used as reference. All the segments whose calculated distance is at least the minimum distance plus the cell's diagonal length are deleted from the list.

Evidently, every cell having only one element in its related segment list is fully contained on that element's Voronoi region and, therefore, any part of the diagram will be located in it. The work with those cells is considered to be finished and they will not be used anymore.

2.3 Approximate Voronoi segment determination and cell refinement

After filtering the cells having only one related segment, the next step is to create a fast approximation of the segment(s) of the Voronoi diagram that lay inside every cell. This is accomplished by performing a linear interpolation of the difference between the distance value to each cell node, and finding the zero-crossing points. This process is shown in Figure 3: considering the black cell shown at the bottom of the figure, the distance to both sites is calculated, and their values are represented as heights, forming the red and blue shapes. Then, using the intersection points of those shapes, the Voronoi segment is created (shown in green).

On the other hand, knowing that the diagram line quality (e.g. being formed by a straight or parabolic segment) changes when it crosses some site related geometries. As an example, the lines marked *Distance rule change line* (two straight lines perpendicular to the segment, crossing on its endpoints) shown in Figure 1 limit the straight an parabolic segments of the Voronoi diagram. Integrating this knowledge to the grid and, in consequence, to the diagram, leads to a better diagram approximation. To this end three mechanisms for cell refinement are applied, and thus "child" cells are created inside the original cells.

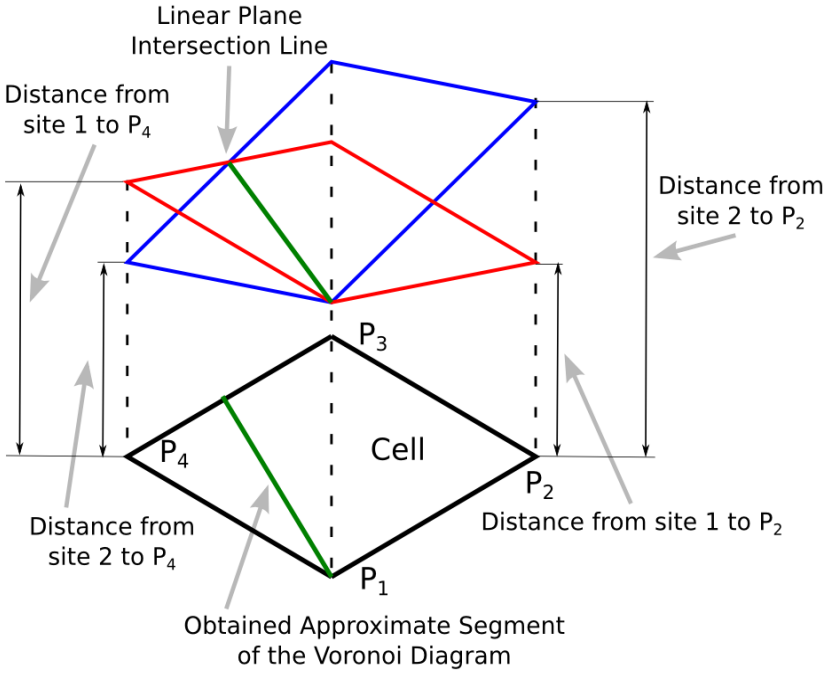

Figure 3. Approximate linear segment of a Voronoi diagram in one cell

Cell refinement rule #1: If there are one or more site endpoints inside the cell, the cell is subdivided, leaving each endpoint in a corner of the new children cells (Figure 4). This is the only rule that does not need the Voronoi segments to be calculated and, given that the endpoints does not change, it only needs to be applied once to the grid.

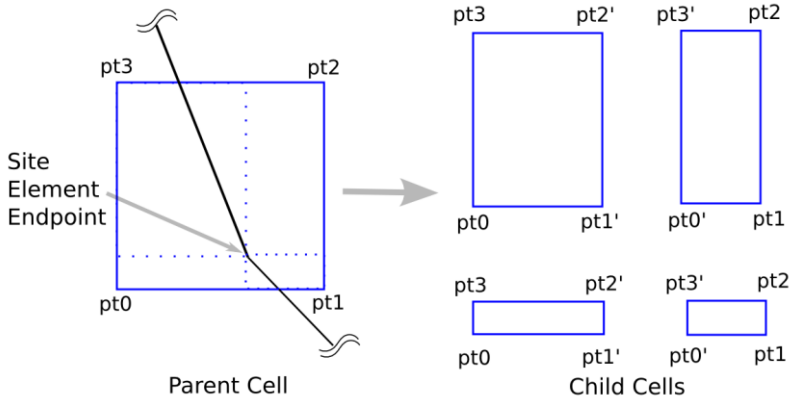

Figure 4. Example of cell refinement rule #1

Cell refinement rule #2: As noted before, some site geometries have different distance rules depending on the relative position to them, creating distinguishable regions and consequently producing predictable changes in the quality of the Voronoi segments. In order to incorporate this information to the diagram, temporary non-rectangular cells are used in the following way: (a) the *Distance rule change line* (described above) are obtained for every related cell, (b) using these lines the cell is subdivided, and this process is repeated for every site, until all the temporary cells are obtained, (c) those cells are used to obtain approximated Voronoi segment, (d) from those segments the endpoints located inside the original cell (not on its boundaries) are used to subdivide the cell. This process is shown in Figure 5.

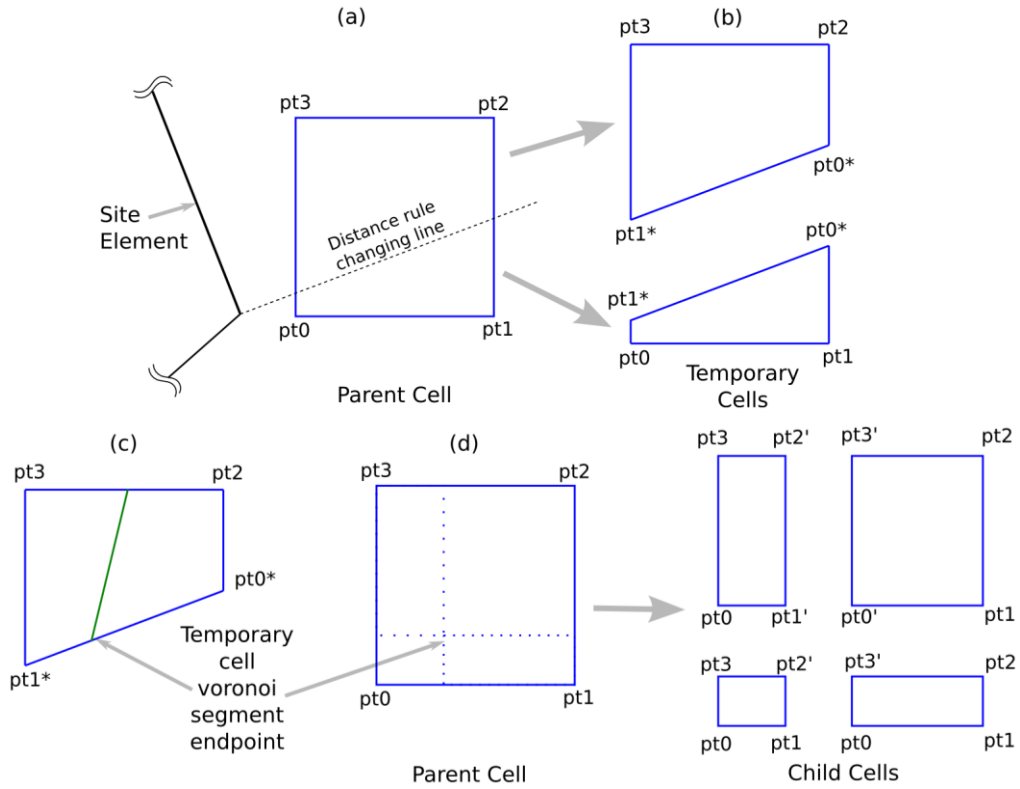

Figure 5. Example of cell refinement rule #2

Cell refinement rule #3: If none of first two rules is applied, the precision of the obtained Voronoi segments is checked against a global maximum tolerance value, established by the user, in its endpoints and in the middle point by calculating the difference of the segment distance to every pair of related cells. If the segment fulfills this tolerance value the cell will no longer be refined. The obtained segment is kept and used in the final diagram. If the segment error is still larger than the established tolerance, the cell is subdivided into equally sized children cells (Figure 6).

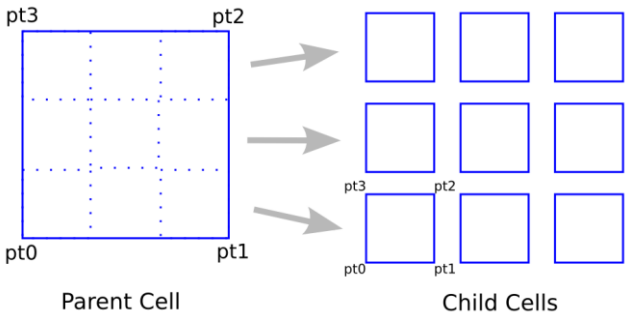

Figure 6. Example of cell refinement rule #3

The number of divisions *N* created in every cell by this rule is determined by the user. It was found that for *N=3* (*3x3* cells) a good balance is provided between speed of detail determination and filtering out areas without Voronoi segments. As heavily distorted cells are sometimes generated, having one side length considerably larger than the other resulting from applying rules #1 and #2, a special rule that produces *Nx1* or *1xN* cells depending on the cell orientation is also included, effectively limiting the propagation of heavily distorted cells.

2.4 Iterative refining and maximum level stop rule

On the newly created children cells, the site relationship fast filtering rule (described in section 2.2) and cell refinement rule (section 2.3) are repeatedly applied creating "grandchildren" cells, "great-grandchildren" cells, and so on.

Along the cells treated by rules #1 and #2 the shape of the diagram is also modified at all points where three or more of the its branches meet. Nevertheless, and contrasting with the points treated with those rules, identifying and localizing the intersections is desirable in some applications. Therefore, in order to detect them precisely this cell refinement is allowed to continue on those cells, which are identified by the following rules: (a) they have three or more related segments, (b) they produce at least one Voronoi segment. However, a maximum level rule in cell refinement needs to be implemented, in order for the computation to stop at some point, obtaining the location of those points with a certain precision.

2.5 Linking the Voronoi segments

As a result from the proposed algorithm, a number of straight line segments is obtained, each one of them being a part of the complete Voronoi diagram. However, in order for them to be used, they should to be linked together describing a graph. Thus, the last step in the proposed algorithm is to link them together, which is accomplished by searching among the nearby cells for other Voronoi diagram segments, and adding a pointer to them.

In order to increase the efficiency of this procedure, the cells are linked together also by pointers, having a different pointer to every direction. However, since only the largest cells have aligned boundaries, which is a consequence of the subdivision rules #1 and #2, every cell is linked to its parent (and the parents to their children), allowing them to easily find those cells where a related segment could be located. The linking of cells is shown in Figure 7.

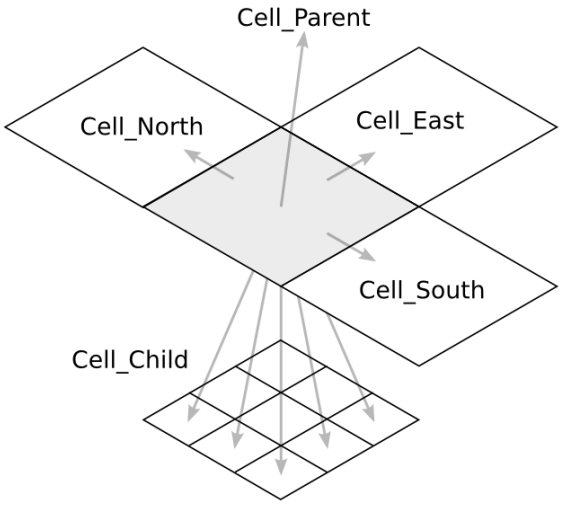

Figure 7. Cell linking

3 IMPLEMENTATION

The proposed method was implemented using the Python language following an object-oriented paradigm. The classes were implemented on top of basic geometric (i.e. point, line, segment) classes. There are basically four classes:

Site classes: Those classes are used to describe a geometry boundary. Point, straight line segment, and circular arc have been implemented, but other geometries could also be implemented. They include methods to obtain a distance to a point and to describe the boundaries of its distance rules (to be used in the cell refinement rule #2).

Grid class: This class creates the first group (parent) cells, and implements the cell refinement rule #1. Then it calls cell the refinement and Voronoi segment methods. It also contains tools for graphical representation (used for testing and debugging).

Cell class: Responsible for most of the calculations, implementing the cell refining rules #2 and #3, and creates the Voronoi segment. It has pointers to nearby cells, as well as to parent and children cells.

Voronoi segment class: A 3D straight segment class (having values for *x*, *y*, and *minimum distance to the boundary*). It also includes pointers to connected segments.

In Figure 8 a Voronoi diagram obtained using this implementation of a given shape is presented. This shape was created using a maximum refinement level of 5, having 17 cells on the largest side of the original grid, and using 3x3 divisions in the cell refinement rule #3. The computation time was 11.17s in a 1.4GHz i3 processor. The shape used has eighteen segments (eight on the outside and ten on the inside), being nine of them linear and nine circular. The obtained Voronoi diagram is composed by 3955 segments. In Figure 8 the diagram was colored to show the relative distance to the shape boundaries.

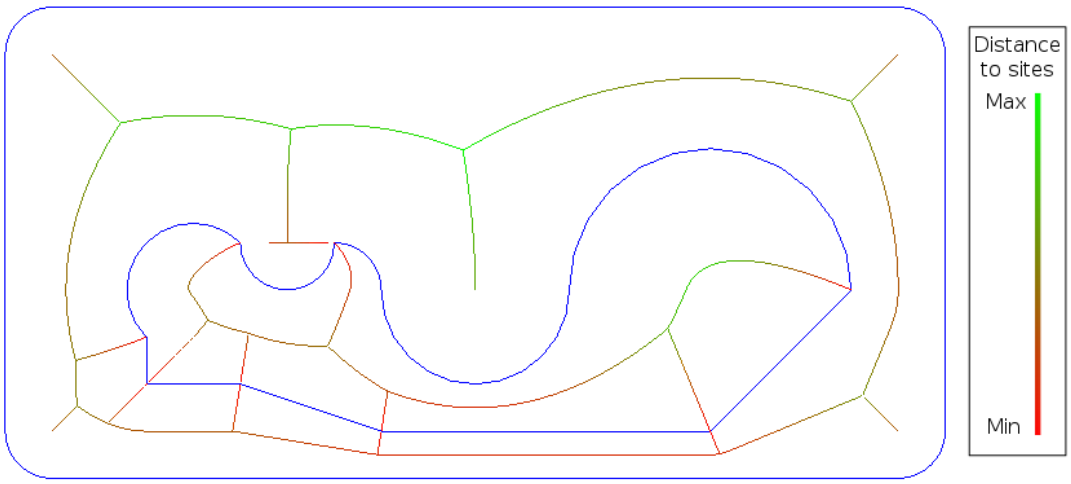

Figure 8. Obtained Voronoi diagram (red to green) for a shape (blue)

In Figure 9 the same shape and Voronoi diagram are presented, but it shows the cells used during the computation.

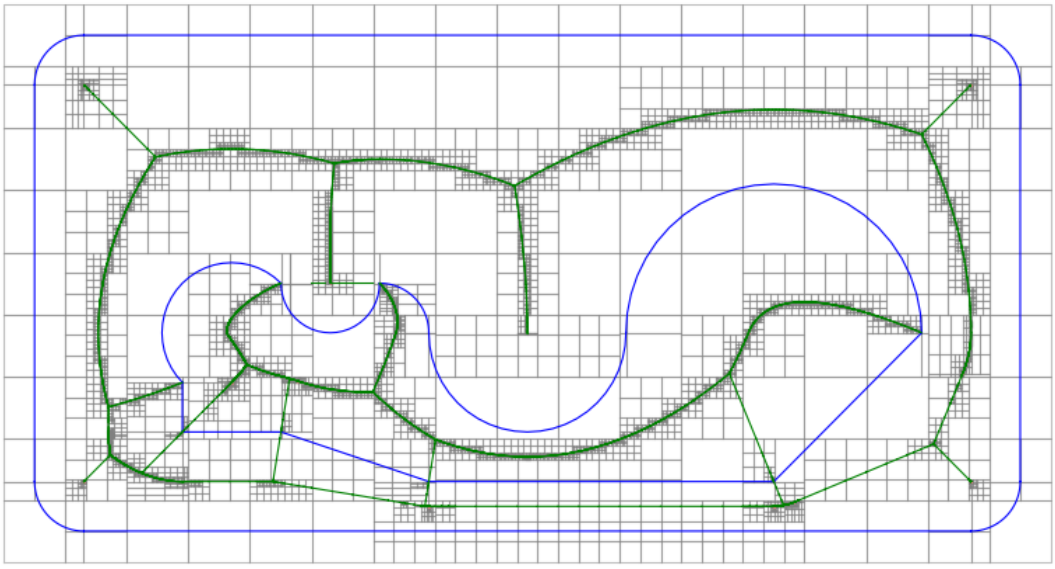

Figure 9. Voronoi diagram from Figure 8, showing the used cells (in gray).

4 RESULTS

The method was applied to another geometry, which consists of two concentric circular shapes, whose radiuses are 3.0 and 5.0 units. The resulting diagram corresponds to a circle with the same center as the original shapes. In this example each side of the grid was divided into nine cells, and a *3x3* value was used for cell refinement rule #3**.** An example of the diagram obtained for this geometry is shown in Figure 10, and the results for each refinement level (i.e. number of generations) are presented in Figure 11. In this case, since the exact diagram is known (which is a concentric circle with a 4.0 unit radius), the error can be measured by calculating the distance from the obtained Voronoi diagram with this circle. The maximum value of those distances is also presented in this graph.

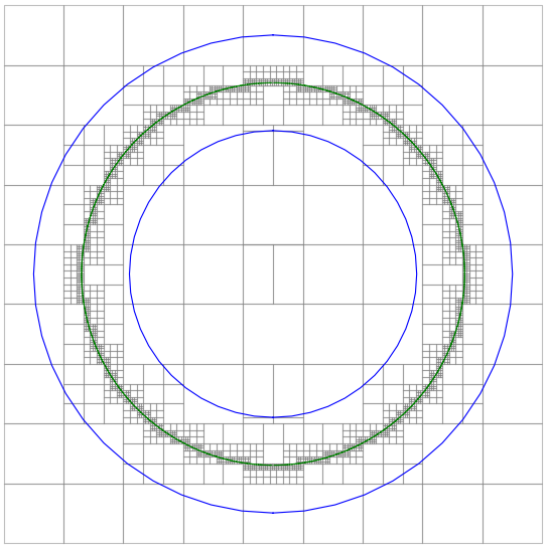

Figure 10. Shape used to evaluate the error of the obtained results.

This case is studied as the worst case scenario because the refinement process continues indefinitely for all the cells that contain a portion of the Voronoi diagram if the maximum

tolerance value is set to a small value (any value lower than 2.79×10^{-4} in this example). However, this situation allows evaluating the implementation: since the maximum error value is reduced every time that the cells are refined, it proves that the refinement lead to a higher precision attained by the algorithm.

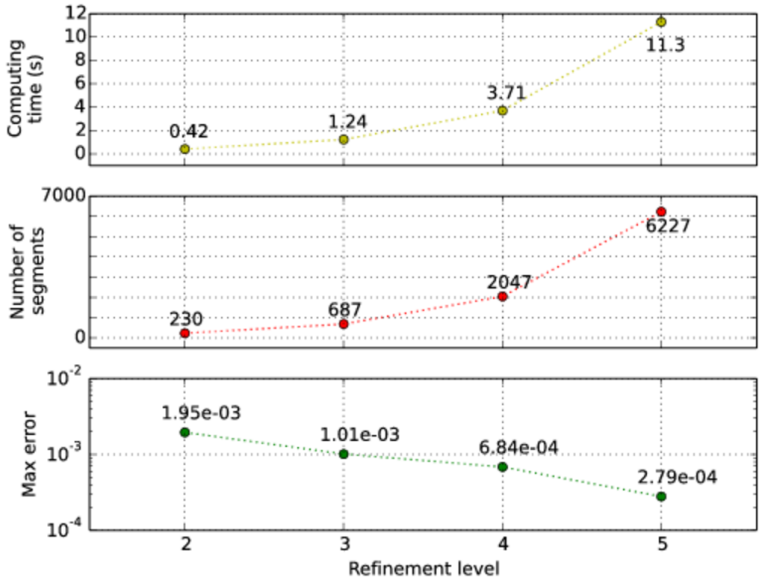

Figure 11. Results from each refinement level of the analyzed shape

5 APPLICATION ON PETROGRAPHIC MICROSCOPE IMAGES ANALYSIS

One possible application of the proposed method is the analysis of petrographic microscope images, which can be done by performing the following steps: (a) initially an image should be taken and the polygons that limit specific regions should be obtained, (b) by applying the method described in this paper the complete Voronoi diagram can be obtained, (c) use selected segments of the obtained diagram to calculate the internal angles of the polygons. While there are computational tools to achieve the first step, those tools will not be described here, since it is beyond the scope of this paper. For calculating the internal angles of the shape the following method is proposed:

(a) Using the distance-to-site values obtained in the creation of the Voronoi segments (the height values of the Figure 3) the value of the difference between the start and end points is obtained.

(b) This distance is divided by the segment length, resulting in a "segment slope value". In general, this value is the rate of change of the distance to the sites along the Voronoi diagram.

(c) However, in the particular case when the Voronoi segment is related to the bisector of two straight lines (as the part of the diagram marked as S_1-S_2 in Figure 1), the angle formed by those lines is obtained using Equation (1).

$$angle = 2 \cdot \arcsin(slope) \tag{1}$$

This whole process is shown in Figure 12. In Figure 12(a) an example microscopic image is shown (taken from: http://www.nslc.ucla.edu/pet/mineral_html/Nosean.html). The polygons obtained are shown in Figure12(b), and the resulting diagram, as well as some selected angle values, are presented in Figure 12(c).

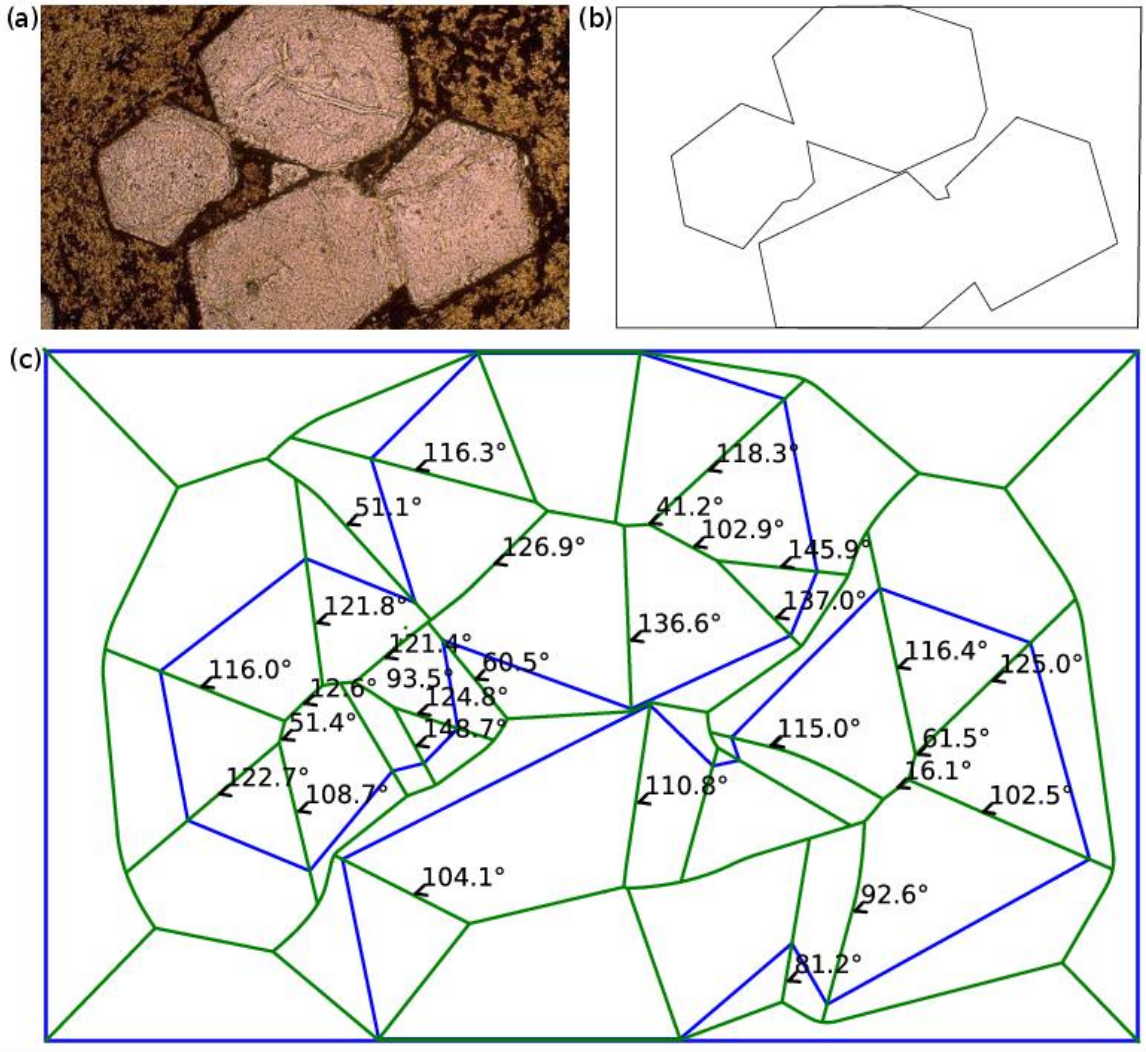

Figure 12. Application of the proposed method in the analysis of petrographic microscope images

6 CONCLUSIONS AND FUTURE WORK

This work presented a method to obtain an approximated version of a Voronoi diagram using a self-refining mesh algorithm. It leads to adequate results in the calculation of the Voronoi diagram for a given shape described by straight lines and circular segments. The proposed method is also capable of calculating any section of the Voronoi diagram with arbitrary precision, and this precision influences directly the processing time.

We intend to continue the development of the algorithm in two areas: (a) integrating other kinds of elements (for example, Bézier curves), (b) integrating the proposed method with a library that will be used to generate a milling tool path, which is the current focus of our research.

Another possible future work could be the implementation of a similar method for finding center geometries of 3D shapes.

ACKNOWLEDGMENTS

This study was financially supported by the National Council for Scientific and Technological Development (CNPq) of Brazil.

REFERENCES

Blum, H, 1967. A transformation for extracting new descriptions of shape. In Wathen-Dunn W., ed, *Models for the Perception of Speech and Visual Form*, pp. 362-38. MIT Press.

Devillers, O. 1992. Randomization yields simple O(n log* n) algorithms for difficult Ω(n) problems. *International Journal of Computational Geometry & Applications*, vol. 02, n. 01, pp. 97-111.

di Baja, G. S.; Thiel, E. 1996. Skeletonization algorithm running on path-based distance maps. *Image and Vision Computing*, vol. 14, n. 1, pp. 47-57.

Etzion, M.; Rappoport, A. 2002. Computing Voronoi skeletons of a 3-D polyhedron by space subdivision. *Computational Geometry*, vol. 21, n. 3, pp. 87-120.

Fortune, S. 1987. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, vol. 2, n. 1-4, pp. 153-174.

Held, M. 2001. VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Computational Geometry*, vol. 18, n. 2, pp. 95-123.

Held, M.; Spielberger, C. 2009. A smooth spiral tool path for high speed machining of 2D pockets. *Computer-Aided Design*, vol. 41, n. 7, pp. 539-550.

Hinduja, S.; Mansor, M. S. A.; Owodunni, O. 2010. Voronoi-diagram-based linking of contour-parallel tool paths for two-and-a-half-dimensional closed-pocket machining. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 224, n. 9, pp. 1329-1350.

Hoff, K. E., Culver, T., Keyser, J., Lin, M., Manocha, D. 2000. Fast computation of generalized Voronoi diagrams using graphics hardware. *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*. SCG'00, pp. 375-376. ACM Press.

Ibaraki, S.; Yamaji, I.; Matsubara, A. 2010. On the removal of critical cutting regions by trochoidal grooving. *Precision Engineering*, vol. 34, n. 3, pp. 467-473.

Jin, L., Kim, D., Mu, L., Kim, D., Hu, S. 2006. A sweepline algorithm for Euclidean Voronoi diagram of circles. *Computer-Aided Design*, v. 38, n. 3, pp. 260-272.

Lee, D. T. 1982. Medial axis transformation of a planar shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, n. 4, pp. 363-369.