A fast approximation of the Voronoi diagram for a set of pairwise disjoint arcs

Dmytro Kotsur
Taras Shevchenko National
University of Kyiv
64/13, Volodymyrska, st.,
Kyiv, Ukraine
dkotsur@gmail.com

Vasyl Tereshchenko
Taras Shevchenko National
University of Kyiv
64/13, Volodymyrska, st.,
Kyiv, Ukraine
vtereshch@gmail.com

ABSTRACT

We propose a method for fast approximation of the Voronoi diagram for a set of pairwise disjoint arcs on a plane. The arcs are represented by parameterized curves. A set of input curves is discretized into partition set, for which the Voronoi diagram is constructed. After merging corresponding Voronoi cells and removing redundant edges, the Voronoi graph is approximated by Bezier curves. We also propose the elaboration and optimization of the approximation. The total complexity of the algorithm is $O(N \log N)$ in the worst-case.

Keywords

Approximation, Voronoi diagram, Voronoi cell, Bezier curve, discretization, partition set, parametric curve.

1. INTRODUCTION

Relevance. Today there is a wide variety of algorithms for solving problems of computational geometry. But typically, the scope of these algorithms is very narrow. For example, well-known algorithms for constructing the Voronoi diagram (e.g., "divide and conquer" [Prep85b], Fortunes algorithm [Fort87a]) can be effectively applied only to a set of points and line segments. However, when we solve practical problems, we usually deal with more complex geometrical shapes than just points or line segments. Normally these complex objects can be represented by parametric curves of arbitrary shape [Aic10a].

However, the construction of the exact Voronoi diagram for the set of parametric curves is not a trivial task. Even in the simplest example of the Voronoi diagram for two arbitrary disjoint curves (which is just a bisecting curve between them) we have to consider a large number of particular cases. The construction of the Voronoi diagram for three or more objects represented by parametric curves is even more sophisticated and would require a huge amount of computational time. Therefore, for practical applications it is reasonable to reduce the problem of exact Voronoi diagram construction to a problem of its approximation construction, which can be performed in a reasonable computational time. However, the problem of constructing approximations of Voronoi diagram is not trivial and still requires further study.

Analysis of recent research and publications. In a large amount of works devoted to approximation of Voronoi diagram, authors consider the spatial discretization of 2D plane into cells using a discrete grid with a fixed step [Sud06c], [Hof99c]. In this case, the plane is sampled and then the result (discrete image) is used for further transformations. After performing all necessary transformations, the Voronoi diagram is obtained. A significant drawback of these methods is that the accuracy of the constructed Voronoi diagram depends on the size of the grid, and reducing its size leads to a significant increase in the number of cells (quadratic dependence). Thus, in order to achieve acceptable results in terms of precision, we would require a lot of computational power. Therefore, these methods can be extremely time-consuming in some cases.

Another approach is an approximation by means of constructing the Voronoi diagram for the simplest geometric objects, such as points or line segments. In particular, in [Ho09c] authors demonstrate an approach to approximate the Voronoi diagram for arbitrary geometric objects using Bezier curves and taking into account the Voronoi diagram for points.

However, this approximation is made only for the part the Voronoi diagram (they solve the problem of finding the minimum path). The approximation by means of the Voronoi diagram for points has significant prospects as well as it makes possible to operate with less amount of simple objects. The points on the curves can be selected based on the discretization step, which can be fixed or can depend on the characteristics of the curve. It allows to speed up the construction of approximation, and at the same time, to maintain the desired accuracy in critical regions. Since the method is based on the construction of the Voronoi diagram for simple geometric objects (points or line segments), the critical regions of such approximation can be easily refined by supporting dynamic Voronoi diagram for points and inserting new points (or line segments), where it is necessary.

Also there are some attempts to compute exact Voronoi diagram [Ary02c], [Ary02b], [Har01c]. In paper [Seo08c] authors describe an algorithm for computing the precise Voronoi Diagram of planar freeform curves, which represented with a parametric form. The authors try to build the precise bisector between two curves and precise junction points. But this effort leads to the necessity of solving the system of three nonlinear equations, which is not trivial task itself, since it requires time-consuming numerical methods. Another similar approach is described in [Ram99a]. It has an asymptotic time of $O(n^2)$, where n is a number of curves. This method has the same drawback as previous: computation of junction points (authors use Newton-Raphson method).

Novelty and ideas. The purpose of this paper is to develop an algorithm for fast and accurate approximation of Voronoi diagram, which has $O(N \log N)$ complexity in the worst case. Our approach is based on a point sampling for input curves and further construction of Voronoi diagram for sampled points. Then we use a developed procedure for merging the Voronoi cells and approximate edges of Voronoi diagram by curves.

2. A METHOD FOR FAST APPROXIMATE VORONOI DIAGRAM CONSTRUCTION

Before describing the method and the solution of a problem we recall the basic concepts [Aur13b] used in this paper.

2.1 Basic concepts and statement problem

Definition 1. Suppose we are given a set of generator points $P = \{p_1, p_2, \dots, p_n\} \subset \mathbb{R}^{\mathrm{d}}$, where $(2 \le n < \infty)$. We call set P the generator set of the Voronoi diagram. Let's denote by I_n a set of generators indices and Euclidean distance between two objects x and y as $\rho(x, y)$. We call the region given by:

$$VP(p_i) = \{x | \rho(x, p_i) \le \rho(x, p_j)\}, \quad (1)$$

where $(j \ne i), i, j \in I_n, x \in \mathbb{R}^2$, the Voronoi cell (Voronoi polygon) associated with p_i. Then the Voronoi Diagram generated by P (or the Voronoi diagram of P) is defined as follows:

$$VD(P) = \{VP(p_1), VP(p_2), \dots, VP(p_n)\} \quad (2)$$

For any two generators p_i and p_j we define a region of dominance of p_i over p_j:

$$H(p_i, p_j) = \{x | \rho(x, p_i) \le \rho(x, p_j), x \in \mathbb{R}^d\}, \quad (3)$$

where $i, j \in I_n$. Thus, the Voronoi cell can be defined by the following statement:

$$VP(p_i) = \bigcap_{j \in I_n \setminus \{i\}} H(p_i, p_j) \quad (4)$$

The boundary or bisector between two regions of dominance $H(p_i, p_j)$ and $H(p_j, p_i)$ is denoted by $b(p_i, p_j)$ and defined as follows:

$$b(p_i, p_j) = H(p_i, p_j) \cap H(p_j, p_i), \quad (5)$$

or alternatively:

$$b(p_i, p_j) = \{x | \rho(x, p_i) = \rho(x, p_j), x \in \mathbb{R}^d\} \quad (6)$$

For a given generator set P and a set of indexes I_n, the boundary $b(p_i, p_j)$ can be denoted in a short form as $b_{i,j}$.

Definition 2. Let $C(t) = (x_c(t), y_c(t))$ be a continuous parametric curve on a plane, $t \in [0,1]$, and parameter Δt determines the step of approximation. We call set of points $P_c = \{\tilde{c}(i\Delta t) | \ i = \overline{0, n}\}$ the partition set of the curve C, where $n = \left[\frac{1}{\Delta t}\right]$.

Problem statement.

Let $\mathcal{C} = \{C_1(t), C_2(t), \dots, C_n(t)\}$ be a set of continuous parametric curves, which are pairwise disjoint. Given the partition sets $P_{c_1}, P_{c_2}, \dots, P_{c_n}$ of curves $\mathcal{C}$ and their union $\mathcal{P} = P_{c_1} \cup P_{c_2} \cup \dots \cup P_{c_n}$, build an approximation of Voronoi diagram $VD(\mathcal{C})$ for set of curves $\mathcal{C}$.

2.2 The solution for arbitrary objects

At the first we build Voronoi diagram for union of partition points $\mathcal{P}$:

$$VD(P) = \{VP(p) | p \in P\} \quad (7)$$

Let $l: P \to \mathbb{N}_+$ be the function, that for a given point returns index of curve, which this point belongs to: $l(p) = i \overset{def}{\Longleftrightarrow} p \in P_{c_i}$. Then the approximation of Voronoi cell associated with curve C_i is obtained by the union the Voronoi cells for each point in the corresponding partition set:

$$\widetilde{VP}(C_i) \cong \bigcup_{l(p)=i} V(p) \quad (8)$$

Thus, the resulting approximation of the Voronoi diagram for the set $\mathcal{C}$ is:

$$\widetilde{VD}(\mathcal{C}) \cong \{\widetilde{VP}(C_i) | i = \overline{1, n}\} \quad (10)$$

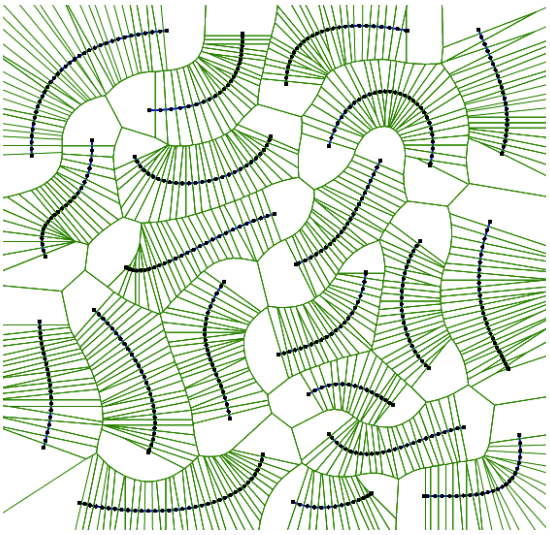

Figure 1. Voronoi diagram (green) for a set of sampled points (black) comprising the partition set of input nonintersecting Bezier curves (blue)

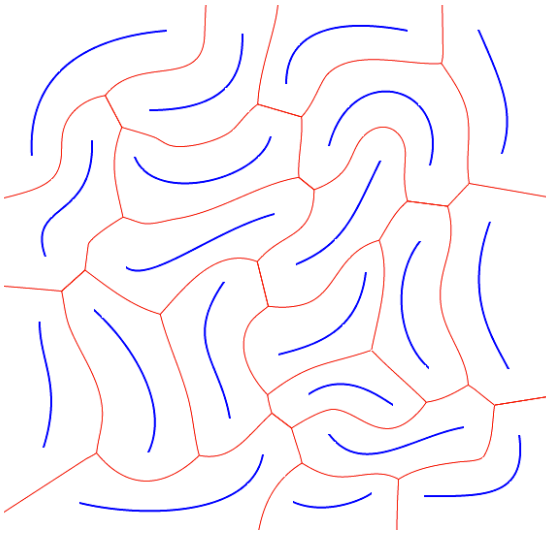

Figure 2. An example of approximate Voronoi diagram (red) for a set of 20 nonintersecting Bezier curves (blue)

Initially, we choose a certain set of curve points for constructing a Voronoi diagram, while maintaining the connection of each point with the corresponding curve. Next, we merge together the Voronoi cells, whose centers belong to the same curve, and remove the adjacent edges.

Figures 1 and 2 show an example of constructing a Voronoi diagram for 20 nonintersecting Bezier curves.

2.3 Approximating Voronoi edges with curves

After constructing the Voronoi diagram for points and merging corresponding cells (removing respective edges), we obtain an approximation by polygonal chains (each chain connects two junction points). At the next step we approximate obtained chains with curves.

At first to make the approximation by curves we should choose the canonical equation of a curve. The type of approximating curve (canonical equation) for Voronoi edge depends on the type and parameters of curves, which it separates. We consider the general case and make approximation by quadratic and cubic Bezier curves. Other types of curves may be similarly considered.

One of the most appropriate methods of approximation by curves is least square approximation. We fix the first and last points (start and end point) of Bezier curve and then use the least squares method to find best curve fit.

Thus, for quadratic Bezier curves we find the coordinates of point P_1:

$$B(t) = t^2P_0 + 2(1-t)tP_1 + (1-t)^2P_2 \qquad (11)$$

In this case the least square method is reduced to the solution of a linear equation with one variable for each coordinate, which can be easily solved:

$$\varphi(P_1) = \sum_{i=1}^{n}\left(P_i^* - B(t_i)\right)^2 \to \min \Longrightarrow \frac{d\varphi}{dP_1} = 0 \quad (12)$$

In order to approximate polyline with a cubic Bezier curve we should find x, y coordinates of two points P_1 and P_2:

$$B(t) = t^3P_0 + 3(1-t)t^2P_1 + 3(1-t)^2tP_2 + t^3P_3 \quad (13)$$

This problem reduces to the solution of the system of linear algebraic equations for each coordinate. Each system of equations consists of two equations:

$$\varphi(P_1, P_2) \to \min \Longrightarrow \begin{cases} \dfrac{d\varphi}{dP_1} = 0, \\ \dfrac{d\varphi}{dP_2} = 0. \end{cases} \qquad (14)$$

where

$$\varphi(P_1, P_2) = \sum_{i=1}^{n}\left(P_i^* - B(t_i)\right)^2 \qquad (15)$$

Thus, the total number of equations in the system of linear equations depends on the order Bezier curve.

Implementation details. For each type of approximation curve we get an analytical solution (expressions for each of unknown point), which is easy to implement in code.

3. COMPLEXITY ANALYSIS

The analysis of the complexity of the proposed method is provided in the following statements.

Theorem 1. If an input set consists of m objects on a plane represented by nonintersecting parametric curves and the total number of points used to discretize these m objects is N. Then, approximation of Voronoi

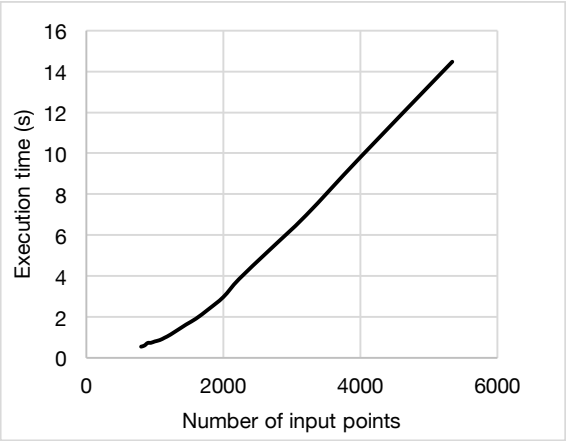

Figure 3. Execution time of the proposed method for Voronoi diagram approximation (x-axis is a number of input objects, y-axis shows an execution time in seconds)

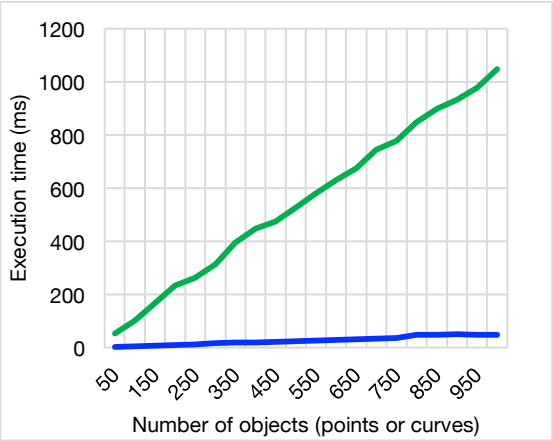

Figure 4. Execution time comparison: green curve is proposed method for an approximate Voronoi diagram construction for *N* curves; blue curve shows execution time of "divide and conquer" [Prep85b] algorithm for *N* points on a plane.

diagram for the set of m arbitrary-shaped objects represented by parametric curves, can be computed in time $O(N \log N)$.

Proof. In papers [Prep85b, Fort87a, Sha75c] authors provide the detailed description and complexity analysis of an algorithm for a Voronoi diagram construction for a set of N points in $O(N \log N)$. If Voronoi diagram is represented by doubly connected linked list, then the following lemmas hold:

Lemma 1. Merging two neighboring Voronoi cells represented by doubly-connected linked lists can be performed in $O(1)$ time.

Proof. In order to merge two neighboring Voronoi cells we should merge corresponding doubly-connected linked lists. This operation is simple pointers reassignment and it can be performed in $O(1)$ time.

Lemma 2. Approximation of the Voronoi diagram for arbitrary-shaped parametric curves on a plane can be performed using the pre-computed Voronoi diagram for points in $O(N)$ time.

Proof. An approximation of Voronoi diagram is performed by merging the neighboring Voronoi cells, whose generators correspond to the same curve. The maximal number of Voronoi cells is N and one pair of neighboring cells can be merged $O(1)$. Thus, we can get an approximation of Voronoi diagram with edges represented by polylines in time $O(N)$ (by merging all necessary pairs of cells).

Taking into account Lemmas 1, 2 and the following statement: quadratic or cubic Bezier curves fit polynomial chains in time $O(M)$, where M - number of points in chain; we can formulate following lemma:

Lemma 3. An approximation of Voronoi diagram for arbitrary-shaped objects on a plane using Bezier curves can be computed in $O(N \log N)$ time.

Therefore, the approximation of the Voronoi diagram for a set of m arbitrary-shaped objects on a plane, which are represented by non-intersecting parametric curves can be performed in time $O(N \log N)$, that concludes the proof.

4. IMPLEMENTATION DETAILS

In the implementation part we constructed the partition for a set of curves (based on the uniform point sampling) and then build the Voronoi diagram for the obtained partition using the "divide and conquer" algorithm described in [Sha75c]. In order to store the correspondence between curve indexes and points in partitioning, we used a hash map. For every index of a point it stores the index of the corresponding curve and also index of previous and next sampled point on a

Point Index	Next point index	Previous point index	Curve Index
1	2	-1	1
2	3	1	1
3	4	2	1
…	…	…	…
N_1	-1	N_1-1	1
N_1+1	N_1+2	-1	2
N_1+2	N_1+3	N_1+1	2
…	…	…	…
N	-1	N-1	M

Table 1. Example of a hash table, which maps point indices to curve indices, it also stores indices of previous and next points on a curve;

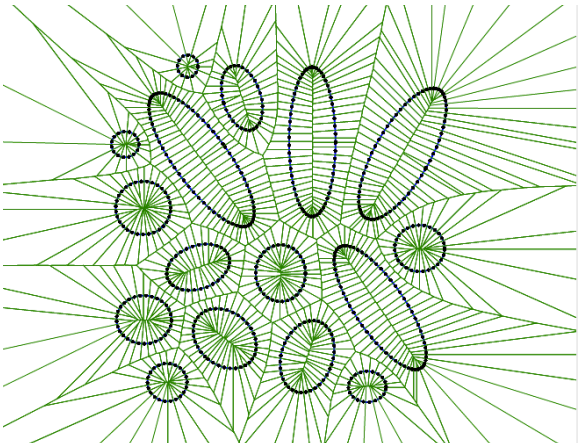

Figure 5. Voronoi diagram for a set of sampled points comprising the partition set of input curves

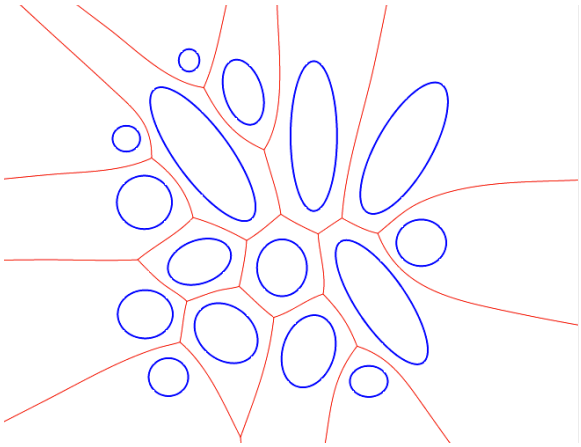

Figure 6. An approximate Voronoi diagram (red) for a set of 16 ellipses (blue)

curve (for example, see Table 1, value -1 indicates no data). Another hash table maps index of a curve to an index of one of its endpoints from partition set. At the next step we merge Voronoi cells for neighboring points of a curve and get an approximate Voronoi cell of a curve (whose edges are polygonal chains). Voronoi diagram is represented by doubly-connected edge list (DCEL). The procedure of merging is the following: we start from some partition point of curve p; run BFS(p) and iterate through all neighboring points of the same curve. At each step we merge pair of Voronoi cells corresponding to the neighboring points and remove redundant edges of Voronoi cells. During the procedure of merging we also determine junction points of the resulting Voronoi diagram.

5. EXPERIMENTAL RESULTS

The practical implementation is made in C++ using OpenGL graphic library to visualize data and Qt framework for GUI. An input of data is provided either by user manually or from SVG-files. The implemented code allows also to visualize the main stages of our algorithm (see Figures 1-2, 5-9). We also tested the performance of our method. All experiments in this paper were carried on Intel Core i7 2.3GHz processor computer with 4GB RAM.

Figure 3 illustrates the results of the execution time testing. The execution time of the proposed algorithm was compared to the execution time of "divide and conquer" algorithm as described in [Prep85b]. This comparison (see Figure 4) shows how the complexity of input objects influences the computational efficiency of the method.

Figure 4 also demonstrates the increase in computational time for curves in comparison to a set of points by the factor of approximately 20 (in case of the discretization step equal to ~0.1). The main reasons for such increase are computational overheads (curve discretization, approximation) and increase in total number of processed points.

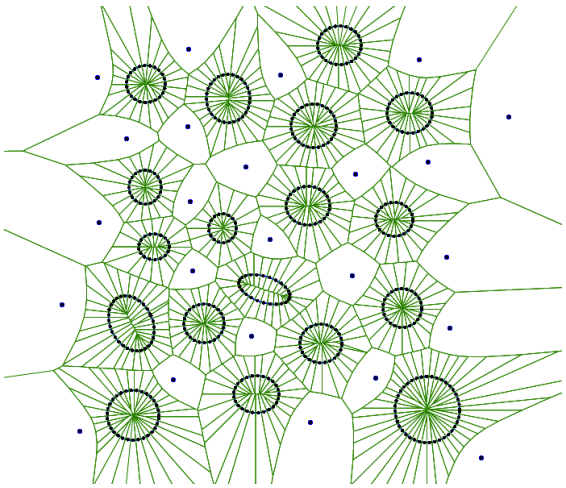

Figure 7. Voronoi diagram for a set of sampled points, which comprises a partition set

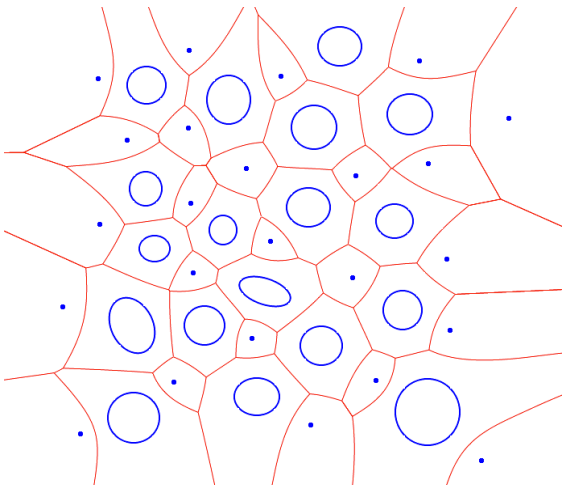

Figure 8. An approximate Voronoi diagram (red) for a set of 18 ellipses and 24 points (blue)

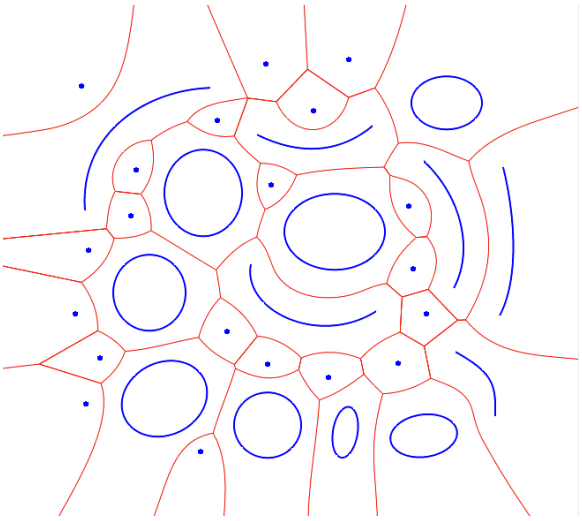

Figure 9. An example of approximate Voronoi diagram (red) for a set of 6 nonintersecting Bezier curves, 8 ellipses and 20 points (blue)

6. CONCLUSION

Thus, we propose an algorithm for approximation of the Voronoi diagram for a set of pairwise disjoint arcs on a plane. Arcs are represented by parametric curves. For curves we construct a partition set for which the Voronoi diagram is built. At the next step we perform a transformation of the obtained Voronoi diagram by merging neighboring Voronoi cells, which correspond to the same curve, and removing unnecessary edges. Thus, we obtain an approximation of Voronoi diagram with polynomial chains. We also propose to approximate these polygonal chains by Bezier curves and arcs. Cases of cubic and quadratic Bezier curves were analyzed. The type of approximating curve is chosen analytically. The total complexity of the proposed algorithm is $O(N \log N)$.

However, we do not consider the case, when curves intersect or share the endpoint(s). As it has been shown in [Ram99a] applying the technique of sampling leads to inadequacy of approximations in the mentioned situations. Topological inconsistencies [Ram99a] are also considered (during the process of merging). A significant advantage of this approach is the ability to refine approximations for critical areas, defined by specific practical problems. This approach makes it possible to refine a critical local region of Voronoi diagram by supporting dynamic data structures like concatenable queues [Sha75c].

Current research is implemented in software, the result is illustrated on Figures 1-2 and 5-9.

7. REFERENCES

[Prep85b] Preparata, F. P. and Shamos, M.I., Computational Geometry: An introduction. Springer-Verlag, Berlin, 1985.

[Fort87a] Fortune, S. A sweepline algorithm for Voronoi diagrams. Algorithmica, N 2, pp. 153-174, 1987.

[Sud06c] Sud, A., Govindaraju, N. and Manocha, D. Interactive computation of discrete generalized Voronoi diagrams using range culling. Proc. International Symposium on Voronoi Diagrams in Science and Engineering, P. 1-10, 2006.

[Hof99c] Hoff, K. E. , Culver, T., Keyser, J., Lin, M. and Manocha, D. Fast computation of generalized Voronoi diagrams using graphics hardware. Proc. of ACM SIGGRAPH Annual Conference on Computer Graphics, ACM, pp. 277–286, 1999.

[Ho09c] Ho, Y. J. and Liu, J. S. Collision-free curvature-bounded smooth path planning using composite bezier curve based on Voronoi diagram. IEEE International Symposium on Computational Intelligence in Robotics and Automation, Korea, pp. 463-468, 2009.

[Ary02c] Arya, S. and Malamatos, T. Linear-size approximate Voronoi diagrams, Proc. 13th ACM-SIAM Sympos. Discrete Algorithms, pp. 147–155, 2002.

[Ary02b] Arya, S., Malamatos, T. and Mount, D. M. Space-efficient approximate Voronoi diagrams, Proc. of STOC, pp. 721-730, 2002.

[Har01c] Har-Peled, S. A replacement for Voronoi diagrams of near linear size. Proc. of FOCS, pp. 94-103, 2001.

[Seo08c] Seong et al. Voronoi diagram computations for planar NURBS curves. Proc. ACM Symp. Solid & Phys. Modeling, NY, pp. 67–77, 2008.

[Ram99a] Ramamurthy, R. and Farouki, R. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries: I. Theoretical foundations. J.Comput.Appl.Math. 102, pp. 119–141, 1999.

[Sha75c] Shamos, M. and Hoey, D. Closest-point problems. Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci., pp. 151-162, 1975

[Ram99a] Ramamurthy, R. and Farouki, R. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries: II. detailed algorithm description. J.Comput.Appl.Math. 102, pp. 253–277, 1999

[Aur13b] Aurenhammer, F., Klein, R. and Lee, D. T. Voronoi Diagrams and Delaunay Triangulations. World Scientific Publishing Co., 2013.

[Aic10a] Aichholzer, O., Aigner, W., Aurenhammer, F., Hackl, T., Jüttler, B., Pilgerstorfer, E., Rabl, M. Divide-and-conquer for Voronoi diagrams revisited". Computational Geometry: Theory and Applications, 2010, V 43, Is. 8, P. 688-699.