Hierarchical Planar Voronoi Diagram Approximations

I. Boada N. Coll J. A. Sellarès
Institut d'Informàtica i Aplicacions
Universitat de Girona, Spain
{imma, coll, sellares}@ima.udg.es

Abstract

We present a new approach for constructing a polygonal approximation, at a prefixed level of detail, of a planar Voronoi diagram valid for generalized sites (points, line-segments, curve-arc segments, ...) and for different distance functions (Euclidean metrics, convex distance functions, ...). The approach is based on two related algorithms. The first algorithm constructs a quadtree-based Voronoi diagram codification, the Voronoi-Quadtree. The second algorithm, taking as input the Voronoi-Quadtree, provides a DCEL representation of the approximated Voronoi region boundaries associated to the Voronoi-Quadtree.

1 Introduction

The generalized Voronoi diagram [1] of a set of sites partitions the plane into regions, one per site, such that all points in a region have the same closest site according to some given distance function. Voronoi diagrams are widely used in many scientific fields and application areas, such as computer graphics, geometric modeling, geographic information systems, visualization of medical datasets, pattern recognition, robotics, shape analysis or crystal and cell growing, just to name a few (see [10]).

Algorithms for computing the Voronoi diagram, when sites are points, have been extensively developed in computational geometry and related areas [1]. These algorithms have a low computational complexity. On the contrary the computation of exact generalized Voronoi diagrams (i.e. when sites are points, line-segments, curve-arc segments, ...) use to be complicated. The algorithms that have to be applied suffer from numerical robustness problems and are time-consuming due to the numerous high precision calculations that are required. However in some applications (motion planning, geographical maps, . . .) the computation of an approximate generalized Voronoi diagram within a predetermined precision is sufficient. Several algorithms have been proposed for approximating generalized Voronoi diagrams. Lavender et al. [4] use an octree representation of objects and performs spatial decomposition to compute the approximation. In [9] an algorithm for a set of disjoint convex sites in any dimension is presented. The algorithm adaptively subdivides the space into axial cells of fixed size and computes the Voronoi diagram up to a given precision. Teichmann et al. [8] compute a polygonal approximation of Voronoi diagrams by subdividing the space into tetrahedral cells. In [3] an approach for computing generalized Voronoi diagrams in the plane using interpolation-based polygon rasterization hardware is presented.

In this paper we present an approach to approximate planar Voronoi diagrams for different site shapes (points, line-segments, curve-arc segments, ...) and different distance functions (Euclidean metrics, convex distance functions, ...). Not all the sites must have associated the same distance function. The restrictions are: the Voronoi regions must be connected, the degree of a Voronoi vertex must be lower than four, and the bisectors can not contain two-dimensional pieces. The approach is based in a quadtree-based codification, the Voronoi-Quadtree. The Voronoi-Quadtree (VQ) is an extension of the classical quadtree such that from the information encoded in its terminal nodes, a polygonal approximation of the associated Voronoi region boundaries can be computed at different levels of detail. Moreover, by decreasing the resolution of the VQ, the polygonal approximation tends to the exact Voronoi diagram. The most attractive features of this VQ approach are its generality, robustness and easy implementation.

We also present two related algorithms. The first algorithm constructs, with the desired degree of accuracy, the VQ associated to a Voronoi diagram defined by a set of sites and a given distance function. The second algorithm computes the approximated Voronoi region boundaries, represented by a Doubly-Connected Edge List (DCEL), encoded in the nodes of the VQ.

2 Basic definitions

We represent a site S by $S =< G_S, D_S, P_S, L_S >$, where:

• G_S represents the *geometry* of the site S. For example, if S is a circle then G_S is composed by the center and radius of S, if S is a segment then G_S is composed by the endpoints of S;

• D_S is the function that gives the distance from any point p to S. This function depends on the geometry G_S and the distance function considered (Euclidean, convex, ...). For example, if S is the circle of center C and radius r and we consider the Euclidean distance, then $D_S(p) = \parallel p - C \parallel -r$; if S is again the circle of center C and radius r but we consider the convex distance function determined by S, then $D_S(p) = \parallel p - C \parallel /r$;

• P_S is a point such that $D_S(P_S) = 0$. We say that P_S is a *base* point of S. As we will see later, we are going to use P_S to classify the site S with respect to a given rectangular region;

• L_S is a label used to identify the site S.

Let $\mathcal{S} = \{S_1, \cdots, S_n\}$ be a set of sites. Each site $S_i \in \mathcal{S}$ has associated a Voronoi region $VR(S_i) = \{p | d_{S_i}(p) \leq d_{S_j}(p) \,\text{for all}\, j \neq i\}$. The bisector of two different sites $S_i, S_j \in \mathcal{S}$ is defined as $B(S_i, S_j) = \{p | d_{S_i}(p) = d_{S_j}(p)\}$. The boundary of the region $VR(S_i)$ consists of pieces of bisectors $B(S_i, S_j)$ where $i \neq j$. The generalized Voronoi diagram of $\mathcal{S}$, denoted $VD(\mathcal{S})$, is defined as the decomposition of the plane into Voronoi regions [1].

Let R be a rectangular region and S_i a site such that $R \cap VR(S_i) \neq \emptyset$. We say that the S_i is *inside* R when $P_{S_i} \in R \cap VR(S_i)$ and we say that S_i is *outside* R otherwise. A rectangular region R is called a *basic* region when R intersects less than four Voronoi regions.

3 The Voronoi-Quadtree

A quadtree [5, 6] is a tree that encodes the recursive subdivision of a rectangular region in the plane. The root of the tree represents the region. This rectangular region is subdivided into four identical rectangular regions, called quadrants. Each quadrant is represented by one of the four nodes descendants of the root. If the information of a quadrant can not be represented in an exact way, it is labeled as a grey node. Each grey node is subdivided into another four identical rectangular regions which are represented as descendant nodes of the node representing the quadrant in question. This process is repeated recursively until quadrants contain data that can be represented exactly (named *terminal nodes*) or quadrants have a minimum edge length called resolution (*minimal resolution nodes* or *maximal subdivision nodes*).

3.1 Main idea

Let R be a rectangular region containing the sites of $\mathcal{S}$. To obtain a quadtree representation of the part of the Voronoi diagram $VD(\mathcal{S})$ contained in R we use a recursive process that, starting with R, represented in the root node of the quadtree, subdivides each rectangular region in four new identical rectangular regions, denoted $R0$, $R1$, $R2$ and $R3$ and represented in the descendants nodes of R. This subdivision process is repeated until the actual region is a basic region or its resolution is the minimal. This subdivision process is composed of the following steps:

• *Labeling.* At each vertex of the $R0$, $R1$, $R2$ and $R3$ regions of the actual rectangular region R we assign the label of the nearest site belonging to the set of sites classified as inside or outside with respect to R;

• *Inside sites Identification.* At each son node $R0$, $R1$, $R2$ and $R3$ we attach the sites classified as inside with respect to them;

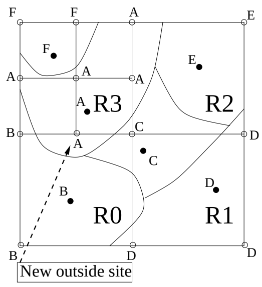


Figure 1: *Outside propagation.*

• *Propagation.* For each vertex v of $R0$, $R1$, $R2$ and $R3$ consider the set $\mathcal{N}$ of nodes that represents rectangular regions whose boundary contains v. Assume that vertex v has been labeled with label L_S in the labeling step. For each node $N \in \mathcal{N}$, if the site S is not an inside site of N then attach the site S to N as an outside site. As it is illustrated in Figure 1 this step might involve rectangles at different subdivision levels. Observe that the son node $R0$ has the site of label A as outside site after the subdivision of the son node $R3$.

Note that to perform the labeling step only inside and outside sites with respect to R are considered. Therefore, an incorrect vertex label assignation might appear. However, the propagation step guarantees that increasing the resolution level we are able to obtain a solution that tends to the exact one. In this way during the VQ construction process the actual VQ is used as an approximation of the Voronoi diagram. This is the main difference with respect to [8] and [9].

3.2 Terminal nodes characterization

Let N denote a node of VQ. Each N is associated to a rectangular region. To identify terminal nodes, for each node we maintain:

• The labels of the vertices of the rectangular region represented by N. Each vertex label corresponds to the (unique) site of $\mathcal{S}$ closest to the vertex. We denote $V(N)$ the set of these four labels attached to N;

• The set of labels of the sites of $\mathcal{S}$ that are classified as inside with respect to the rectangular region represented by N. Let $INsites(N)$ denote this set of labels;

• The set of labels of the sites of $\mathcal{S}$ that are classified as outside with respect to the rectangular region represented by N. Let $OUTsites(N)$ denote this set of labels.

Taking into account all these information, we consider N as a *terminal node* when the Voronoi region boundaries contained in the rectangular region represented by N may be visualized by just using the information encoded in it.

If we define $L(N)$ as the total number of different labels in $V(N)$, $INsites(N)$ and $OUTsites(N)$, then the possible terminal configurations can be characterized by the distribution of the labels of $V(N)$ and the number $L(N)$. Concretely, the four possible terminal patterns, represented in Figure 2, are:

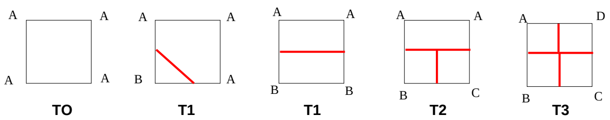


Figure 2: *Terminal node patterns of a VQ.*

• PATTERN $T0$. Obtained when $L(N) = 1$ (i.e. all the vertices of the rectangular region represented by N have the same label). In this case N is contained in a unique Voronoi region. Therefore no Voronoi region boundary intersects the edges of the rectangular region represented by N;

• PATTERN $T1$. Obtained when $L(N) = 2$. The rectangular region represented by N is intersected by two Voronoi regions. The boundary of these regions intersects two edges of the region;

• PATTERN $T2$. Obtained when $L(N) = 3$ and the rectangular region represented by N is intersected by three Voronoi regions. In this case the region is intersected by three Voronoi region boundaries;

• PATTERN $T3$. Given when $L(N) \geq 4$ and the node N is of Maximal resolution. In this case the rectangular region is intersected by four or more Voronoi region boundaries.

4 The VQ construction algorithm

In this section we are going to present, first, the algorithm that has to be applied to construct a VQ. Then, we are going to describe how this algorithm can be extended to obtain approximations of the Voronoi diagram at a determined degree of accuracy, which in the extreme situation tend to the exact solution.

4.1 The basic algorithm

The basic construction algorithm of the VQ requires as input $\mathcal{S}$. The algorithm is based on a top-down strategy. It starts assigning to the root of the quadtree a rectangular region containing $\mathcal{S}$ and then computing the information to be encoded in this node, i.e $V(N), INsites(N)$ and $OUTsites(N)$. Unless the root is terminal, its corresponding rectangular region is subdivided into four identical rectangular regions, each one represented by one of the four descendants of the root, and the information to be encoded in each of them is computed (see Section 3.1). This subdivision process is repeated recursively until the nodes obtained during the subdivision process are terminal nodes or are nodes of minimal resolution.

To carry out this subdivision process a priority queue, Q, is used. The queue Q maintains a pointer to non-terminal nodes. The priority of a node is the level of the node in the quadtree. In this way, the nodes in Q with upper quadtree level are always the first to be processed.

At the end of the process, for each terminal node of VQ we determine the exact set of Voronoi region boundaries intersecting the rectangular region represented by the node. Note that terminal nodes might be distributed at different levels of the quadtree. See Figure3(a), where terminal nodes have been represented as black circles.

This basic algorithm can be summarized as follows:

```
ALGORITHM ConstructVQ(INPUT:S,OUTPUT:VQ)
  N:=DefineRootNode(set of input_sites);
  INsites:=S;
  OUTsites:=NULL;
  SetVerticeLabels(N);
  if !(Terminal(N)) then Q.In(N,0);
  while !(Q.Empty()) do
   N:=Q.Out();
   if !(Terminal(N)) then
      N.DefineSons(N_sons);
      for i:=1 to 4 do
        SetVerticeLabels(N.son[i]);
        ComputeINsites(N.son[i]);
        ComputeOUTsites(N.son[i]);
        PropagateLabels(N.son[i]);
        if !(Terminal(N.son[i])) then
          Q.In(N.son[i],N.son[i].level);
        end_if
       end_for
    end_if
   end_while
```

where functions: `SetVerticeLabels(N)` computes the label of the vertices of node N;

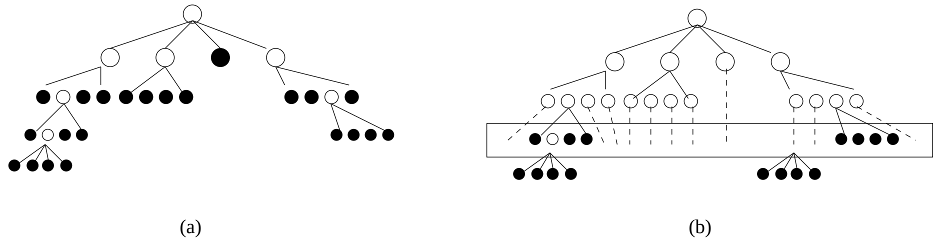


Figure 3: *(a) Distribution of terminal nodes at the end of the basic algorithm. (b) Distribution of terminal nodes when a level of refinament has been introduced.*

`ComputeINsites(N)` computes the inside sites of node N; `ComputeOUTsites(N)` computes the outside sites of node N; `PropagateLabels(N)` performs the propagation step described in section 3.1.

4.2 The improved algorithm

In order to exploit the capabilities of the VQ hierarchical data structure we have extended the basic construction algorithm to obtain approximate Voronoi diagrams at any level of detail. We introduce a new parameter, the level of accuracy, denoted A. This parameter needs also be introduced as input, and it is used to force the subdivision of $T1$, $T2$ and $T3$ terminal VQ nodes whose level is lower than A. In Figure 3(b) the rectangle represents level A. To obtain the Voronoi diagram at accuracy A all terminal nodes obtained during the basic algorithm are subdivided until they reach this level. In the case that a node corresponding to the level A is not a terminal node, the subdivision continues until terminal nodes are reached.

4.3 Computational Cost

To determine the computational cost of the VQ construction algorithm we use results of Integral Geometry [7] relative to the mean number of intersection points between a random rectangular grid and a set of curves. Let n be the number of sites, V the length of the boundary of the Voronoi diagram and L the level of the minimal resolution. It can be proven that the algorithm runs in $o(Ln + VL2^L)$ expected time, and generates $o(V2^L)$ expected number of nodes in the VQ structure.

5 DCEL construction

To represent the polygonal approximation of the Voronoi region boundaries we are going to use the DCEL structure [2]. The method we propose to obtain the DCEL from a VQ is based on a top-down strategy which traverses the VQ and applies the $NodetoDCEL$ function, described in next section, to all the terminal nodes.

5.1 $NodetoDCEL$ function

The $NodetoDCEL$ function computes the part of DCEL contained in a node. Let N be a terminal node of VQ. The vertices of the DCEL contained in N are located on the edges of N with two different labels. When constructing the DCEL the critical point is to determine the position of these vertices in order to guarantee the continuity of the boundary of the Voronoi diagram regions. Our proposed strategy determines this position taking into account the level of N and the level of its terminal neighbor nodes. Concretely: if the neighbor node is in the same or in an upper level, we select the midpoint as the vertex position; and if the neighbor node is in a lower level, we select the midpoint of the edge of this node as the vertex position. Therefore, in this second case, first we detect the level of the neighbor and then we set the vertex point to match with the vertex computed for it (i.e. a higher level vertex is forced to match with the lower one). If the midpoint of the edge was selected as the vertex position, independently of the level of the node, then the final diagram could contain discontinuities. This undesirable situation has been illustrated in Figure 4).

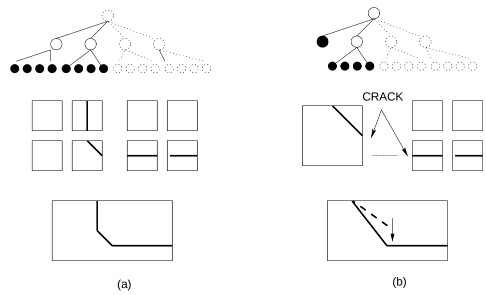
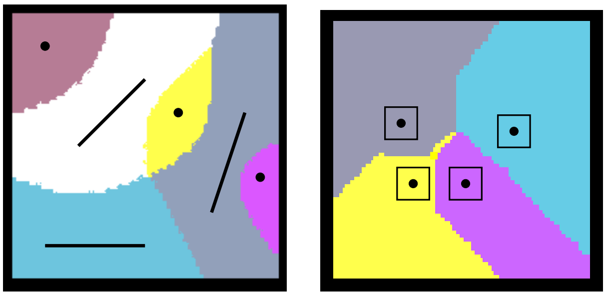


Figure 4: *Black circles identify terminal nodes of the quadtree. (a) When all the nodes are at the same level the edge/segment intersection point coincides with the midpoint of the edge. The Voronoi region boundaries are continuous. (b) When terminal nodes correspond to different quadtree levels, the midpoint generates a discontinuity. The intersection point of the upper node is forced to match with the intersection point position of the lower node.*

The input of the $NodetoDCEL$ function are pointers to N and to its neighbors. Neighbors nodes are maintained in a pile P to facilitate its detection. Note that neighbor nodes can be: neighbors of the parent node, sons of the parent node, or sons of the neighbors of the parent node.

5.2 Computational cost

As we maintain the neighbor information of each VQ node, the DCEL construction algorithm runs in time proportional to the number of nodes. The space required to support the DCEL is proportional to the number of terminal nodes of VQ.



(a) (b)

Figure 5: *Voronoi diagram examples obtained using: (a) Euclidean distance; (b) a convex distance function.*

6 Conclusions and future work

The VQ construction algorithm and the Voronoi diagram approximation algorithm have been implemented. Figures 5(a) and 5(b) show the visualization of some of the Voronoi diagrams obtained considering different distances and resolutions.

We are extending the construction algorithm so that the VQ can also be constructed in some special cases: Voronoi vertices of degree greater than three, bisectors that contain two-dimensional pieces, etc.

Among future developments, we consider of high interest to evaluate the capabilities of the VQ to add and to delete sites. Due to the hierarchical nature of the VQ, this data-structure is well-suited to support multiresolution, therefore an other promising line of research is the design of adaptive algorithms to generate the VQ at different chosen levels of detail in selected regions.

Our final aim is to extend our approach to three-dimensional space for approximating generalized Voronoi diagrams, defining a Voronoi-Octree structure and computing the polyhedral approximation of the corresponding Voronoi diagram.

Acknowledgements

This work was supported by DURSI 2001SGR-00296. The first author was supported in part by grants TIC2000-1009 and TIC2001-2226-C02-02. The second and third authors were supported in part by grants MEC-DGES-SEUID PB98-0933, and TIC2001-2392-C03-01.

References

[1] F. Aurenhammer and R. Klein. *Voronoi diagrams. In Handbook of Computational Geometry. J.-R. Sack, J. Urrutia, editors.* Elsevier Science Publishers, 2000.

[2] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. *Computational Geometry. Algorithms and applications.* Springer, 1997.

[3] K. Hoff, T. Culver, J. Keyser, M. Lin, D. Manocha. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. *Proc. SIGGRAPH'99, ACM Press*, pp 277-286, 1999.

[4] D. Lavender, A. Bowyer, J. Davenport, A. Wallis and J. Woodwark. Voronoi diagrams of set-thoretic solid models. *IEEE Comput. Graph. Appl.*, 12(5):69-77, 1992.

[5] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129-147, 1982.

[6] H. Samet. *Applications of Spatial Data Structures.* Addison Wesley, Reading, MA, 1990.

[7] L.A. Santalò. *Geometric Probability.* Society for Industrial and Applied Mathematics, 1976.

[8] M. Teichmann and S. Teller. Polygonal approximation of Voronoi diagrams of a set of triangles in three dimensions. *Technical Report 766*, Laboratory of Computer science, MIT, 1997.

[9] J. Vleugels and M. Overmars. Approximating Generalized Voronoi Diagrams in Any Dimension. *Technical Report UU-CS-1995-14*, Utrech University, 1995.

[10] Voronoi Diagrams page on the Web: Applications. *http://www.voronoi.com/ApplFrames.htm*