Approximations of 3D Generalized Voronoi Diagrams

Imma Boada* Narcís Coll* Narcís Madern* J. Antoni Sellarès*

Abstract

We introduce a new approach to approximate generalized 3D Voronoi diagrams for different site shapes (points, spheres, segments, lines, polyhedra, etc) and different distance functions (Euclidean metrics, convex distance functions, etc). The approach is based on an octree data structure denoted Voronoi-Octree (VO) that encodes the information required to generate a polyhedral approximation of the Voronoi diagram. Since the medial axis of a polyhedron is a subset of a generalized Voronoi diagram the VO can also be used to encode it. Using the information of the VO we generate and visualize a polyhedral approximation of a generalized Voronoi diagram, and we also solve proximity problems that do not require an explicit representation of the Voronoi diagram such as nearest neighbor queries.

1 Introduction

Given a set of primitives, called Voronoi sites, the generalized Voronoi diagram partitions the space into regions, one per site, such that all points in a region have the same closest site according to some given distance function. Many variants of these diagrams can be considered: by taking sites of different shape or nature, associating weights to the sites, changing the underlying metrics, or using individualized distance functions for the sites [1, 2, 9]. Generalized Voronoi diagrams are widely used in many scientific fields such as computer graphics, geometric modelling, shape analysis, robot motion planning or scientific visualization [6].

Computing a generalized 3D Voronoi diagram involves the manipulation of high-degree algebraic surfaces and their intersections. The exact computation of the diagram poses many problems in terms of robustness and CPU time due to the high number of precision calculations that have to be made [9]. Since the exact computation is known to be hard, in most of the applications approximations of the real diagram within a predetermined precision are used.

There are few methods to approximate generalized 3D Voronoi diagrams and most of them are restricted to Euclidian distance. Lavender at al [8] proposed a hierarchical approach to compute an approximate Voronoi diagram of a set of general sites in arbitrary dimension. They represent the sites by an octree and the cells of the approximate diagram are obtained by considering the distance to the sites. The continuity of the boundary of the approximate diagram is not guaranteed by any theoretical study. Vleugels et al. [13] also proposed a hierarchical approach restricted to convex sites that adaptively subdivides the space into regular cells and computes the Voronoi diagram up to a given precision. Teichmann et al. [12] proposed a technique restricted to triangle sites that subdivides the space into tetrahedral cells. Sites are inserted into a standard octree and a polygonal approximation of the Voronoi diagram is computed by a wavefront propagation strategy. None of these mentioned works gives properties about the convergence to the exact diagram and the computational cost. Some recent papers, driven by applications and using different approaches, investigate the approximate and the exact practical computation of the medial axis of a polyhedron [3, 4, 5].

2 Our Approach

We aim to define a general approach to obtain polyhedral approximations of generalized Voronoi diagrams that simultaneously support approximate nearest neighbor queries in an efficient way. To reach our objective we consider that the octree is the most suitable data structure since it allows to encode information at different levels of detail. A straightforward way to built this octree is by subdividing recursively the region containing all the Voronoi sites and storing at each vertex of the node the information of its nearest site. Despite the simplicity of this strategy, it is costly to implement since it requires an evaluation of all sites against all sites at each step of the octree construction process. On the other hand, since we know that rarely a Voronoi region is adjacent to all the other Voronoi regions, it has no sense to take into account all the sites when computing the nearest site of a vertex node. To avoid processing all the sites at each step of the process we have designed a propagation strategy that detects when new information of the sites is known and also where it has to be stored to guarantee the correctness of the final codification. Although this strategy forces us to restrict our proposal to diagrams with connected Voronoi regions the cost of the method is reduced considerably.

*Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, {imma,coll,nmadern,sellares}@ima.udg.es. Partially supported by grant TIN2004-08065-C02-02

The new main idea of our approach is the combination of the subdivision and the propagation processes. Such a combination also provides an efficient way of dynamically maintaining the VO under the insertion or deletion of sites.

In the next sections we describe the VO construction and its dynamic maintenance.

2.1 Basic Definitions

The set of input sites is denoted as $\mathcal{S} = \{s_1, \cdots, s_n\}$ and R is the cubic region where the approximation of the generalized Voronoi diagram must be computed. Each site s is represented by a tuple $s =< G_s, D_s, P_s >$, where G_s defines the geometry of the site s, D_s is the function that gives the distance from any point p to s and P_s, called the base point of s, is a point such that $D_s(P_s) = 0$ and $P_s \in R$. Each site $s_i \in \mathcal{S}$ has an associated Voronoi region $VR(s_i)$. The generalized Voronoi diagram of $\mathcal{S}$, denoted $\mathcal{V}$, is defined as the partition of the plane induced by the Voronoi regions.

Let N be a node of a octree and s a site. We say that s is an *I-site* with respect to N when $P_s \in N$, a *V-site* with respect to N when a vertex v of N satisfies $v \in VR(s)$ and a *F-site* with respect to N when it is not a V-site and a face f of N satisfies $f \cap VR(s) \neq \emptyset$. The *sites of* N are the set of sites that are I-site, V-site or F-site with respect to N. A node N is a terminal node if it is completely contained in a Voronoi region, i.e. the total number of V-sites, I-sites and F-sites is one.

2.2 The VO Construction Algorithm.

The VO construction algorithm is based on a breadth first strategy using a priority queue sorted by the level of the node in the octree. This allows us to introduce and update the information of the sites stored in the VO when it is required in order to guarantee the completeness and correctness of the encoded information. Let R be a cubical region containing $\mathcal{S}$, Q the priority queue and L_M the maximal subdivision level of the VO fixed by the user. The pseudo-code of the algorithm is reported below (see Algorithm 1). In the pseudo-code the following primitives are used:

INITIALIZE(R,$\mathcal{S}$). Creates the root node from the vertices of R and assigns all the sites as I-sites. It computes the V-sites of the root node (i.e. the nearest I-site of each root vertex) and initializes Q with the root node.

UPDATE(N). Updates the V-site of each vertex of a node N with the nearest of its sites. The distance from the vertex v to a site s is computed using $D_s(v)$.

LEAF(N,L_M). Checks if node N is a leaf or not.

SUBDIVIDE(N). Creates the eight son nodes of a node N, properly distributes the I-sites and F-sites of N to them and computes the V-sites of the sons taking into account the sites of N.

PROPAGATE(N, Q). Checks the coherence between a node N and its adjacent nodes. For each vertex v of N with V-site s_1 locates the set of its adjacent nodes. Let N' be one of these nodes. If v is a vertex of N' with a different V-site s_2 at v, this procedure updates this V-site with s_1 and sends N' to Q. On the contrary, if v is not a vertex of N' (i.e. it is on a face) and s_1 is different to the V-site s_2 of this face, the procedure assigns s_1 to N' as an F-site and sends N' to Q.

COMPACT(N). Checks if all the sons of the node N are terminal. If they are, it prunes the son nodes.

Algorithm 1 OctreeConstruction(R, S, L_M)

```
octree VO;
node N;
queue Q;
VO.INITIALIZE(R, S);
Push(Q,VO.Root());
while No Empty(Q) do
  N= Pop(Q);
  VO.UPDATE(N);
  if VO.LEAF(N, L_M) then
    VO.PROPAGATE(N, Q);
    VO.COMPACT(N.PARENT());
  else
    VO.SUBDIVIDE(N);
    for i = 1 to 8 do
      VO.PROPAGATE(N.SON(i), Q);
      if No VO.LEAF(N.SON(i), L_M) then
        Push(Q, N.SON(i));
      end if
    end for
  end if
end while
```

2.3 Polyhedral Approximation of the Voronoi diagram

To generate a polyhedral approximation of the Voronoi diagram we use an strategy based on the cuberille iso-surface extraction algorithm proposed by Herman and Liu [7] and enhanced and optimized by many others. Our algorithm proceeds in two steps. First, we select leaf nodes intersected by the boundaries of the Voronoi diagram. Nodes whose corner values are all inside the same Voronoi region are ruled out since no boundaries are contained within and thus they have no effect on the final approximation. Second, for each selected node we approximate the boundaries of the Voronoi diagram contained in it. To perform this approximation each selected node is subdivided in $2 \times 2 \times 2$ cells, one for each vertex of the node. For each one of these vertices we evaluate the

three edges incident to it. We assume that an edge is intersected by a boundary of the Voronoi diagram when it has different sites in its extremes. According to the number of intersected edges incident to the vertex we approximate the boundary contained in the node by none, one, two or three of the internal faces of the cell assigned to such vertex.

Figure 1 shows some of the results obtained with the proposed approach.

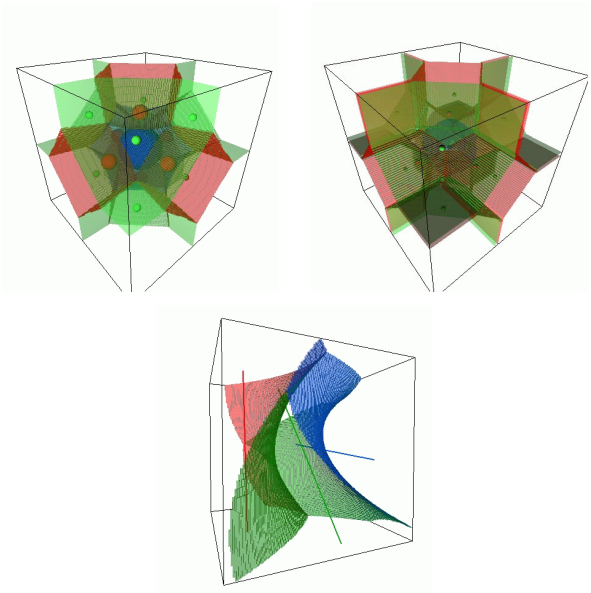

Figure 1: Voronoi diagrams of different scenes obtained with our method: spheres with Euclidean distance, points with L_1 distance and lines with Euclidean distance.

2.4 Computational Cost

To evaluate the VO construction algorithm some considerations have to be taken into account: (i) The mean number of intersection points between a surface of area A and a cubic grid of lines of size u (distance between two consecutive parallel lines of the grid) is $\frac{3A}{2u^2}$ [11]. (ii) If n is the number of sites, for each level we distribute the n sites to some nodes as I-sites. (iii) For each node we need to locate its neighbor nodes. This can be done in L_M worst time, but the expected time for locating neighbors is constant [10]. (iv) The proposed algorithm subdivides nodes that contain a piece of a bisector. We are going to assume that there exist a level L_T such that if $L_M > L_T$ all the sites are a V-site of some node and all the pieces of the bisectors are contained in the Voronoi diagram $\mathcal{V}$. (v) Let $l \leq L_M$ be a level of the octree, V the boundary of the Voronoi diagram and $V(l)$ the boundary of the Voronoi diagram of the sites that are V-site in some node of level l. Let $UV(l) = \bigcup_{i=1}^{l} V(i)$. From the last consideration we have that if $l > L_T$ then the piece of the bisector contained in a node of level l is contained in V, otherwise is contained in $UV(l)$.

Let A be the area of V, $A(l)$ the area of $UV(l)$, $N(L_M)$ the mean number of nodes generated by the construction algorithm of a VO of a maximum level L_M and $T(L_M)$ the mean running time of this algorithm. According to the previous considerations we obtain the following results.

If $L_M \leq L_T$ then $N(L_M) = O\left(\frac{4^{L_M+1}}{a^2}A(L_M)\right)$ and $T(L_M) = O\left(nL_M + \sum_{l=1}^{L_M}\frac{4^l}{a^2}A(l)\right)$. Otherwise, if $L_M > L_T$ then $N(L_M) = O\left(\frac{4^{L_M+1}}{a^2}A\right)$ and $T(L_M) = O\left(nL_M + \sum_{l=1}^{L_T}\frac{4^l}{a^2}A(l) + \frac{4^{L_M+1}-4^{L_T+1}}{a^2}A\right)$.

2.5 Dynamic Maintenance

The potential of the proposed VO has led us to consider the dynamic maintenance of this data structure as an essential operation. Given a VO of a Voronoi diagram approximation and a site s to be inserted the insertion algorithm applies an expansion process that goes from the leaf node containing the base point P_s to all the nodes containing a piece of the boundary of its associated Voronoi region. The algorithm proceeds as follows. First, it applies a top-down VO traversal to identify the leaf node N that contains P_s, enters s as an I-site of N and initializes with N a queue Q used to maintain the nodes to be processed. Then, Q is processed as in the VO construction process (see Algorithm 1). Given a VO of a Voronoi diagram approximation and a site s to be deleted, the deletion algorithm applies a contraction process. It starts with a node containing part of the boundary of the Voronoi region of s, and processes all the nodes containing part of this region. The deletion algorithm performs a top-down VO traversal to identify the leaf node N that contains P_s and erases s as I-site of N. Starting from N, it traverses the nodes that have s as a V-site until a node N' that contains part of the boundary of $VR(s)$ is reached. Next, it initializes with N' a queue Q which is processed as in the VO construction process but applying a new update procedure: a V-site is updated using the sites of the node and the sites of all adjacent nodes and excluding s (see Figure 2). The cost of both algorithms is proportional to the nodes intersected by the Voronoi region associated to the site to be inserted or deleted.

3 Medial Axis

To codify the medial axis in a VO we modify the VO construction algorithm in order to avoid the subdivision of the exterior nodes of the polyhedron and the nodes containing pieces of bisectors between adjacent

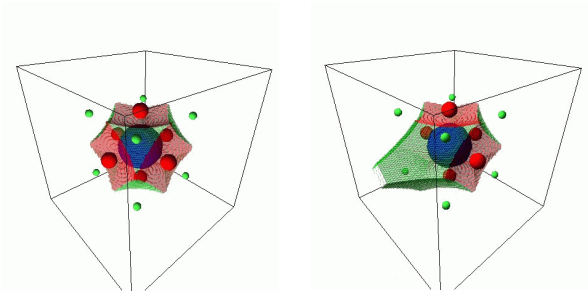

Figure 2: Results obtained by applying to scene of Figure 1(a) the deletion algorithm. Only the Voronoi region of the central sphere before and after the deletion is visualized.

sites. Such a modification implies the definition of two new terminal node criteria.

The first new criterion aims to avoid the subdivision of the exterior nodes. We associate a unitary vector n_s to each site s as follows. If s is a face then n_s is the normal vector to the face oriented outwards the polyhedron, if s is an edge then n_s is the sum of the vectors associated to its adjacent faces and if s is a point then n_s is the sum of the vectors associated to its adjacent faces. Let v be a vertex of a node and s its V-site with base point p_s. The vertex v is exterior to the polyhedron if $(v - p_s) \cdot n_s > 0$. A node N is a terminal node if all its vertices are exterior and any I-site is contained in it. The second new terminal criterion avoids the subdivision of the nodes containing pieces of bisectors between adjacent sites. A node N is a terminal node if all its edges have two V-sites adjacent in the polyhedron.

In Figure 3 we show the medial axis of a polyhedron. Note that what we obtain is a simplified medial axis [5].

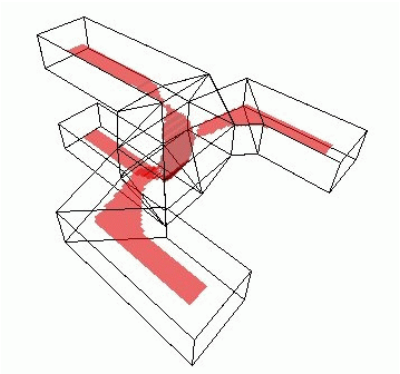

Figure 3: Polyhedron and its medial axis.

4 Nearest Neighbor Queries

The large variety of Generalized Voronoi diagrams that can be encoded by a VO and its hierarchical nature allows us to solve the nearest neighbor problem in many cases. Let $\mathcal{S} = \{s_1, \cdots, s_n\}$ be a set of sites and R a region containing $\mathcal{S}$. Given a query point $q \in R$, we want to find the site s such that $D_s(q) \leq D_{s'}(q)$ for all $s' \in \mathcal{S}; s' \neq s$. To approximately solve the problem it suffices to construct the VO of $\mathcal{S}$ in R, next determine in $O(L_M)$ time the VO leaf node containing the point q and select the nearest site between the set of V-sites of the node.

In a similar way we can solve other kind of queries such as the set of nearest sites of a given site and the closest pair of sites.

References

[1] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. In *ACM Computer Surveys* 23(3), pages 686–695, 1991.

[2] F. Aurenhammer and R. Klein. Voronoi diagrams. In J.R. Sack and J. Urrutia (Eds.), *Handbook of Computational Geometry*, pages 201–290, Elsevier, 2000.

[3] T. Culver, J. Keyser and D. Manocha. Exact Computation of the Medial Axis of a Polyhedron. In *CAGD* 21(1), pages 65–98, 2004.

[4] M. Etzion and A. Rappaport. Computing Voronoi skeletons of a 3-D polyhedron by space subdivision. In *Computational Geometry* 21, pages 87–120, 2002.

[5] M. Foskey, M. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *Journal of Computing and Information Science in Engineering* 3, pages 274-284, 2003.

[6] C. Gold. *The Voronoi Web Site*, http://www.voronoi.com/.

[7] G.T. Herman and K.H.. Liu. Three Dimensional Display of human Organs from Computed Tomograms. In *Computer Graphics and Image Processing* 9(1), pages 1–21, 1979.

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

[9] A. Okabe, B. Boots, K. Sugihara and S.N. Chiu. *Spatial Tessellations: Concepts and Application of Voronoi Diagrams*. John Wiley & Sons, 2000.

[10] H. Samet. *Applications of Spatial Data Structures: computer graphics, image processing, and GIS*. Addison-Wesley, 1993.

[11] K. Sandau. How to estimate the area of a surface using the spatial grid. In *Acta Stereologica* 6/III, pages 31–36, 1987.

[12] 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.

[13] J. Vleugels and M. Overmars. Approximating Generalized Voronoi Diagrams in Any Dimension. In *Int. J. Computational Geometry and Applications* 8, pages 201–221, 1998.