Computers & Graphics 82 (2019) 332–342

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on SMI 2019

Edge topology construction of Voronoi diagrams of spheres in non-general position

Xiang Li[a], Adarsh Krishnamurthy[b], Iddo Hanniel[c], Sara McMains[a,*]

[a] University of California, Berkeley, CA 94720, USA
[b] Iowa State University, Ames, IA 50011, USA
[c] Technion, Israel Institute of Technology, Haifa 32000, Israel

ARTICLE INFO

Article history:
Received 19 March 2019
Revised 17 June 2019
Accepted 18 June 2019
Available online 26 June 2019

Keywords:
Voronoi diagram
Spheres
GPU

ABSTRACT

Although 3D Voronoi diagrams and medial axis transforms have numerous applications in biology, robotics, and manufacturing, most researchers use Voronoi diagrams of points instead of the true 3D input geometry, due to issues of robustness and scalability. In this paper, we present a robust sample-based GPU algorithm for calculating the full topology of Voronoi diagrams of non-general position spheres. Prior work demonstrated that the presence, geometry, and combinatorial basis of spheres that contribute to Voronoi vertices can be efficiently computed by shooting rays from each input sphere, mapping ray intersections with the nearest bisector surface to parametric bounding cubes, and analyzing the results. In this paper, we propose an algorithm on this parametric bounding cube to compute Voronoi edges in addition to the vertices. We successfully extract the full topology of the Voronoi diagram, including special cases such as isolated Voronoi edges that do not contain Voronoi vertices, more than three Voronoi edges emanating from a Voronoi vertex, and Voronoi edges that are shared by more than three Voronoi cells. Our GPU implementation efficiently and robustly handles all input, whether in general or non-general position, and finds all Voronoi vertices and edges, modulo the sampling density, including isolated disconnected edges.

1. Introduction and prior work

A Voronoi diagram is a structure that divides space into regions such that points within each region are closer to a specific input object than to any other input objects. The Voronoi diagram and its variation, the medial axis transformation, are a fundamental topic in computational geometry, with a variety of applications in science and engineering.

A special category of Voronoi diagram is the three-dimensional Voronoi diagram for spheres, which is also called the additively weighted Voronoi diagram, or Apollonius diagram. Because the shape of many real-world objects can be naturally represented by spheres, Voronoi diagram for spheres are widely used in many disciplines such as molecular biology, material science, and physical simulations [1,2].

Many important properties of Voronoi diagram for 3D spheres have been studied [3–5]. The most successful approach to construct the Voronoi diagram for 3D spheres is the edge tracing algorithm proposed by Kim et al. [6]. Based on this algorithm, Kim's group analytically constructs the Euclidean Voronoi diagram of 3D spheres [6], defines its dual structure as a Quasi-triangulation [7,8] and applies the two structures to biomolecular structures [9,10]. A similar algorithm by Medvedev et al. [11] is proposed by developing a geometric structure called the Voronoi S-network.

Kim et al.'s edge tracing approach first calculates a Voronoi vertex, finds its corresponding Voronoi edges, and traces the Voronoi edges to discover other vertices, until all the vertices and edges are found. This algorithm is efficient and relatively robust with an assumption of input spheres in general position; it can not handle disconnected Voronoi edges or high-order Voronoi vertices or edges. These can occur when (1) at least five input spheres sharing a Voronoi vertex; (2) the center of at least four spheres lie on the same plane; or (3) at least five spheres are tangent to the same plane.

Manak and Kolingerova [12] introduced a variation on this edge tracing approach. The idea is to arrange all the Voronoi components into a hierarchy [7], and discover disconnected components by exploring the information in the hierarchy. This method extends the applicability of the edge tracing algorithm to find disconnected Voronoi components, but still limited to Voronoi diagrams with no Voronoi vertices shared by more than four cells and no Voronoi edge shared by more than three cells.

* Corresponding author.
E-mail address: mcmains@berkeley.edu (S. McMains).

https://doi.org/10.1016/j.cag.2019.06.007
0097-8493/

Recently, in Hu et al. [13] we proposed an algorithm to calculate the geometry information of Voronoi vertices and Voronoi face sample points. The algorithm is a sample-based approach that calculates sample points on Voronoi faces by taking the lower envelope of the intersections of rays from each base sphere through its corresponding bisectors. It was able to find Voronoi vertices of both general and non-general position (degenerate-case) inputs by searching for patterns of neighboring sample points that indicate the presence of Voronoi vertices and using numerical iteration to calculate the vertex locations. Unlike just sampling the input geometry, this approach to sampling supports finding actual vertex locations and edge topology. These can be used in molecular analysis to calculate the quasi-triangulation [7], and then calculate properties such as densities and volumes of the proteins [9]. As another example application, we need Voronoi vertex and edge information to serve as nodes and paths to run a path-finding algorithm such as A* to find an efficient collision-free path.

Constructing Voronoi diagrams by computing lower envelopes is a widespread idea in both $\mathbb{R}^2$ and $\mathbb{R}^3$ for different classes of input objects [14–16]. Also, the mathematical representation of bisectors has been studied for different shapes. Hanniel et al. discussed the bisectors and even trisectors among a set of CSG primitives [17]. In Hu et al. we combined the ideas of ray tracing, lower envelope, bisector representation, and numerical iteration, and exploited the parallelism of this approach with an algorithm designed for strengths of the GPU. However, that algorithm only calculates the geometry information of Voronoi vertices; the Voronoi edge topology information is not calculated.

In this paper, we present a follow-on algorithm to calculate Voronoi edges under both general or non-general input positions (including disconnected Voronoi edges and degenerate cases). Combining our output with the prior geometry output, we build the full Voronoi diagram for 3D spheres with inputs under any condition. Our main contributions include:

- GPU framework designed to exploit data parallelism for efficient calculations.
- A robust algorithm to construct the Voronoi edge topology information for both general and non-general position input, including:
  - (a) Self-connected Voronoi edges, as in Fig. 1(a);
  - (b) Infinite Voronoi edges with both ends extending to infinity, as in Fig. 1(b);
  - (c) High order Voronoi vertices or Voronoi edges with non-general position input, as in the example in Fig. 1(c) where the centers of the six equal-sized spheres lie on a hexagon in the same plane.

2. Terminology and definitions

Following Hanniel and Elber [17], the Voronoi diagram for a set of spheres in 3-dimensional space is defined as:

Definition 1. Given a set of spheres $S_0, S_1, \ldots, S_n$ in $\mathbb{R}^3$, the Voronoi cell (VC) of sphere S_i, denoted the "base sphere," is the set of all points closer to S_i than to S_j, $\forall j \neq i$. The Voronoi diagram (VD) is then the union of the Voronoi cells of all (n+1) spheres.

The distance between a point $P = (x, y, z)$ and a sphere S with center (C_x, C_y, C_z) and radius R is defined by the equation:

$$dist(P, S) = \sqrt{(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2} - R. \quad (1)$$

Definition 2. The union of all points that are equidistant from spheres S_i and S_j is called the bisector $B_{i,j}$ of the two spheres. S_i, S_j are called the generating spheres of the bisector $B_{i,j}$.

Any point $P = (x, y, z)$ located on the bisector surface between two spheres S_1 (center $(C_{x_1}, C_{y_1}, C_{z_1})$ and radius R_1) and S_2 (center $(C_{x_2}, C_{y_2}, C_{z_2})$ and radius R_2) satisfies the following equation:

$$\sqrt{(x - C_{x_1})^2 + (y - C_{y_1})^2 + (z - C_{z_1})^2} - R_1 = \sqrt{(x - C_{x_2})^2 + (y - C_{y_2})^2 + (z - C_{z_2})^2} - R_2. \quad (2)$$

The bisector is a plane for two generating spheres with the same radii; for two generating spheres with different radii the bisector is a hyperbolic surface [13,18]. A Voronoi face is the subset of a bisector that is closer to its generating spheres than to any other spheres.

Within a Voronoi cell, Voronoi edges are the intersection between two of its Voronoi faces. In general, Voronoi vertices are the intersection among three Voronoi faces; such a vertex is determined by the four spheres to which it is equidistant (the base sphere and three other spheres corresponding to each of the Voronoi faces).

A base sphere is a generating sphere of a Voronoi face/edge/vertex if such a Voronoi face/edge/vertex appears in the Voronoi cell corresponding to the base sphere. Typically, a Voronoi face/edge/vertex has 2/3/4 generating spheres, respectively. If there are more generating spheres than this general case, they are said to be not in "general position."

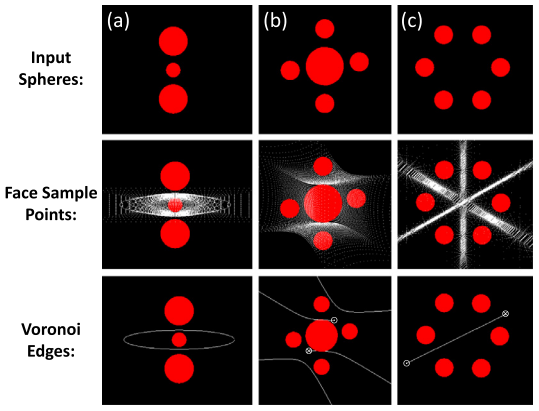

Fig. 1. Example results showing correctly identified Track changes is on 8 Voronoi edge topology for challenging special cases (all figures are 2D rendering of 3D scenes): (a) A self-connected ring-shaped Voronoi edge is identified (the centers of these three input spheres lie on the same line); (b) Four infinite Voronoi edges (both ends extending to infinity) are identified for this case where the centers of the five input spheres lie on the same plane; (c) An infinite Voronoi edge (with both ends extending to infinity) is identified for this case where the centers of six input spheres lie on the same plane. The Voronoi edge passes through the center of the ring of six spheres and is perpendicular to their plane. Symbol "⊙" represents Voronoi edge shooting outwards to infinity (coming towards the reader); symbol "⊗" represents Voronoi edge shooting inwards to infinity (away from the reader).

3. Prior algorithm to calculate Voronoi vertices

We briefly summarize our prior algorithm [13] as follows:

1. Determine the implicit quadratic surface equations, derived from Eq. (2), for the bisectors between the base sphere and all the other input spheres (input spheres can intersect but not completely contain another sphere).
2. From each base sphere, the algorithm creates an axis-aligned bounding cube, and uniformly subdivides each of the six faces to a parameterized domain expressed in variables u and v (Fig. 2). From the center of the base sphere, the algorithm shoots sampling rays through each (u, v) sample point on the bounding-box surface into space.
3. Compute the intersection of each base sphere ray with all the corresponding bisectors, and take the lower envelope of all the intersections (i.e. only keep the nearest intersection

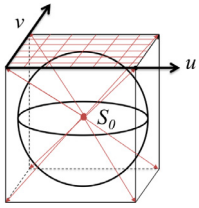

Fig. 2. Mapping sphere to six u-v parametric surfaces on the bounding cube; uniform parametric sampling of top surface shown [13].

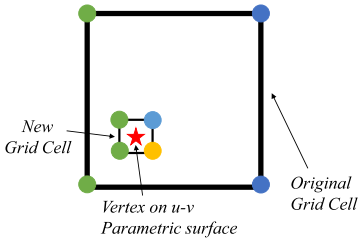

Fig. 4. Construction of the new tiny grid-cell [13].

for each ray) to obtain the sample points on the Voronoi faces of the Voronoi cell for this base sphere.

4. The algorithm color-codes each sample point of the Voronoi cell for the base sphere on the u-v parametric domain based on its corresponding bisector found in step 3. It uses a marching approach to locate the neighborhood of Voronoi vertices by checking each group of four neighboring sample points on the bounding cube, called a "grid-cell." Each 3-color and 4-color grid-cell indicates the appearance of three or more Voronoi faces in this neighborhood. Recall that Voronoi vertices are the intersection among three Voronoi faces in general, so 3-color and 4-color grid-cells indicate the existence of Voronoi vertices.
   Fig. 3 shows the correspondence between sample points in geometric space and the u-v parametric domain.
5. For each 3-color grid-cell, take the average of the location in 3D space of the face sample points as the starting point for iteration, then use the Newton-Raphson method to find the actual vertex location (within a user-defined tolerance) that satisfies the three corresponding implicit bisector equations. If the sampling density is insufficient, special cases of singular Jacobian and 4-color grid-cells would occur, indicating that Voronoi vertices cannot be calculated in those grid-cells. The algorithm uniformly subdivides such grid-cells into four new sub-grid-cells, repeating steps 1–5 for the newly generated u-v points. New sub-cells shoot additional sample rays to the 3D space neighborhood corresponding to the original grid-cell, increasing the local sampling density to provide more information to calculate the Voronoi vertices.
6. The algorithm combines the results of the Voronoi cell calculated for each base sphere to form the full Voronoi diagram. Because each Voronoi vertex has multiple generating spheres (four for general position or more for non-general position), it should be found from all of the Voronoi cells corresponding to the generating spheres. When the sampling density is insufficient, some Voronoi vertices may not be found from all the Voronoi cells. For such "incompletely matched" vertices, from each corresponding base sphere whose Voronoi cell did not find it, the algorithm shoots a new ray from the center of the base sphere to the exact location of this point (the exact location calculated from the other Voronoi cells that found it). The intersection of this ray with the bounding-box surface is the corresponding u-v location of the vertex. Around this vertex's u-v location, the algorithm constructs a much smaller grid-cell (Fig. 4), repeating steps 1–5 for the four newly generated u-v points of the smaller grid-cell. After this targeted sampling around the vertex, the tiny grid-cell will typically have the 3-color patterns corresponding to its generating spheres.

Please refer to [13] for complete details of the algorithm.

4. Updates in the construction of geometry information

In this section, we revisit some implementation details for the prior algorithm summarized above. We turn the process of creating tiny grid-cells of incompletely matched vertices into an iterative searching process (Section 4.1), improving the robustness of the prior algorithm and establishing the relationship between the tiny grid-cell and its containing grid-cell. On the u-v domain, we track the neighboring information between grid-cells by adding pointers in the data structure, and update the information (pointers) during subdivision and the iterative searching process (Section 4.2).

4.1. Iterative search for incompletely matched vertices

In step 6 of the prior algorithm, tiny grid-cells are created to check if the vertex actually exists in the Voronoi cells that did not find it [13]. If the tiny grid-cell is a 3-color or 4-color grid-cell, and its colors are consistent with those in the other cells that initiated the targeted search (Fig. 5(a)), the vertex exists. (Note that for a non-general position Voronoi vertex that has more than four corresponding colors (contributing spheres), if the three or four colors from the tiny grid-cell are a subset of those corresponding colors, it is also considered consistent with other contributing spheres.) If the colors are not consistent, the algorithm will recursively create even smaller grid-cells using the same reduction ratio, until a consistent 3-color or 4-color grid-cell is found or reaching a maximum depth of recursion.

The prior algorithm only calculated the Voronoi vertices' geometry, not their connectivity via Voronoi edges. To calculate the latter, we need to connect each of the four sample points on the tiny grid-cell to each of the corresponding four sample points on its

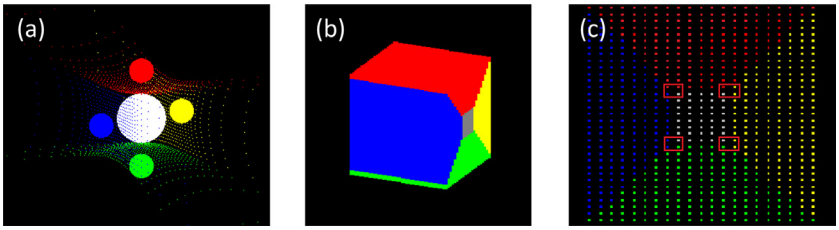

Fig. 3. (a) Sample points on Voronoi faces for white base sphere with four spheres of the same size evenly spaced around it, all five with co-planar centers; (b) corresponding color map of u-v domains on bounding cube with gray representing sample rays that go to infinity; (c) sample point grid on one face of the bounding cube with 3-color grid-cells indicated by boxes [13]. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

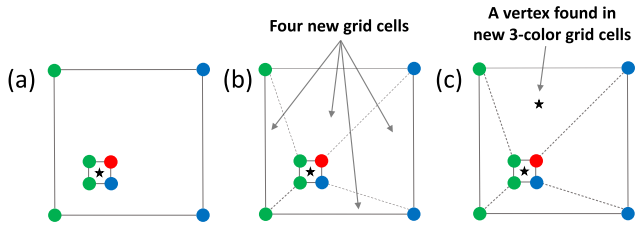

Fig. 5. One iteration in the iterative search process for incompletely matched vertices.

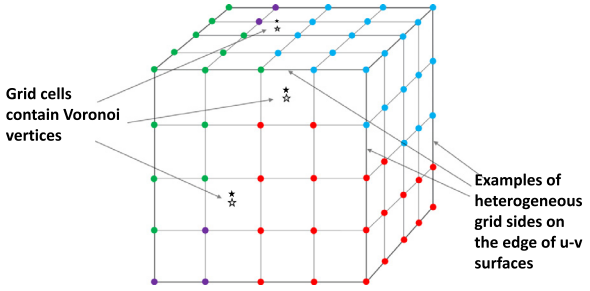

Fig. 6. Color map on a bounding cube (sampling density 5*5).

original containing grid-cell. For example, as shown in Fig. 5(b), the blue sample point on the bottom-right corner of the tiny grid-cell is connected to the blue sample point on the bottom right corner of the original grid-cell.

By this process, we will create four new grid-cells (in addition to the tiny grid-cell). If any of these new grid-cells is 3-color or 4-color, it corresponds to a new vertex that has not been found in this Voronoi cell before (Fig. 5(c)). We calculate the position of this new vertex using numerical iteration and check that it appears in the Voronoi cells all of its other generating spheres (corresponding to the 3 colors of the grid-cell). If it is missing for any of the generating spheres, we repeat this process for this new incompletely matched vertex (in the u-v domain of any generating sphere that does not yet contain it in its Voronoi cell).

We repeat this process for each new incompletely matched vertex, until no more such vertices are found.

4.2. Create neighboring information of grid-cells

Just as in the u-v domain we call each group of four neighboring face sample points a "grid-cell," similarly we call each pair of neighboring face sample points a "grid-side." Each grid-cell has four grid-sides initially (if neighboring cells are subdivided, its sides will also be split whenever a new sample point is introduced in the middle of a side). A grid-side connecting a pair of sample points of the same color is called a homogeneous grid-side; a grid-side connecting a pair of sample points of different colors is called a heterogeneous grid-side. Each grid-side has two neighboring grid-cells that both contain this grid-side. If it is on the edge of the bounding cube, it is shared by two grid-cells on different u-v parametric surfaces (Fig. 6); these grid-cells are still neighbors because of sharing the same grid-side. Having this kind of neighborhood relationship allows traversing between different parametric surfaces on the same bounding cube. In the subdivision process, we may divide an original grid-cell into four sub-grid-cells (e.g. sub-grid-cells (6, 7, 8, and 9) in Fig. 7). We use sub-grid-cell 6 as an example; it shares small grid-sides e_1 and e_2 with original grid-cell 1 and 5, respectively. In such situations, we still describe them as neighboring grid-cells: grid-cells 1 and 6 are neighbors by edge e_1, and grid-cells 5 and 6 are neighbors by edge e_2. Grid-cell

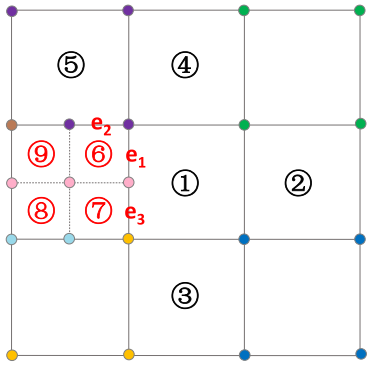

Fig. 7. Neighboring information after subdivision.

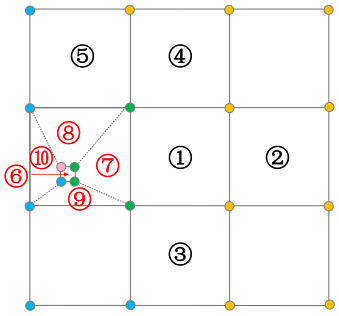

Fig. 8. Neighboring information after targeted search.

1 has five neighboring grid-cells (2, 3, 4, 6, and 7); Sub-grid-cell 6 has four neighboring grid-cells (1, 5, 7, and 9). As shown in Fig. 8, in the case of iterative targeted sampling, we create a tiny grid-cell (6) around the u-v domain location of the vertex. By connecting the four corner points of the tiny grid-cell to the corresponding four corner points of the original grid-cell, grid-cells (7, 8, 9, and 10) are created. Their neighboring information is still determined by shared edges.

5. Construction of edge topology information

In the previous sections, we described how to obtain the exact locations of the Voronoi vertices and their corresponding u-v grid-cells from each of the contributing base spheres. To determine the connectivity among the Voronoi vertices, we detect Voronoi edges by exploring the connectivity of grid-cells on the colored bounding cubes (u-v parametric surfaces).

Within a Voronoi cell, each Voronoi edge is the intersection between two of its Voronoi faces, so neighboring pairs of u-v sample points of different colors (heterogeneous sides of grid-cells) indicate the existence of Voronoi edges in its corresponding neighborhood in 3D space. Each heterogeneous grid-side indicates the presence of a particular Voronoi edge generated by the base sphere and two spheres corresponding to the two differently colored sample points. The pair of colors is called the "edge identifier" of its corresponding Voronoi edge in this Voronoi cell. For a particular Voronoi edge, we look for heterogeneous grid-sides with the same edge identifiers. If a grid-cell has two heterogeneous grid-sides with the same color pair (edge identifier), it indicates this particular edge enters this grid-cell from the neighboring grid-cell sharing one such side, and exits to the neighboring grid-cell sharing the other such side. We call such grid-cells "through-grid-cells" of a particular Voronoi edge because the edge goes through those grid-cells.

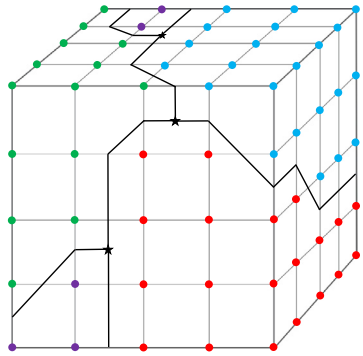

Fig. 9. Voronoi edge topology on bounding cube (sampling density 5*5).

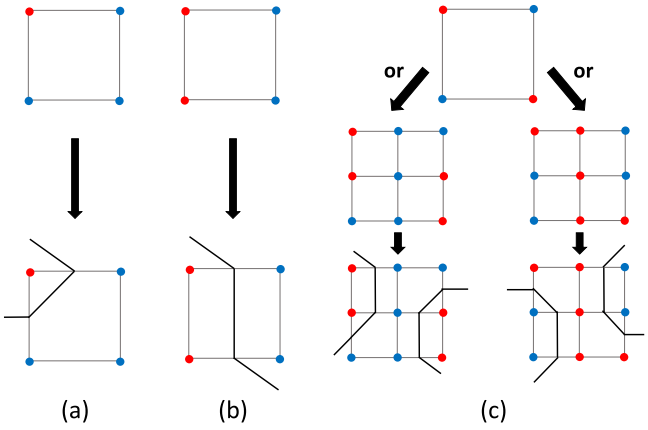

Fig. 10. Four topological configurations and the corresponding 2-color grid-cells. For example (c), the color of the middle sample after subdivision will typically disambiguate the two cases, unless the subdivision gives rise to another case (c), in which case we continue subdividing those sub-grid-cells.

As illustrated in Fig. 9, our algorithm premise is straightforward: tracing the paths of each Voronoi edge by following series of its "through-grid-cells." (Note that henceforth we are using the term "edge tracing" in this sample-space context; it has no relation to the "edge tracing algorithm" of Kim et al.) In a grid-cell containing a Voronoi vertex, each heterogeneous grid-side represents one particular Voronoi edge that exits into the neighboring grid-cell that shares this grid-side. Starting from each grid-cell that contains a Voronoi vertex, we trace the paths of each of its incident Voronoi edges along a sequence of through-grid-cells connected by grid-sides with the edge identifier associated with that Voronoi edge, until reaching another grid-cell also searching with the same edge identifier.

Our algorithm has four stages: preprocessing 2-color grid-cells with particular color patterns (Section 5.1), tracing "through grid-cells" on the bounding cube (Section 5.2), searching for isolated Voronoi edges (Section 5.3), and sorting of the Voronoi edges (Section 5.4).

We now describe the steps in detail.

5.1. Subdivision preprocessing of 2-color grid-cells

There are three possible configurations (and their inverses) of 2-color grid-cells as shown in Fig. 10.

In configurations (a) and (b), there are two heterogeneous grid-sides. When tracing the Voronoi edges, if the edge being traced enters this grid-cell from one of the heterogeneous grid-sides, it will exit on the other heterogeneous side to the next neighboring grid-cell.

When the sampling density is insufficient, we might have configuration (c). Just as for Marching Cubes [19], there is insufficient information to determine the topology inside this grid-cell. We use the same uniform subdivision as in Section 3 step 5 to subdivide this grid-cell into four sub-cells, and get five new colored u-v sample points. If the four sub-grid-cells are all configuration (a) or (b), we can determine the Voronoi edge trajectory inside them. If any of the sub-grid-cell is still in configuration (c), we continue subdividing until no grid-cells have such a configuration, or a maximum depth of recursion is met.

5.2. Tracing Voronoi edges via "through-grid-cells"

After preprocessing all the 2-color grid-cells to configuration (a) and (b), all the grid-cells are ready for our edge tracing process. For each base sphere, the search for Voronoi edges is based on the colors of the u-v sample points on its corresponding bounding cube. Our search starts at each grid-cell containing a Voronoi vertex. Such grid-cells are at least 3-color grid-cells, which contain multiple heterogeneous grid-sides with different edge identifiers (color pairs). Each such color pair indicates a unique Voronoi edge emanating from the Voronoi vertex and exiting to the neighboring grid-cell that shares the grid-side with that edge-identifier (Fig. 11(a)). We trace this edge to this next (neighboring) grid-cell. We check if this new grid-cell is a through-grid-cell for the particular edge we are tracing. If so, we identify the other (exiting) grid-side with the same edge identifier as the grid-side through which we entered the grid-cell, and proceed to the corresponding neighboring grid-cell. We keep tracing the Voronoi edge to its next grid-cell, and repeat the process above (Fig. 11(b)). In each iteration, we check if the new grid-cell is also a "through-grid-cell" of this edge. If so, we proceed to the neighboring grid-cell sharing the exiting grid-side (with the matching edge identifier), and mark this grid-cell as "traced" for this particular edge identifier color pair. In addition to calculating edge geometry sample points in each iteration, we take the average of the 3D space coordinates corresponding to the two sample points of the exiting grid-side, and using that average location as our start point, run Newton-Raphson iteration (similar to Section 3 step 5) to find a point on the Voronoi edge in actual 3D space.

We trace all unique edges from all Voronoi vertices in parallel. Each trace terminates when either of the following conditions is met:

1. The next "through-grid-cell" of the edge is already marked with the same edge identifier, which means it has met up with the search from the other end of the same edge (Fig. 11(c)). In this situation, we record the connectivity between the starting Voronoi vertices corresponding to each path, and combine the 3D sample points we calculated along the two paths, reversing the order of points from one trace. Thus we obtain not only the Voronoi edge topology, but also ordered sample point locations on the edge geometry. These points can be used for visualization or analysis.
2. The next grid-cell is not a "through-grid-cell" of the edge, and has one or more sample points at infinity. The existence of sample points at infinity and the absence of an exiting grid-side with the corresponding edge identifier indicates that the Voronoi edge we are tracing goes to infinity. (Fig. 3 shows an example with sample points at infinity.) In this situation, we terminate the search, and record the topology information and 3D point locations on the edge geometry of this infinite Voronoi edge. This situation is further discussed in Section 5.2.2.

It is also very rarely the case that the next grid-cell is not a "through-grid-cell" of the edge but doesn't go to infinity, in which

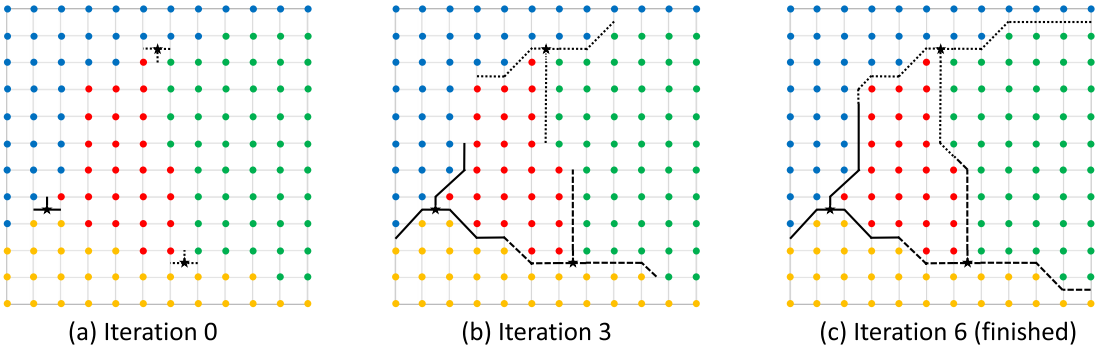

Fig. 11. Topology construction process on a u-v surface (sampling density 12*12); stars indicate the presence of a Voronoi vertex in the grid-cell. Traces from different vertices are shown with different line styles.

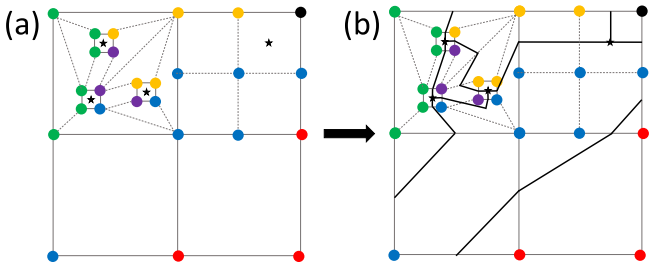

Fig. 12. Edge topology construction process of a non-general u-v parametric face (a) with an original sampling density 3 by 3. A star indicates the presence of a Voronoi vertex in the grid-cell. (b) The resulting edge trace topology from the Voronoi vertices is shown with bold lines.

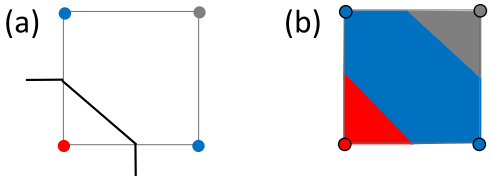

Fig. 13. (a) The tracing path in a grid-cell containing an infinite sample point at infinity and a new grid-side to update. (b) The actual color pattern inside the grid-cell (grey represents infinity). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

case it needs to be subdivided to continue tracing the edge, as discussed in Section 5.2.3.

Finally, we gather all the information from each base sphere. Each Voronoi edge occurs in the Voronoi cell of at least three base spheres (three for general position, more for non-general position). The 3D point locations on the edge geometry will be different for each base sphere's representation of the same Voronoi edge. Because the choice of 3D points do not affect the accuracy of topology construction, we randomly keep one group of 3D point locations for each Voronoi edge.

Some special conditions may bring more complexity into our tracing process. Although the algorithm we described above is able to handle them, some implementation details should be emphasized. We discuss such conditions below.

5.2.1. Non-uniform grid-cells

Because of the generation of sub-grid-cells from the subdividing operation (e.g. Section 3 step 5) and/or from shooting new rays towards matched vertices from base spheres that did not initially find them (Section 4.1), sometimes we do not have uniform grid-cells on the parametric bounding cube. Fig. 12(a) shows an example of non-uniform grid-cells on a parametric face that had an original sampling density of 2 by 2 cells and had subsequent sub-grid-cells added by both of these operations.

In the edge tracing process, just like with uniform grid-cells, we check if the exiting grid-side (with corresponding edge identifier) exists among all the grid-sides in this non-uniform grid-cells. If it does, we continue tracing to the next corresponding grid-cell, otherwise we subdivide this grid-cell and continue the trace through the new generated sub-grid-cells (details in Section 5.2.3). The updating process in Section 4.2 still applies to obtain the neighboring/grid-side information and construct the edge topology on non-uniform grid-cells (Fig. 12(b)).

It is necessary to be aware, as described in Section 4.2, that on these non-uniform bounding cubes, some grid-cells would have more than 4 neighboring grid-cells. When the initial sampling density is too low, more non-uniform grid-cells will be generated. Grid-cells may even have tens of neighbors. Such poor uniformity would damage the efficiency of our parallel algorithm, so choosing an appropriate initial sampling density is important in the implementation. We will discuss this more in Section 6.

5.2.2. Infinite sample points

Voronoi edges do not always terminate at Voronoi vertices. Some Voronoi edges extend to infinity, on one or both ends. We call such Voronoi edges infinite edges.

During our edge tracing process, if the current grid-cell does not have an exiting grid-side with corresponding edge identifier, and has at least one corner sample point at infinity, this Voronoi edge extends to infinity in 3D space. In this situation, we will record that this edge extends to infinity and stop the search.

If a grid-cell contains a sample point at infinity, but still has matched entering and exiting grid-sides, we should continue tracing the Voronoi edge through the grid-cell neighboring at the exiting grid-side (Fig. 13).

5.2.3. Additional subdivision for topological disambiguation

During the edge tracing process, the trace might enter a grid-cell that doesn't have a clearly matched exiting grid-side, yet has no sample points at infinity. This situation happens when the sampling density is insufficient. There are two cases: the grid-cell has (1) no other grid-sides with the corresponding edge identifier; or (2) multiple grid-sides with the corresponding edge identifier.

An example of the first case is shown in Fig. 14(a), where the target Voronoi edge is close to another edge but not intersecting with it. In this example, if we are tracing the upper Voronoi edge from the right to the left, the edge enters grid-cell 1 from its right grid-side *e*. Since there are no other grid-sides with the red-blue edge identifier, grid-cell 1 is not a "through-grid-cell"; we need a greater local sampling density in grid-cell 1 to continue tracking this Voronoi edge.

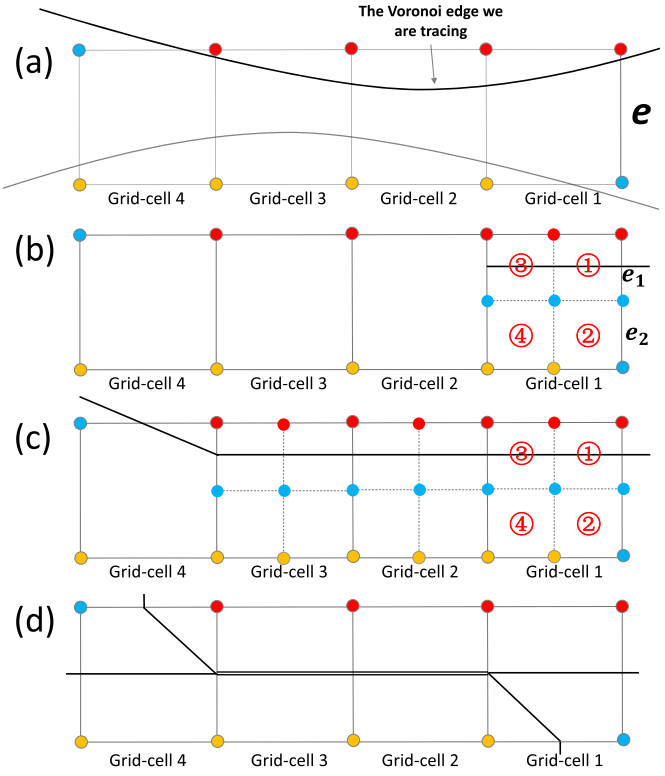

Fig. 14. The subdivision operation on grid-cells that are neither "through-grid-cells" nor contain sample points at infinity.

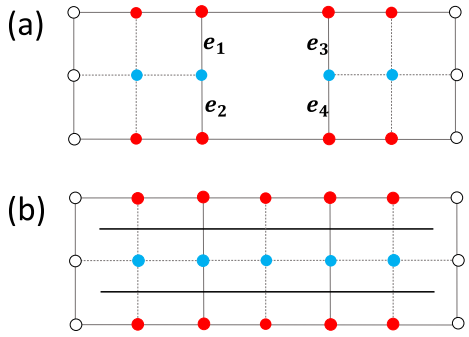

Fig. 15. An example of a grid-cell that has more than two grid-sides with the same red-blue identifier: (a) When the edge tracing enters the middle grid-cell by any of the grid-side e_1, e_2, e_3, or e_4, it will have three other grid-sides with the same (red-blue) edge identifier; (b) the result of the edge tracing process after the middle grid-cell is subdivided. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

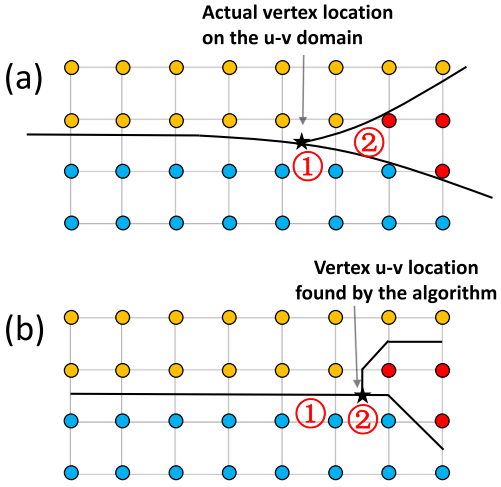

Fig. 16. (a) The actual vertex and edge location in the u-v domain. (b) The calculated vertex and edge location in the u-v domain by our algorithm.

In this situation, we subdivide the current grid-cell, then continue tracing on the appropriate new sub-grid-cells by checking the color identifier over the two new sub-grid-sides generated from the previous entering grid-side. In Fig. 14(b), we subdivide grid-cell 1, then continue tracing on sub-grid-cell (1) at a new entering grid-side e_1. The same situation occurs when the trace enters grid-cell 2 and grid-cell 3; we repeat the subdivision process, iteratively subdividing the two grid-cells when the trace enters each of them, and get the edge path as shown in Fig. 14(c). If necessary, we repeat the subdivision process until all the grid-cells along this trace are "through-grid-cells," or a maximum depth of subdivision is reached.

At the maximum recursion depth, if we still cannot distinguish their respective paths, we treat the edges as coincident in this neighborhood. In Fig. 14(d), assume all the grid-cells are already at max recursion depth. When tracing to grid-cell 1, no other grid-sides match either of the blue-red or blue-yellow edge identifiers corresponding to the two entering traces. Under the local sampling density at the maximum recursion depth, the two edges are still too close to each other, with no sample point of the color (blue) detected on the exiting grid-side (the left grid-side of grid-cell 1). In this case, we treat the two edges as coincident in the neighborhood of grid-cell 1, and trace the exiting yellow-red-(blue) "super edge." We keep tracing this super-edge by the yellow-red edge identifier, until the edge identifiers of both the two individual edges (blue-red and blue-yellow) re-occur (in grid-cell 4). We then continue each of the individual traces of those two edges.

An example of the second case that requires disambiguation, where the grid-cell has multiple possible exiting grid-sides with the same edge identifier, is shown in Fig. 15. We subdivide such grid-cells with the same process as the first case, unless it is a 2-color grid-cell in configuration (c) (Fig. 10(c)), and already met the maximum depth of recursion (Section 5.1). To disambiguate the true layout in that case, we will compare the edge topology result from two possible layouts to the results from the other Voronoi cells that share the edge (Section 5.4).

5.2.4. Case of u-v deviation

Under insufficient sampling density, there might be two or more Voronoi edges entering a grid-cell through different grid-sides and exiting through the same grid-side, without intersecting with each other inside this grid-cell. In such situations, the grid-cell might be 3-color but has no Voronoi vertices located in its corresponding 3D space neighborhood. An example is shown in Fig. 16(a), where two Voronoi edges enter grid-cell 2 through different grid-sides (the top and the right grid-sides) and both exit through the same grid-side (the left grid-side). It is a 3-color grid-cell, but the corresponding vertex geometry is not within its u-v sub-domain. If we were to shoot a ray directly to this Voronoi vertex, it would be in the u-v sub-domain of grid-cell 1. However, because of missing the red color, grid-cell 1 only has two colors, indicating no existence of Voronoi vertices in the corresponding 3D space.

We call this situation a "u-v deviation," because instead of being located in the u-v sub-domain of a ray to the actual Voronoi vertex geometry (grid-cell 1), the vertex location "deviates" to a nearby 3-color grid-cell with the correct color codes corresponding to this Voronoi vertex.

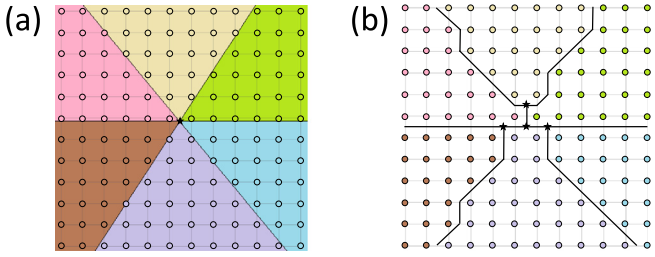

Fig. 17. A high-order Voronoi vertex shared by six Voronoi edges: (a) The actual vertex and edge location on the u-v domain. (b) The calculated vertex and edge location on the u-v domain by our algorithm.

Although we may have such "u-v deviations," the 3D space geometry and topology information for such Voronoi vertices are still obtained correctly. Our 3D space geometry calculation is based on the bisector equations among the generating spheres (Section 3 step 5). With the same base sphere and 3-color pattern, starting from a different grid-cell around the true u-v location only means an offset of the iteration start point; the numerical iteration still finds the correct 3D geometry of the vertex. For the topology, as shown in Fig. 16(b), the Voronoi edges correctly connect to the corresponding 3-color grid-cell, even though the grid-cell containing the actual u-v location of the ray to the vertex is the neighboring grid-cell (1).

In degenerate cases, if a Voronoi vertex is shared by more than three Voronoi edges, multiple corresponding 3-color grid-cells for the vertex locations will typically exist. In Fig. 17(a), a Voronoi vertex is shared by six Voronoi edges. In the u-v domain, there exists four 3-color grid-cells containing vertices (Fig. 17(b)). In the Voronoi vertex sorting process (Section 3 step 6), we will determine that these four vertex locations correspond to the same Voronoi vertex because all the calculated vertices have the same 3D space coordinates.

In our edge tracing process, connecting to any of these four 3-color grid-cells is treated as connecting to this particular Voronoi vertex. The zero-length "Voronoi edges" between Voronoi vertices in 3-color grid-cells corresponding to the same actual Voronoi vertex, allow them to be merged when calculating edge topology. As seen in Fig. 17, "u-v deviation" is what allows us to handle such high-order Voronoi vertices. Under non-general position input, a Voronoi vertex may deviate to multiple corresponding grid-cells on the parametric bounding box, extending the ability of our algorithm to detect the vertex's connectivity with more than four Voronoi edges (even though a non-subdivided grid-cell can at most detect four Voronoi edges through its four grid-sides).

5.3. Detecting isolated Voronoi edges

After the edge tracing process described in Section 5.2, we will have constructed the topology of all the Voronoi edges that connect Voronoi vertices.

However, not all Voronoi edges are connected with Voronoi vertices. Two types of isolated Voronoi edges are disconnected from any of the Voronoi vertices, and we cannot find them from our general edge tracing process. They are:

1. Infinite Voronoi edges with both ends extending to infinity. An example of this situation is shown in Fig. 18, in which five spheres have their center on the same plane. If we look at the white sphere's parametric bounding cube (Fig. 18(c)), there are four Voronoi edges with both of their ends corresponding to grid-cells with infinite sample points.
2. Self-connected Voronoi edges. As shown in Fig. 1(a), this ring-like Voronoi edge does not have any actual endpoints.

In (Section 5.2), starting from each of the Voronoi vertices, we marked all the "through-grid-cells" with corresponding edge identifiers along each tracing path. After the tracing process, we check if each "through-grid-cell" has been marked for each distinct edge identifier of all its grid-sides. If a "through-grid-cell" is unmarked for any of its edge identifiers, this "through-grid-cell" is related to an isolated Voronoi edge corresponding to the unmarked edge identifier.

We collect all such unmarked through-grid-cells. First we sort them into different groups by their unmarked edge identifiers. If an unmarked "through-grid-cell" has multiple edge identifiers, each edge identifier represents a particular isolated edge; it will be put in all the corresponding groups. For each group, we randomly choose one of its grid-cells as our starting grid-cell, and start tracing the paths of the edge through the two grid-sides corresponding to the two edge identifier colors of this through-grid-cell. In each trace, we repeat the same iteration process as for edge tracing (Section 4.1). For the search for each isolated edge, if the two traces both stop at grid-cells indicating infinity (Section 5.2.2), the Voronoi edge is an infinite Voronoi edge with both ends extending to infinity. If the two traces meet each other, the Voronoi edge is a self-connected edge.

5.4. Sorting of the Voronoi edges

After calculating all the Voronoi edges (including Voronoi edges connected with vertices or isolated Voronoi edges) from each Voronoi cell (base sphere), we combine the edge information of each individual Voronoi cell to form the whole Voronoi diagram by sorting and merging the Voronoi edges detected for each base sphere's Voronoi cell.

A Voronoi edge has at least three contributing spheres (three for general position and four or more for non-general position), so it will have at least three "edge uses" in the Voronoi cells for those spheres. Each Voronoi edge use has its own 2-color edge identifier, and one color code corresponding to the base sphere of its Voronoi cell. We call this unique Voronoi cell related color code the "cell identifier" of this Voronoi edge use. The color triplet, indicates the three contributing input spheres of the corresponding Voronoi edge.

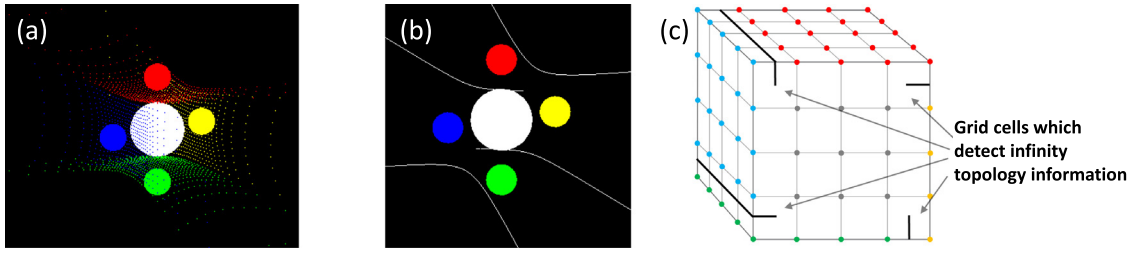

Fig. 18. An example of inputs generating infinite isolated Voronoi edges (grey represents infinity). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

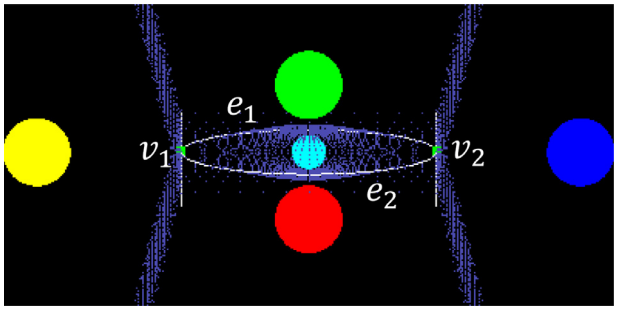

Fig. 19. An example of two Voronoi edges (e_1 and e_2) sharing the same three contributing spheres (the green, cyan, and red spheres) and two Voronoi vertices (v_1 and v_2) they connect. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

We sort all the Voronoi edge uses by these corresponding color triplets and the Voronoi vertices they connect (typically two, but special cases of self-connected Voronoi edges with zero or one Voronoi vertices may exist). For each sorting group with the same color triplet and corresponding Voronoi vertices, if there are three Voronoi edge uses in the group and each of them has a different cell identifier, the particular Voronoi edge corresponding to this group exists between the corresponding Voronoi vertices (or infinities), and can be found in all of its corresponding Voronoi cells. A rare case occurs when two Voronoi edges exist between the same corresponding pair of Voronoi vertices, and both of them were found in the same corresponding Voronoi cells (Fig. 19). In this case, six Voronoi edge uses are in one sorting group, and correspond to two distinct Voronoi edges.

By this sorting process, we are able to find all the general-position Voronoi edges, self-connected Voronoi edges, and infinite Voronoi edges in the Voronoi diagram. However, "high-order" Voronoi edges shared by more than three cells will remain unmatched, requiring a second round of sorting.

For any of the Voronoi edge uses not satisfying the conditions in the first sort, we pick one Voronoi edge use and its color triplet as the first member in its group. Starting from this triplet, we iteratively search the remaining unmatched Voronoi edge uses for those where two of the three colors in the triplet match any member triplet colors in the group and have the same corresponding Voronoi vertices to all the members, adding the corresponding Voronoi edge use as a new member in this group if so. After the search for the first group, we repeat this process for any remaining un-grouped Voronoi edge uses, until all of them are grouped. In each group, if the number of different colors equals to the number of Voronoi edge uses, and each of them has a different cell identifier, the particular high-order Voronoi edge corresponding to this group exists between the corresponding Voronoi vertices (or infinities), and can be found in all of its corresponding Voronoi cells.

After this second round of sorting, the only unmatched Voronoi edge uses should correspond to ambiguous grid-cells at maximum subdivision depth (Section 5.2.3). Such grid-cells had two possibilities, only one of them is a true Voronoi edge. For such a pair of Voronoi edge uses corresponding to the same ambiguous grid-cell, if only one of them matches with unmatched Voronoi edge uses from other Voronoi cells and can be grouped with them in a consistent Voronoi edge, we are done and stop considering the other possibility. If neither can be grouped consistently, additional subdivision is needed to disambiguate the corresponding grid-cell.

After these two rounds of sorting, all the Voronoi edge uses should be grouped into corresponding Voronoi edges in the Voronoi diagram; otherwise we restart the algorithm, increasing the initial sampling density and the maximum recursion depth. Similarly, during the edge-tracing process, if the maximum depth of recursion is met in any step (Section 4.1, 5.1, or 5.2.3), we restart the algorithm with a higher initial sampling density and maximum recursion depth.

6. GPU framework

Most steps of our algorithm are implemented on the GPU (Fig. 20) using CUDA programming to exploit data parallelism. For example, in the edge detecting step, for all paths we trace (starting from all the vertices on all u-v bounding cubes), we use the same operation that iteratively finds neighboring grid-cells sharing the same heterogeneous grid-side. In CUDA, each unit of data is processed on one GPU thread. The method/function being executed by all the GPU threads in parallel is called a kernel. Table 1 summarizes the kernel and thread information of each step performed on the GPU.

7. Results

Our algorithm to compute the whole Voronoi diagram (geometry and topology) was run on a PC with an Intel® Core™ Processor i7-9700K CPU with 16GB RAM and an NVIDIA GeForce GTX 1080 Ti graphics card. To test our ability to handle large real-world inputs, we implemented our algorithm on protein structures from the protein data bank [20], where protein molecule structures are represented as combinations of atom spheres with different radii.

In the implementation of our algorithm, the selection of appropriate sampling density (the number of u-v samples on each face of bounding cubes) is important for obtaining good parallelism. If the sampling density is too low, we will not obtain enough information to find Voronoi vertices, and then in the calculating vertices step, we will need to subdivide many grid-cells and reconstruct their neighboring information. Subdivisions will damage the uniformity of the grid-cells, reducing data parallelism, and decreasing the efficiency of our algorithm. On the other hand, if the sampling density is too high, we obtain too many sample points that

Table 1
Thread and kernel information for all steps performed on the GPU. Colors correspond to the timing breakdown of geometry/topology in Fig. 21.

Step	Per Thread	Kernel
Calculate Bisectors	Each input sphere	Calculates all the bisectors between each sphere and all other input spheres (Section 3 Step 1)
Sample Rays	Each ray	Samples the rays from all input spheres (Section 3 Step 2)
Take Lower Envelopes	Each ray	Calculates the intersections between each ray and all bisectors, keeping the intersection point with minimum ray distance (Section 3 Step 3)
Calculate Vertices	Each grid-cell	Finds the grid-cells containing Voronoi vertices and calculate the vertices by numerical iteration (Section 3 Step 4 and 5)
Preprocess 2-color Grid-cells	Each grid-cell	Checks if the grid-cell is 2-color and in configuration (c), and subdivides such grid-cells to configuration (a) and (b) (Section 5.1 Fig. 10)
Detect Edges	Each trace	Starting from each grid-cell containing a Voronoi vertex, for each edge identifier in such grid-cells, the kernel traces the corresponding Voronoi edge via "through-grid-cells" (Section 5.2)
Detect Isolated Edges	Each trace	Starting from a random member of each group of unmarked "through-grid-cells" with the same edge identifier, the kernel traces the corresponding isolated Voronoi edge (Sections 5.2 & 5.3)

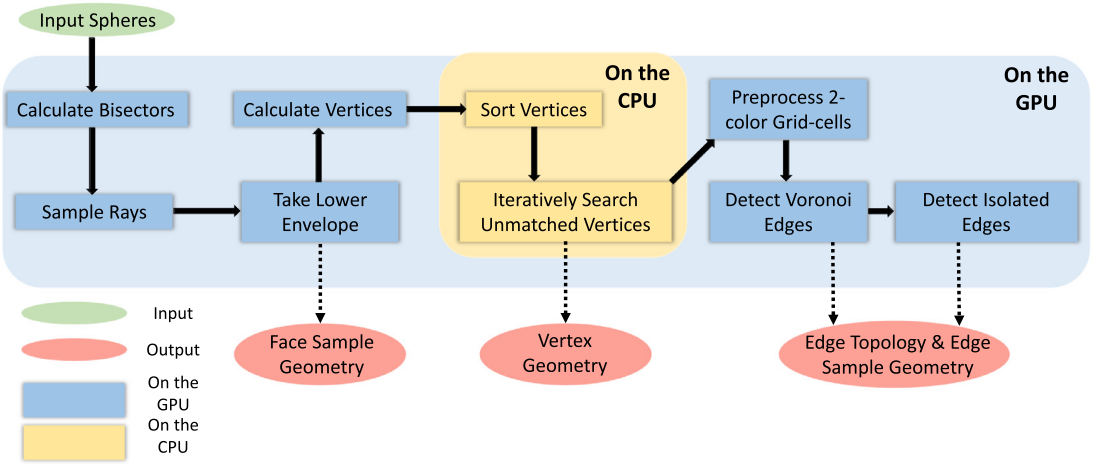

Fig. 20. The GPU Framework.

Table 2
Number of subdivisions and deepest level of subdivision with different sampling densities, for protein 1crn-PDB with 327 input atoms. Total number of grid-cells includes original grid-cells and sub-grid-cells generated by subdivision and targeted search.

Sampling Density	# of Subdivision Operations	Total # of Grid-cells	Deepest Level of Subdivision
1*1	4237	15,046	8*th*
10*10	405	197,945	5*th*
20*20	195	785,660	4*th*
40*40	81	3,139,584	3*rd*
80*80	23	12,556,922	2*nd*
160*160	3	50,227,212	1*st*
320*320	0	200,908,800	N/A

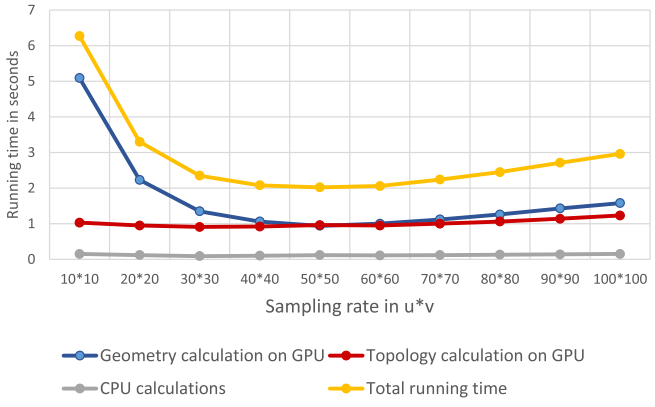

Fig. 21. Running time vs. sampling rate, protein 1crn-PDB with 327 atoms.

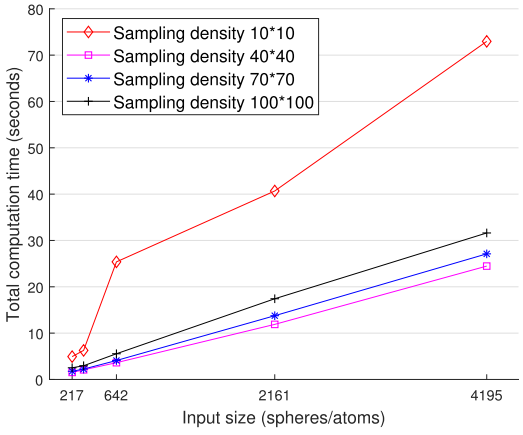

Fig. 22. Computation time at different sampling densities on protein models: (1) "1al1-PQR" with 217 atoms; (2) "1crn-PDB" with 327 atoms; (3) "1crn-PQR" with 642 atoms; (4) "1bh8-PQR" with 2161 atoms; and (5) "1JD0-PDB" with 4195 atoms.

are unnecessary for finding Voronoi vertices and edges (such as sample points on 1-color grid-cells). To illustrate this trade off, we ran our algorithm on protein "1crn-PDB" with 327 input spheres and different sampling densities. Table 2 shows how the total number of grid-cells increases along with the increase of sampling density, but the number of subdivision operations decreases. Total number of grid-cells includes original grid-cells and sub-grid-cells generated by subdivision and targeted search. Fig. 21 shows how the running time varies with sampling density for the same input. Among all the proteins tested, sampling density in the range 40*40 to 50*50 provides the lowest total running time. When the sampling density is lower than 40*40, the steps up to and including calculating vertices (steps of geometry calculation on GPU) take an extremely long time because of the lack of parallelism in the subdivision operation. When the sampling density is higher than 50*50, the parallelism of our algorithm is excellent but the increasing number of unnecessary sample points hurts the running time.

On other protein models we tested containing 217–4195 spheres, the running times also indicated that sampling densities from 40*40 to 50*50 were the most efficient (lowest overall running time).

The total computation time of our algorithm under different input sizes is shown in Fig. 22. When the sampling density is lower (e.g. 10*10 here), the computational efficiency is inherently dependent on the geometric distribution of the input spheres, which determines the data parallelism (number of subdivision operations) of our algorithm. When the sampling density obviates most subdivision (usually more than 40*40), the computation time increases roughly linearly with the number of input atoms (spheres).

To test our ability to handle non-general position inputs, we designed example inputs in three non-general situations: self-connected Voronoi edges, infinite Voronoi edges with both ends extending to infinity, and high order Voronoi vertices or edges. Our algorithm successfully handled all the non-general situations; one result from each of situation is shown in Fig. 1.

For the verification of our experimental results, we implemented a naive brute-force algorithm for detecting all Voronoi vertices, to serve as ground truth. The algorithm is based on the fact that a Voronoi vertex always corresponds to a sphere that is tangent to all of its (four or more) contributing spheres, and does not intersect or contain any other input sphere.

Therefore, for each combination of four spheres from the input, we calculate their tangent sphere using the algorithm described by Gavrilova and Rokne [21], and check if the resulting tangent sphere intersects (tangent not included) or contains any of the other input spheres. If not, the center of this tangent sphere is a Voronoi vertex. We exhaustively make this check for all the four-sphere combinations and gather the information of all the calculated Voronoi vertices.

In our experiments, all of the results produced by our algorithm matched the results from this brute-force algorithm. Furthermore, in all cases, including large-size inputs from the protein data bank (the five shown in Fig. 22 and ten other randomly selected proteins) and the non-general position cases, the Voronoi edge calculation is consistent among all the Voronoi cells (meaning there are no unmatched Voronoi edge uses after two rounds of sorting). Futhermore, the same output Voronoi diagram vertices and edge topology is produced for all sample densities that we tested, even with the coarsest possible initial 1×1 sampling (just the 8 corners of each bounding cube). In the tests, we set the maximum subdivision depth to 10, and this bound was never met (Table 2); in other words, less than 10 levels of subdivision was fine enough to trace all the Voronoi edges.

8. Conclusion

We have presented an algorithm to construct edge topology for Voronoi diagrams of spheres in $\mathbb{R}^3$. It successfully handles input spheres in both general and non-general position, including self-connected Voronoi edges, infinite Voronoi edges, and high-order position inputs. We design a GPU framework to exploit data parallelism, and find the approximate range of sampling densities to maximize the efficiency of our algorithm. Under sufficient sampling densities, the total calculation time of the Voronoi diagram for large inputs is roughly in a linear relationship with the number of the input spheres.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We gratefully acknowledge support from National Science Foundation grant 1331352, Zhongyin Hu for the development of the codebase described in our prior paper, and suggestions from the anonymous reviewers.

References

[1] Richards FM. The interpretation of protein structures: total volume, group volume distributions and packing density. J Mol Biol 1974;82(1):1–14.
[2] Voloshin V, Beaufils S, Medvedev N. Void space analysis of the structure of liquids. J Mol Liq 2002;96:101–12.
[3] Gavrilova M. Proximity and applications in general metrics. University of Calgary; 1998.
[4] Will H-M. Computation of additively weighted Voronoi cells for applications in molecular biology. ETH Zurich; 1999.
[5] Boissonnat J-D, Karavelas MI. On the combinatorial complexity of Euclidean Voronoi cells and convex hulls of d-dimensional spheres. In: Proceedings of the SODA'03. SIAM; 2003. p. 305–12.
[6] Kim D-S, Cho Y, Kim D. Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. Comput-Aided Des 2005;37(13):1412–24.
[7] Kim D-S, Kim D, Cho Y, Sugihara K. Quasi-triangulation and interworld data structure in three dimensions. Comput-Aided Des 2006;38(7):808–19.
[8] Kim D-S, Cho Y, Sugihara K. Quasi-worlds and quasi-operators on quasi-triangulations. Comput-Aided Des 2010;42(10):874–88.
[9] Kim D-S, Ryu J, Shin H, Cho Y. Beta-decomposition for the volume and area of the union of three-dimensional balls and their offsets. J Comput Chem 2012;33(13):1252–73.
[10] Kim D-S, Kim C-M, Won C-I, Kim J-K, Ryu J, Cho Y, et al. BetaDock: shape-priority docking method based on beta-complex. J Biomol Struct Dyn 2011;29(1):219–42.
[11] Medvedev NN, Voloshin V, Luchnikov V, Gavrilova ML. An algorithm for three--dimensional Voronoi S-network. J Comput Chem 2006;27(14):1676–92.
[12] Manak M, Kolingerova I. Extension of the edge tracing algorithm to disconnected Voronoi skeletons. Inf Process Lett 2016;116(2):85–92.
[13] Hu Z, Li X, Krishnamurthy A, Hanniel I, McMains S. Voronoi cells of non-general position spheres using the GPU. Comput-Aided Des Appl 2017;14(5):572–81.
[14] Hanniel I, Muthuganapathy R, Elber G, Kim M-S. Precise Voronoi cell extraction of free-form rational planar closed curves. In: Proceedings of the 2005 ACM symposium on solid and physical modeling. ACM; 2005. p. 51–9.
[15] Seong J-K, Cohen E, Elber G. Voronoi diagram computations for planar NURBS curves. In: Proceedings of the 2008 ACM symposium on solid and physical modeling. ACM; 2008. p. 67–77.
[16] Hoff III KE, Keyser J, Lin M, Manocha D, Culver T. Fast computation of generalized Voronoi diagrams using graphics hardware. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques; 1999. p. 277–86.
[17] Hanniel I, Elber G. Computing the Voronoi cells of planes, spheres and cylinders in R^3. CAGD 2009;26(6):695–710.
[18] Elber G, Kim M-S. Computing rational bisectors. IEEE Comput Gr Appl 1999;19(6):76–81.
[19] Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. In: Proceedings of the ACM SIGGRAPH, 21. ACM; 1987. p. 163–9.
[20] The RCSB Protein Data Bank. http://www.rcsb.org/pdb/home/home.do; Accessed: 2015-12-14.
[21] Gavrilova ML, Rokne J. Updating the topology of the dynamic Voronoi diagram for spheres in Euclidean d-dimensional space. CAGD 2003;20(4):231–42.