BULL! - The Molecular Geometry Engine Based on Voronoi Diagram, Quasi-Triangulation, and Beta-Complex

Deok-Soo Kim[1,2,⋆], Youngsong Cho[2], Jae-Kwan Kim[2], Joonghyun Ryu[2], Mokwon Lee[1], Jehyun Cha[1], and Chanyoung Song[1]

[1] Department of Mechanical Engineering, Hanyang University, Seoul, Korea
dskim@hanyang.ac.kr
http://voronoi.hanyang.ac.kr
[2] Voronoi Diagram Research Center, Hanyang University, Seoul, Korea
{ycho,jkkim,jhryu,mwlee,jhcha,cysong}@voronoi.hanyang.ac.kr

Abstract. Libraries are available for the power diagram and the ordinary Voronoi diagram of points upon which application programs can be easily built. However, its counterpart for the Voronoi diagram of spheres does not exist despite of enormous applications, particularly those in molecular worlds. In this paper, we present the BULL! library which abbreviates "Beta Universe Library Liberandam!" for computing the Voronoi diagram of spheres, transforming it to the quasi-triangulation, and extracting the beta-complex. Being an engine library implemented in the standard C++, application programmers can simply call API-functions of BULL! to build application programs correctly, efficiently, and easily. The BULL! engine is designed so that the application programs developed by embedding API-functions are completely independent of the future modifications of the engine.

Keywords: application program interface, engine, molecular structure, computational geometry, geometric modeling, C++.

1 Introduction

Molecular structure determines molecular function. While the meaning of "structure" varies in molecular worlds, "geometry" is always central to molecular structure and there exist rich prior studies on the geometry of molecules. However, the studies were mostly conducted in ad hoc manner depending on discipline or even depending on a particular aspect of a problem at hand; There have been no unified framework of theory to deal with the geometry of molecular structure.

Authors' group at the Voronoi Diagram Research Center (VDRC) [1], Hanyang University, has been developing the **Molecular Geometry (MG)** theory during the past decade based on the Voronoi diagram, in particular the Voronoi diagram of spheres, and its derivative structures [6,11,4,8,7,10]. Suppose that P

⋆ Corresponding author.

H. Hong and C. Yap (Eds.): ICMS 2014, LNCS 8592, pp. 206–213, 2014.

is a molecular structure problem of interest at hand. Let S(P) be its solution to be found. In the MG paradigm, the problem P is first transformed to a corresponding geometry problem G of three-dimensional spheres whose solution S(G) can be easily found via a geometry engine. Then, S(G) is inverse-transformed to get $S^{-1}(G)$ which is expected to be close to S(P). Assuming a geometric engine for transforming G to S(G), $S^{-1}(G)$ can converge to S(P) with a sufficient number of iterations if the forward- and backward-transformations are well-defined.

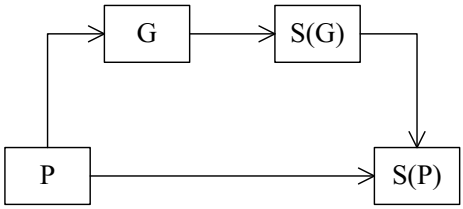

Fig. 1. The **Molecular Geometry** paradigm for solving all geometry problems in molecular worlds

BULL! is the engine for the MG paradigm based on the Voronoi diagram of spherical atoms and its two derivative structures: the quasi-triangulation and the beta-complex. This paper is the initial proposal of the BULL! library which abbreviates "Beta Universe Library Liberandam!" meaning that the library liberates researchers who are working on molecular structure from the hard and tedious job of developing accurate and efficient geometric algorithms and their implementation. The BULL! library is implemented in the standard C++ language and will be freely available from VDRC (http://voronoi.hanyang.ac.kr).

2 Three Fundamental Constructs in BULL!

There are three fundamental computational constructs in the MG theory: the primal, the dual, and the interested subset of the dual which correspond to the Voronoi diagram, the quasi-triangulation, and the beta-complex in BULL!, respectively. Any, perhaps all, geometry problems in molecular worlds can be correctly, efficiently, and easily solved using either one of these constructs or their combinations.

Among various types of Voronoi diagram, the Voronoi diagram of spheres is the key construct. The Voronoi diagram of spheres is the generalization of the power diagram [2] which is a generalization of the ordinary Voronoi diagram of points from the Euclidean distance point of view [3,13]. By the same token, in the dual space, the quasi-triangulation [9,7] is the generalization of the regular triangulation and the Delaunay triangulation and the beta-complex [8] is the generalization of the (weighted) alpha-complex [5]. Note that the Voronoi

diagram of spheres is also called the additively-weighted Voronoi diagram. The quasi-, regular, and the Delaunay triangulations are the dual of the Voronoi diagram of spheres, the power diagram, and the ordinary Voronoi diagram of points, respectively.

The Voronoi diagram $\mathcal{VD}$ of three-dimensional spheres can be computed by the edge-tracing algorithm taking $O(n^3)$ time in the worst case but $O(n)$ time on average for molecules [6]. The quasi-triangulation $\mathcal{QT}$ is obtained by transforming $\mathcal{VD}$ in $O(n)$ time in the worst case. Then, the beta-complex is extracted from $\mathcal{QT}$ using a binary search in $O(n \log n + k)$ time in the worst case where k is the number of simplexes in the resulting beta-complex. More powerful approach to general queries on the quasi-triangulation is available [10]. Fig. 2 summarizes the process from the Voronoi diagram to the quasi-triangulation to the beta-complex, given an input of the arrangement of atoms.

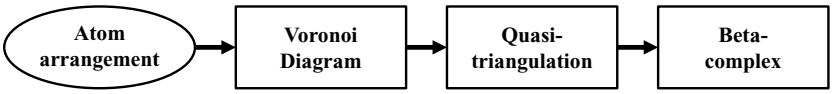

Fig. 2. An atom arrangement is given as an input. The Voronoi diagram is first computed and is transformed to the quasi-triangulation which is then used to extract the beta-complex.

Fig. 3 shows an example of this process in the plane. Fig. 3(a), (b), and (c) show the Voronoi diagram of circular disks, the quasi-triangulation, and the beta-complexes corresponding to the probe of a certain radius. The beta-complex defines the neighborhood information between atom pairs within the boundary of the disk set defined by the probe whereas the quasi-triangulation defines that for all disks. The figures are all created using the `BetaConcept` program [12] freely available at VDRC.

3 Data Structures in BULL!

Data structure is one of the key issues in BULL!. *Radial-edge data structure* (REDS) stores the topology of the Voronoi diagram because Voronoi diagrams have a cell structure. Fig. 4 shows the schematic diagram of the REDS used in BULL!. In the following, "V-" denotes "Voronoi." Each V-cell of the Voronoi diagram has a direct pointer to each of its V-faces and thus has $|f|$ V-face pointers where $|f|$ is the number of V-faces of the V-cell. Each V-face has two pointers to the incident V-cells. Each V-face has a pointer to each one of its bounding loops and each loop points to the V-face that it lies on. A V-face has one or more loops where the first one is external and the others are for interior holes. Thus a V-face has $|l|$ V-loop pointers. Each loop has a pointer to the V-face that it belongs to. Each loop points to one of the partial edges that belongs to the loop

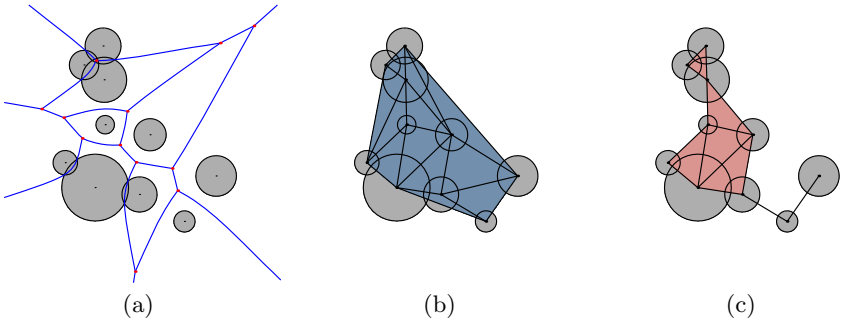

Fig. 3. (a) The Voronoi diagram of disks, (b) the quasi-triangulation, and (c) the beta-complexes corresponding to a circular probe. Figures created using the `BetaConcept` program [12].

and each partial edge points to the loop that it belongs to. Each partial edge has two more types of pointers for the two types of cycles on a V-face: one for the *radial cycle* which consists of a single pointer; the other for the *loop cycle* which consists of two pointers for its predecessor and successor. Each V-edge has two pointers to its V-vertices and each V-vertex has four pointers where each points to a V-edge incident to it.

V-cell, V-face, V-edge, and V-vertex may be associated with corresponding geometry. Each V-cell has the coordinates and radius of the spherical atom generator corresponding to itself and each V-vertex has its coordinate data. Each V-face and V-edge may or may not have its surface and curve equation, respectively, depending on application. If the geometry part of V-face and V-edge is not explicitly stored, they can be easily computed if the topology of the Voronoi diagram is available.

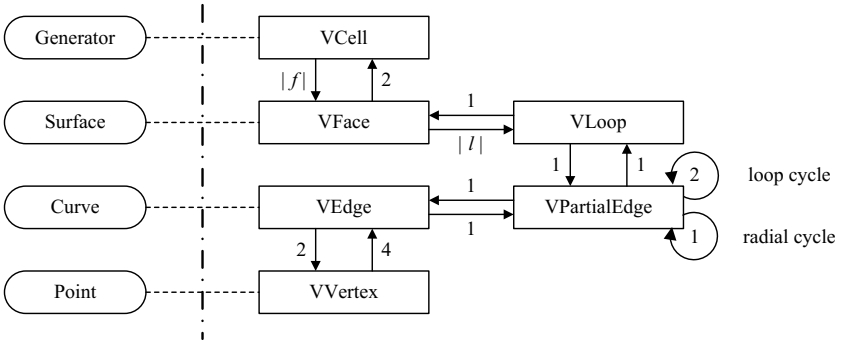

Fig. 4. Radial edge data structure (REDS) for Voronoi diagram of spherical atoms

The quasi-triangulation $\mathcal{QT}$ is stored in the *Inter-world data structure* [9] (IWDS) which is schematically shown in Fig. 5(a). Each QT-cell has four pointers to each of its QT-faces and each QT-face has two pointers to the incident QT-cells. Each QT-cell has also four pointers to each of its QT-vertices and each QT-vertex has a pointer to the incident QT-cells. Each QT-face has three pointers to each one of its QT-edges and each QT-edge has $|w|$ pointers to the incident QT-faces. Each QT-edge has two pointers to its QT-vertices and each QT-vertex has a pointer to one of the incident QT-edges. If there are a fixed number m of incident QT-simplexes to a QT-simplex σ, σ has m pointers. Otherwise (i.e., if there are an arbitrary number of QT-simplexes incident to a QT-simplex σ), σ has only one pointer to one of the incident QT-simplexes from which all the other QT-simplexes can be traversed. The number $|w|$ is to connect small-worlds. The explicit representation of QT-faces and QT-edges are necessary for the extraction of beta-complexes from the quasi-triangulation. Otherwise, if it is not necessary to store QT-faces and QT-edges explicitly, then a more compact data structure can be devised as shown in Fig. 5(b) which is now we call a compact IWDS, abbreviated cIWDS. Note that these two data structures were called differently in our earlier papers.

It is important for us to state the following: BULL! uses REDS to store all the three types of Voronoi diagrams in 3D: The Voronoi diagram of spheres, the power diagram, and the ordinary Voronoi diagram of points. This is possible because REDS can store the most general one: the Voronoi diagram of spheres. By the same token, BULL! uses IWDS to store all the three types of triangulations in a compact form: the quasi-, the regular, and Delaunay triangulations. Note that this observation is critically used in the design of the classes in BULL!.

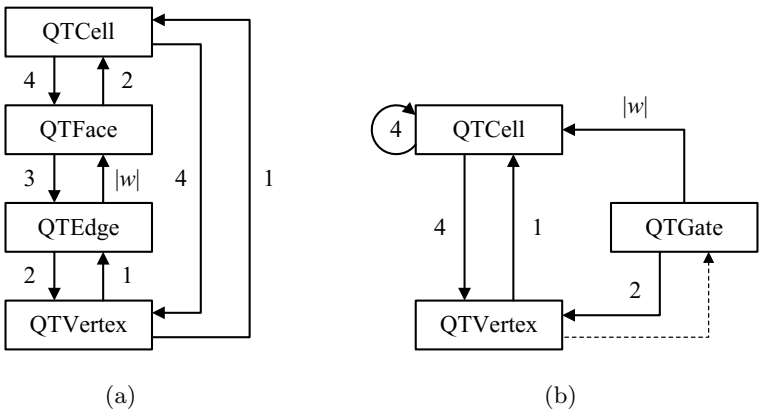

Fig. 5. Data structure for the quasi-triangulation. (a) Inter-world data structure (IWDS) and (b) compact inter-world data structure (cIWDS).

4 Architecture of BULL! and an Example Application Program

The architecture of BULL! is designed so that programmers can create application programs easily and conveniently through out entire software life cycle. To achieve this goal, we have designed BULL! so that application program is completely separated from the internal functions which may be modified as the development of BULL! goes on.

BULL! has a three-tier architecture as shown in Fig. 6: **API-tier**, **Core-tier**, and **Geometry-tier**. The API-tier is only visible to and the other two are completely hidden from application programmers. Thus, application program interacts only with the API-tier by including the related head files and embedding the API-functions. The API-tier interacts with Core-tier which implements the application neutral data structure of the primal and dual structures. The Core-tier interacts with the Geometry-tier which actually contains the Voronoi diagram construction codes, possibly implementing more than one algorithms for the Voronoi diagram construction. Currently, the Geometry-tier contains the implementations of the edge-tracing algorithm and the region-expansion algorithm for three-dimensional spheres. In this architecture, the modifications to be made in the Geometry-tier in future does not cause any change in the codes of already-existing application programs.

In principle, the API-tier currently contains three main classes: `AtomSetVoronoiDiagram`, `QuasiTriangulation`, and `BetaComplex`. Each contains API-functions that can be embedded in application programs to perform various computations. There are their respective counterparts in the Core-tier: `BallSetVoronoiDiagramCore`, `QuasiTriangulationCore`, and `BetaComplexCore`. Both the transformation between the Voronoi diagram and the quasi-triangulation and the extraction of the beta-complexes from the quasi-triangulation are in fact all performed in the Core-tier. Each of the API-tier classes communicates its counterpart in the Core-tier. In the Geometry-tier, there are currently two classes for the Voronoi diagram of spheres: `SphereSetVoronoiDiagramByEdgeTracing` and `SphereSetVoronoiDiagramByRegionExpansion` where both communicate with `BallSetVoronoiDiagramCore`. Note that we try to exclusively use the words "atom," "ball," and "sphere" for API-, Core-, and Geometry-tier, respectively.

Fig. 7 illustrates an example of simple application program which calls a few API-functions of BULL!. Assume that necessary header files are included. Given

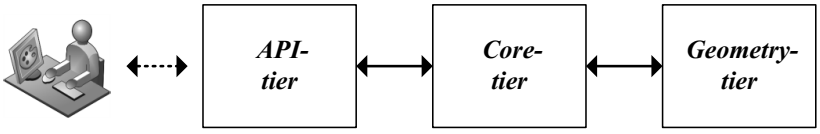

Fig. 6. 3-tier architecture of BULL!

a set of 100 input disks (ie, atoms) generated by the line 2, the simple API-functions in the lines 3 and 4 creates the Voronoi diagram `atomVD` and stores in `atomlist`. The lines 5 and 6 retrieves all the Voronoi cells in the Voronoi diagram. The lines 7 through 11 counts the number of adjacent Voronoi cells of each Voronoi cell in the Voronoi diagram and accumulates these numbers for all Voronoi cells in the Voronoi diagram. Then, the lines 12 and 13 prints out the average number of adjacent Voronoi cells in the entire Voronoi diagram. The lines 14 through 17 perform similar operation for the quasi-triangulation which is transformed from the Voronoi diagram in the line 15. The lines 18 through 22 perform similar operation for the beta-complex which is extracted from the quasi-triangulation by the line 20. As this code shows, the creation of such an application program is very straightforward once a user understands the basics of the theory and learns a few API-functions which are necessary to build an application program. Application programs performing more complicated tasks can be similarly created by using API-functions of BULL!.

```
1   int main()
    {
2       list<Sphere3d> atomlist = generateRandomAtoms(100);

3       AtomSetVoronoiDiagram atomVD;
4       atomVD.compute( atomlist );

5       list<VCellHandle> all_cells;
6       atomVD.getAllVCells( all_cells );

7       int countAdjacentCells = 0;
8       list<VCellHandle>::iterator i_cell;
9       for (i_cell=all_cells.begin(); i_cell!=all_cells.end(); i_cell++ ) {
10          countAdjacentCells += atomVD.countNumberOfAdjacentCells( *i_cell );
11      }
12      cout << "The average number of adjacent cells is ";
13      cout << (double)countAdjacentCells/atomVD.countNumberOfCells() << endl;

14      QuasiTriangulation QT;
15      QT.transform( atomVD );
16      cout << "The number of 2-adjacency anomaly is ";
17      cout << QT.countNumberOf2AdjacencyAnomaly() << endl;

18      double betaValue = 10.0;
19      BetaComplex BC;
20      BC.extract( QT, betaValue );
21      cout << "The number of atoms on the boundary of BC(beta=50) is ";
22      cout << BC.countNumberOfBCVerticesOnBoundary() << endl;

23      return 0;
    };
```

Fig. 7. An example application program

5 Conclusion

The Voronoi diagram of spherical atoms has many important applications, particularly for molecular worlds. While there are libraries for the ordinary Voronoi diagram and power diagram, its counterpart for spheres is not available. In this paper, we present the BULL! engine library for the Voronoi diagram of spheres, the quasi-triangulation, and the beta-complex with which application programmers can easily build application programs dealing with molecules. The BULL! engine is designed so that the application programs developed based on the API-tier is completely independent of future modifications. BULL! will be available from VDRC (http://voronoi.hanyang.ac.kr) at Hanyang University.

Acknowledgement. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2012R1A2A1A05026395).

References

1. Voronoi Diagram Research Center, `http://voronoi.hanyang.ac.kr/`
2. Aurenhammer, F.: Power diagrams: Properties, algorithms and applications. SIAM Journal on Computing 16, 78–96 (1987)
3. Aurenhammer, F.: Voronoi diagrams – a survey of a fundamental geometric data structure. ACM Computing Surveys 23(3), 345–405 (1991)
4. Cho, Y., Kim, D., Kim, D.S.: Topology representation for the Voronoi diagram of 3D spheres. International Journal of CAD/CAM 5(1), 59–68 (2005), `http://www.ijcc.org`
5. Edelsbrunner, H., Mücke, E.P.: Three-dimensional alpha shapes. ACM Transactions on Graphics 13(1), 43–72 (1994)
6. Kim, D.S., Cho, Y., Kim, D.: Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. Computer-Aided Design 37(13), 1412–1424 (2005)
7. Kim, D.S., Cho, Y., Sugihara, K.: Quasi-worlds and quasi-operators on quasi-triangulations. Computer-Aided Design 42(10), 874–888 (2010)
8. Kim, D.S., Cho, Y., Sugihara, K., Ryu, J., Kim, D.: Three-dimensional beta-shapes and beta-complexes via quasi-triangulation. Computer-Aided Design 42(10), 911–929 (2010)
9. Kim, D.S., Kim, D., Cho, Y., Sugihara, K.: Quasi-triangulation and interworld data structure in three dimensions. Computer-Aided Design 38(7), 808–819 (2006)
10. Kim, D.S., Kim, J.K., Cho, Y., Kim, C.M.: Querying simplexes in quasi-triangulation. Computer-Aided Design 44(2), 85–98 (2012)
11. Kim, D., Kim, D.S.: Region-expansion for the Voronoi diagram of 3D spheres. Computer-Aided Design 38(5), 417–430 (2006)
12. Kim, J.K., Cho, Y., Kim, D., Kim, D.S.: Voronoi diagrams, quasi-triangulations, and beta-complexes for disks in $\mathbb{R}^2$: The theory and implementation in BetaConcept. Journal of Computational Design and Engineering 1(2), 78–86 (2014)
13. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd edn. John Wiley & Sons, Chichester (1999)