J Math Model Algor (2009) 8:343–355
DOI 10.1007/s10852-008-9097-6

A Fast Robust Algorithm for Computing Discrete Voronoi Diagrams

Mirko Velić · Dave May · Louis Moresi

Received: 15 October 2008 / Accepted: 21 November 2008 / Published online: 16 December 2008

Abstract We describe an algorithm for the construction of discretized Voronoi diagrams on a CPU within the context of a large scale numerical fluid simulation. The Discrete Voronoi Chain (DVC) algorithm is fast, flexible and robust. The algorithm stores the Voronoi diagram on a grid or lattice that may be structured or unstructured. The Voronoi diagram is computed by alternatively updating two lists of grid cells per particle to propagate a growth boundary of cells from the particle locations. Distance tests only occur when growth boundaries from different particles collide with each other, hence the number of distance tests is effectively minimized. We give some empirical results for two and three dimensions. The method generalizes to any dimension in a straight forward manner. The distance tests can be based any metric.

Keywords Discrete Voronoi · Particle-in-cell · FEM · Quadrature

1 Introduction

The particle-in-cell (PIC) Finite Element Method described in [1] is a hybrid of the classical Finite Element Method (FEM) with a set of Lagrangian material points (or particles) which is used to track history-dependent fluid properties including composition, anisotropic fabric, elastic stored stresses, plastic strain and accumulated damage. Each moving particle represents a finite volume of the nearby fluid.

Within the FEM the governing equations are expressed in a weak (integral) form. The weak form is defined over a mesh of elements Ω^e, which partition the domain Ω.

M. Velić (✉) · D. May · L. Moresi
School of Mathematical Sciences, Monash University,
Building 28, Clayton, Victoria 3800, Australia
e-mail: mirko.velic@sci.monash.edu.au

For numerical work, the weak form over each element is typically approximated via a quadrature summation, i.e.,

$$I := \int_{\Omega^e} \phi(\boldsymbol{x})\, d\Omega^e \approx \sum_{p=1}^{N} w_p \phi(\boldsymbol{\xi}_p) J_p, \tag{1}$$

where $\phi(\boldsymbol{x})$ is a field quantity, N is the number of quadrature points, $\boldsymbol{\xi}_p = (\xi_p, \eta_p, \zeta_p)$ and w_p are the abscissa and weight of the p^{th} quadrature point respectively and J_p is the Jacobian transformation between the $\boldsymbol{x}$ and $\boldsymbol{\xi}$ coordinate systems.

In [1] the material points are in fact the *quadrature* points used within the summation in Eq. 1. Hence, material properties appearing in the weak form are naturally incorporated into the quadrature approximation without needing to perform any interpolation. In the context of the fluid simulation, it is appropriate to associate the quadrature weight w_p with the fluid volume, and hence require $w_p \geq 0$.

By using the material point coordinates as the quadrature abscissa, we only have the weights w_p as free parameters to construct an accurate approximation of I. This is in contrast to standard quadrature schemes such as Gauss and Lobatto quadrature where both the abscissa and weights are chosen to minimize the approximation error. To ensure optimal convergence of the finite element solution, we must consider the highest order complete polynomials appearing in the basis functions used in our approximation space [2]. In the PIC method, bilinear/trilinear basis functions are usually used which, in the two dimensional case for example, require that the following conditions on the integration weights must be satisfied [1]:

$$\sum_p w_p = 4, \tag{2}$$

$$\sum_p \xi_p w_p = 0, \qquad \sum_p \eta_p w_p = 0. \tag{3}$$

A Voronoi diagram constructed from the locations of the material points within each element Ω^e automatically satisfies Eq. 2 if we associate each w_p with the volume of the Voronoi cell of point p. To satisfy the constraints given by Eq. 3 we choose to use the centroid of the fluid volume local to the material point as the location of our quadrature point. This step does not introduce any additional approximation error. Satisfaction of Eqs. 2 and 3 still holds when the exact Voronoi diagram is replaced by a discrete approximation to the Voronoi diagram provided the approximation partitions the element exactly.

The DVC algorithm described in this paper provides a practical, robust and fast method to partition cells within the context of the PIC FEM and can be implemented efficiently on a variety of computer architectures, which is a requirement for large scale numerical simulations on parallel machines.

A Voronoi diagram is a unique partitioning of a space into a set of convex polyhedra in n-dimensions. Each polyhedron contains one generating point site and the facets of the polyhedron coincide with the hyperplane that bisects the space between the generating points. i.e., any point inside a given polyhedron is closer to the generating point of that polyhedron than it is to any other generating point. If the facets of two polyhedra coincide, then the generating points of the two polyhedra are classified as natural neighbors to each other.

Discrete Voronoi diagrams have been studied extensively and have numerous applications, including the determination of path plans for robots [3], distance approximation [4, 5], distance transforms [6–9], polygon collision detection [10, 11] and virtual networking [12]. One of the earliest examples is the cellular automaton method of Adamatzky [13, 14]. For an in depth review of Voronoi diagrams in general see [15] or [16]. For other examples of the use of Voronoi diagrams within a geophysical numerical application see [17].

A discrete implementation for constructing Voronoi diagrams was chosen for our particular application as it is robust (the computational effort scales weakly) to large variations in the number and spatial distribution of the seed points. Our algorithm specifies a spatial resolution below which we are not interested in resolving the local fluid volumes. In the PIC FEM there is little benefit in having a large amount of detail represented in the sub-grid scale information carried by the particles [18] and so the ability to limit the work spent on building the quadrature scheme to achieve a pre-determined accuracy is a strong advantage. There are many other practical computer-based applications where an approximate, discrete Voronoi diagram is a sufficient or more appropriate choice of algorithm [19]. Furthermore, a grid based discrete method is very well suited for the construction of a Voronoi diagram that is bounded in a finite region.

The discrete algorithms of [7, 8] and [9] are specifically designed to be fast when implemented to take advantage of the properties of a graphical processing unit (GPU). The range culling method of [9] reduces the number of potential distance tests performed over the grid by limiting the range from a seed that distance tests need to be done. Both algorithms still perform many distance tests but significantly improve on the method first given on a GPU by [20].

We have taken an alternative approach: to minimize the number of expensive operations and to try to achieve an efficient implementation on standard CPUs. The method used to propagate cell-ownership which is used in our DVC algorithm superficially resembles the classic cell growth method which can be seen, for example, in [16]. However our method is not a cellular automaton method, which scans the whole grid repeatedly and updates cells based on rules for each cell in the grid. Instead, our method of propagation is based on the use of a stored boundary chain and scans of cells local to the boundary; this significantly reduces the number of distance tests performed as we resolve the Voronoi diagram.

The method of [19] appears to be potentially fast on a CPU but generates an approximation to the facets of the Voronoi diagram via an adaptive grid, discarding cells that are no longer associated with the boundaries of the Voronoi regions. As such, it is not as straight-forward to extract geometrical information such as the volumes and centroid locations of Voronoi cells.

2 The Discrete Voronoi Chain Algorithm

2.1 Algorithmic Description

To construct an approximate Voronoi diagram on a domain, Ω, we partition the domain into a set of m non overlapping cells. i.e.,

$$M = \{c_1, c_2, \ldots, c_m\}, \tag{4}$$

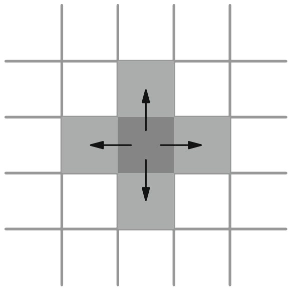


Fig. 1 A cell with its neighbors shown in *grey* for a two dimensional regular grid

where the c_k are the cells partitioning Ω such that,

$$\Omega = \bigcup_{k=1}^{m} c_k. \tag{5}$$

We also define a list of particles (generating points)

$$P = \{p_1, p_2, \ldots, p_n\}, \tag{6}$$

where each particle p_i is to claim ownership of cells c_k from M to form an approximate Voronoi diagram.

The simplest way to define each cell is usually to divide Ω into an equispaced Cartesian grid. Then all cells c_k are squares of equal size in two dimensions and cubes in three dimensions. In general, we may divide Ω into a set of polygons (or polyhedra in three dimensions). We define neighbors for each cell, such that two cells are neighbors if they share a common facet[1], as in Fig. 1.

The algorithm consists of alternatively updating two lists per particle as the particles claim ownership of cells c_k in the grid. The first list is the newly claimed cell list $\mathscr{L}_C^i$. For each iteration per particle, this list is reset to be empty and as the i^{th} particle claims cells, these cells are added to $\mathscr{L}_C^i$. From this list, the boundary chain list $\mathscr{L}_B^i$ is generated as the union of the neighbors of all the cells in $\mathscr{L}_C^i$ that are not already owned by particle p_i (see Fig. 2). Then for each particle i, $\mathscr{L}_C^i$ is reset to be empty and its boundary chain $\mathscr{L}_B^i$ is traversed. Each cell in $\mathscr{L}_B^i$ that is unowned is claimed by the i^{th} particle. Only when a cell in $\mathscr{L}_B^i$ is owned by another particle must we resolve the ownership of the cell based on a distance test: whichever particle is closest to the centroid of the cell in question wins the “battle” and claims or retains ownership of the cell as in Fig. 3. As cells are claimed they are added to the newly claimed cell list for that particle. This process is repeated until no particles can claim any new cells. i.e., when all the lists $\mathscr{L}_C^i$ are empty. The algorithm is initialized by every particle first claiming the cell that it is located within, and then building the particle’s boundary chain $\mathscr{L}_B^i$, based on this first claimed cell.

If the mesh is simply connected, then all cells are visited and eventually claimed. The algorithm will always terminate so long as a particle cannot win back a cell that it previously lost to another particle. This is always the case if the distance test is consistent.

[1]A shared facet is a common edge in two dimensions and a common face in three dimensions.

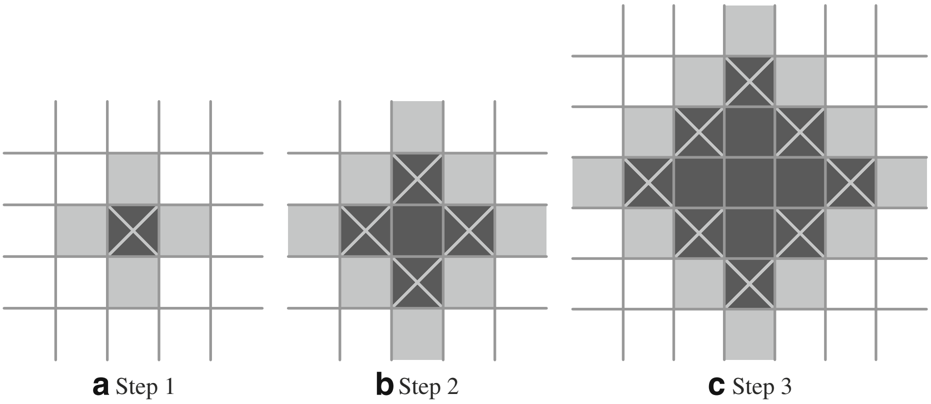


Fig. 2 Growth pattern for a single particle: *light grey cells* comprise the boundary chain list $\mathscr{L}_B$, while *dark cells* are those already claimed by the particle. Cells marked with an X are members of the newly claimed cell list $\mathscr{L}_C$, from the previous iteration (**a**–**c**)

The algorithm is summarized in Algorithm (1). The procedure initializeClaimedCellList simply returns a list containing the single cell in the mesh that the particle p_i is located in. The updateboundaryChain procedure returns a list of cells that are the neighbors to all the cells in the newly claimed cell list $\mathscr{L}_C^i$ for particle p_i, that are not already owned by the particle. The claimCells procedure traverses the cells in the boundary chain list $\mathscr{L}_B^i$ and will assign ownership of these cells to particle p_i unless the cell is owned by another particle. In this case, a battle for ownership ensues as previously described.

In the case where two particles are located in the same cell, one or the other will own that cell initially. However, each particle will have constructed boundary chains, and both will attempt to claim the cells in the chain. Ownership of these cells will be determined by distance tests. This way, a proper discrete Voronoi diagram will still

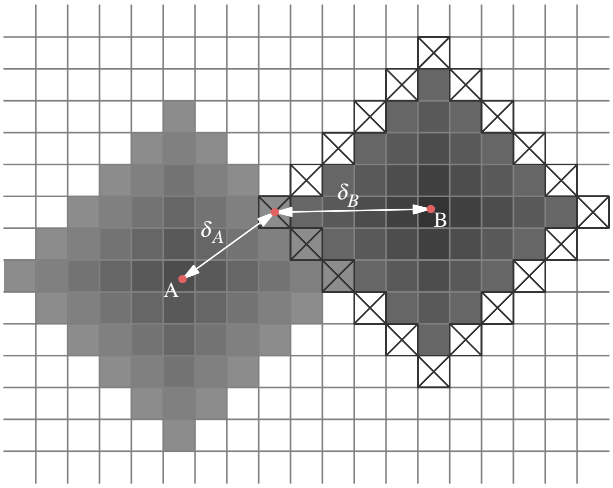


Fig. 3 A Voronoi battle: the cells marked with an X comprise the latest boundary chain of particle B. Particle B is attempting to claim a cell owned by particle A, so there must be a distance test to resolve ownership. In this case, $\delta_A < \delta_B$ so particle A retains ownership of the cell in question

Algorithm 1 Create Voronoi Diagram

```
for all $p_i \in P$ do
    $\mathscr{L}_C^i \Leftarrow$ initializeClaimedCellList($p_i$, $M$)
end for
while $\mathscr{L}_C^i \neq \emptyset \ \ \forall i$ do
    for all $p_i \in P$ do
        $\mathscr{L}_B^i \Leftarrow$ updateBoundaryChain($\mathscr{L}_C^i$, $M$, $P$)
        $\mathscr{L}_C^i \Leftarrow$ claimCells($\mathscr{L}_B^i$, $M$, $P$)
    end for
end while
```

be constructed. The figures in Fig. 4 show the convergence onto the exact Voronoi diagram for the given set of points. Figure 4a is at very low resolution and shows that one point (marked in white) is not resolved. i.e., it has claimed no cells in the Voronoi diagram. The growth pattern from each point is contiguous over the grid because the cells that are newly claimed are always neighbors to previously owned cells. When growth areas collide, the final boundary between particles is determined by distance tests. Therefore, the center of any claimed cells are closer to the particle that owns it than to any other particle. Due to this property, as the grid resolution increases, the approximate Voronoi diagram will always approach the exact Voronoi diagram. This is demonstrated by comparing Fig. 4a to c where the grid resolution is successively refined. Boundaries between particles are determined by interaction only between nearby particles. This makes the algorithm local in nature. i.e., particles that are far from each other do not interact. These properties make the DVC algorithm robust in execution time.

2.2 Further Applications and Generalizations

Whilst initially used to develop a quadrature scheme for PIC methods, the nature of the DVC algorithm has permitted its uses in numerous other areas related to our fluid dynamics application. For such numerical methods, being able to perform efficient calculations on a CPU as opposed to a GPU is important. Below, we list several

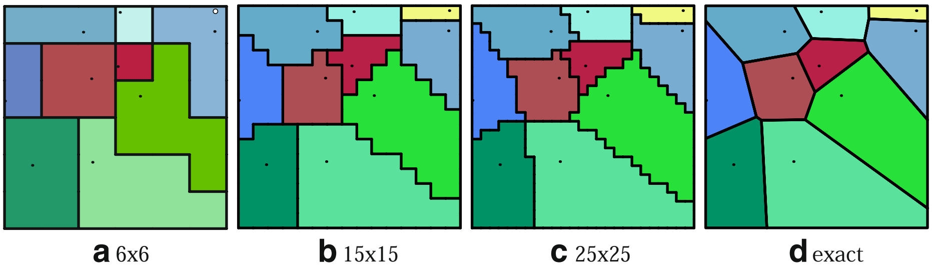


Fig. 4 A successively refined discrete Voronoi diagram for 10 particles with various resolutions, showing convergence to the exact Voronoi diagram on the right (**a**–**d**)

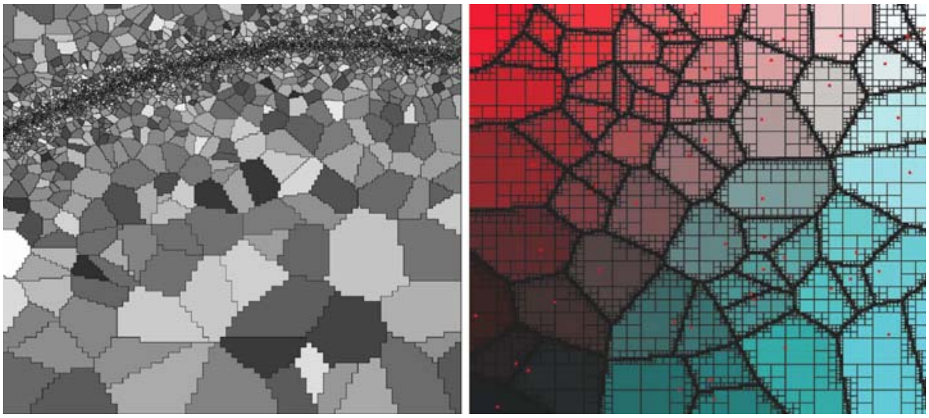

a Adaptive grid zooming in on an interface **b** Adapative grid refiningVoronoi boundaries

Fig. 5 Illustration of the DVC algorithm on adaptive grids (**a**, **b**)

applications where the DVC algorithm has proven to be useful in the context of our numerical method:

1. Irregular bounding domains on structured grids: Rather than having to mesh an irregular shaped domain via an unstructured grid, one can employ a rectangular grid of structured cells and simply "tag" regions with a unique index which are considered part of the same domain. In this way, we only allow the DVC algorithm to claim cells within the region possessing the same cell tag. Fig. 6a demonstrates this idea. This concept has applications in free surface simulations.
2. Irregular grids: For the sake of clarity, the DVC algorithm has been analyzed using regular grids only, but it should be noted that any mesh that is simply connected can be used. All that is required is a notion of local connectivity to define the neighbors to cells so that the boundary chains can be constructed and traversed. For example, in Fig. 5a the grid used is adaptively refined towards and interface. In Fig. 5b the grid is adaptively refined to better resolve the boundaries between cells in the Voronoi diagram, while in Fig. 6b the grid is an unstructured triangular mesh.

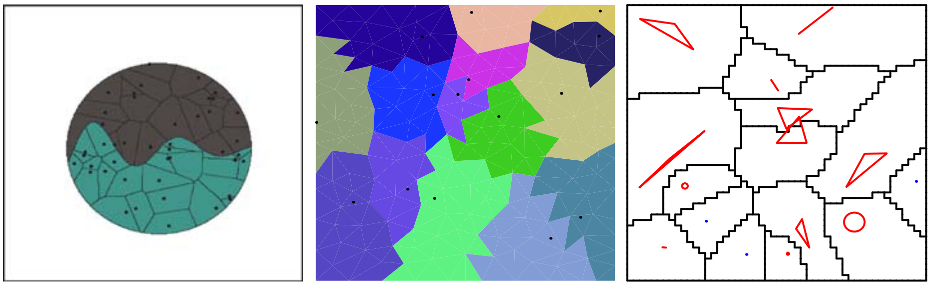

a Using tagged cells to create masks **b** Voronoi diagram on an unstructured grid **c** Generalized Voronoi diagram

Fig. 6 Illustration of various applications of the standard DVC algorithm (**a**–**c**)

3. Generalized Voronoi diagrams: In the DVC algorithm described, a Euclidean distance was employed to resolve "battles". Any metric can be used, thus enabling generalized Voronoi diagrams to be constructed as in Fig. 6c
4. Parallel Voronoi diagrams: The DVC algorithm is local in nature. This makes it straight-forward to implement a parallel implementation of the DVC algorithm: One can divide up the total mesh across processors letting each Voronoi region grow independently. Only those regions that collide with processor boundaries need be communicated to neighboring processors. This means that arbitrarily large Voronoi diagrams may be calculated using many processors in parallel.

3 Empirical Characteristics of Algorithm

We require a fast and robust algorithm for the computation of approximate Voronoi diagrams that easily yields certain geometrical information: the Voronoi region volumes as well as the location of their centroids. From Fig. 7 we see that this information can be gained with only a small overhead. Several existing discrete Voronoi algorithms of note are the JF algorithm [7], the range culling method of [9] and the adaptive algorithm of [19]. The JF algorithm appears to be quite fast while the method of [9] does not suffer from the errors of the JF algorithm. We require a method that runs efficiently on a CPU within the context of a numerical simulation, as opposed to a GPU. The DVC algorithm is error free (it will always converge to the exact Voronoi diagram) and is significantly faster than the JF algorithm when implemented on a CPU (Fig. 8). The JF algorithm relies on its efficient propagation of seed information across a grid which is suited for the parallel nature of a GPU, whilst the DVC algorithm significantly reduces the number of distance tests performed on the CPU. A comparison of the number of distance tests performed can be seen in Fig. 9.

If there are many particles for each cell in the grid, the DVC algorithm will scale linearly in time with the number of particles as all the boundary chains are the same size and the number is proportional to the number of particles. If there are very few particles compared to the mesh cells then the DVC algorithm will scale linearly with respect to the grid size. The usual application of the DVC algorithm lies between these limits. Figures 10 and 11 show some representative times of how

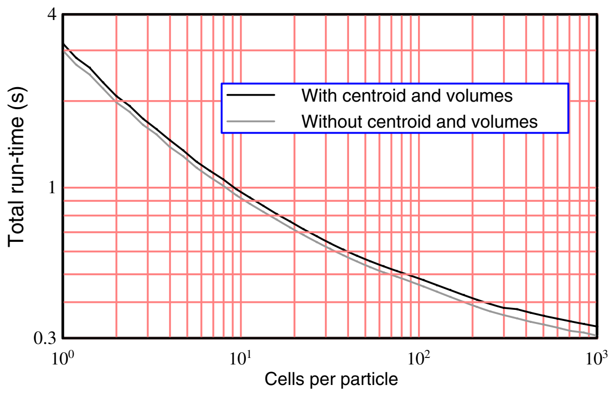


Fig. 7 Total median run times for the DVC algorithm showing only a slight overhead to compute volumes and centroidal information in two dimensions for a 1024 × 1024 grid for various particle densities

Fig. 8 Total run time for the JF vs the DVC algorithms for two dimensions on a CPU for a 1024 × 1024 grid

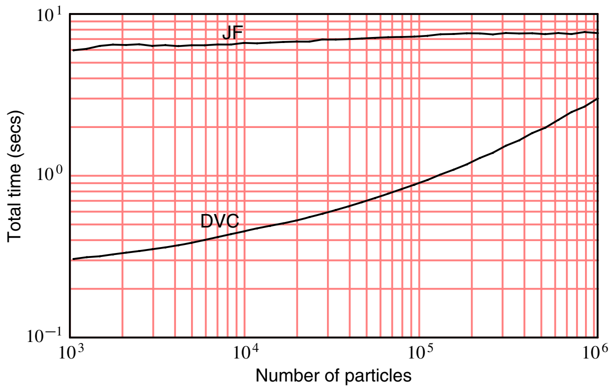


Fig. 9 Number of distance tests for the JF vs the DVC algorithms for two dimensions for a 1024 × 1024 grid

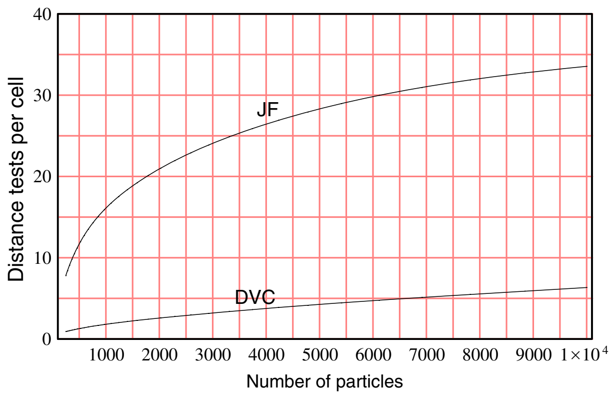
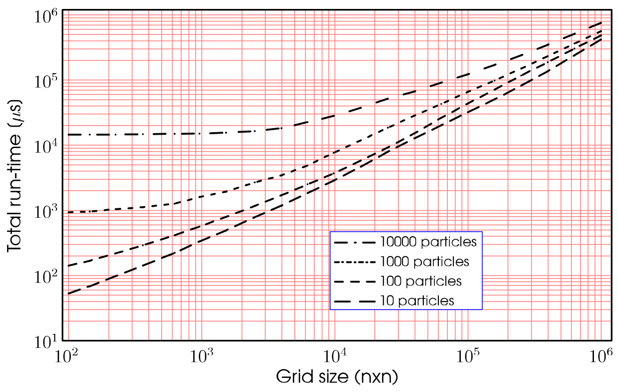


Fig. 10 Median total time in μs for construction of Voronoi diagrams for random particles in two dimensions as function of grid size for 10, 100, 1000 and 10000 particles



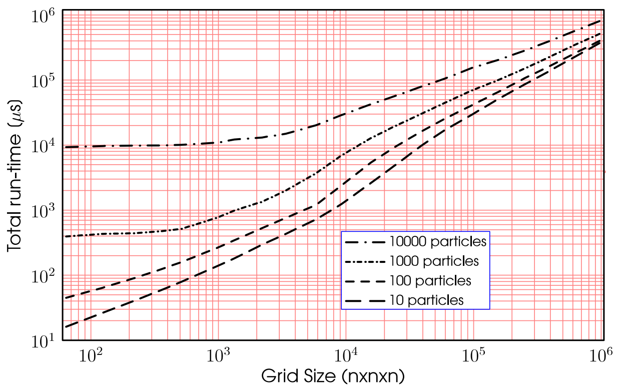


Fig. 11 Median total time in μs for construction of Voronoi diagrams for random particles in three dimensions as function of grid size for 10, 100, 1000 and 10000 particles

the algorithm scales for the construction of Voronoi diagrams for various resolutions and numbers of particles. The left hand part of the graph shows constant times for Voronoi diagrams with large numbers of particles compared to the grid resolution whilst the right hand side shows the timing lines converging when the grid resolution is large compared to the number of particles.

The storage requirement of the DVC algorithm may be proportional to either the number of grid cells or the number of particles, depending on which dominates. The algorithm is not useful if there are many more particles than there are cells in the grid. Therefore it is reasonable to take the storage to be initially proportional to the number of particles as we require that the number of grid cells be proportional to the latter. The additional storage due to the boundary chains used in the construction of the Voronoi diagram is shown in Fig. 12. The lines quickly converge on an upper limiting line. Whilst not shown here, we note that grid resolutions greater than 100^3 exhibit essentially identical behavior to the 100^3 line shown in Fig. 12. The upper limiting lines for the newly claimed cell list and the boundary chain for reasonably high resolutions as in Figs. 13 and 14 show that the additional storage for the chains is bounded per particle. Therefore we may take the storage to be $O(n)$, where n is the number of particles.

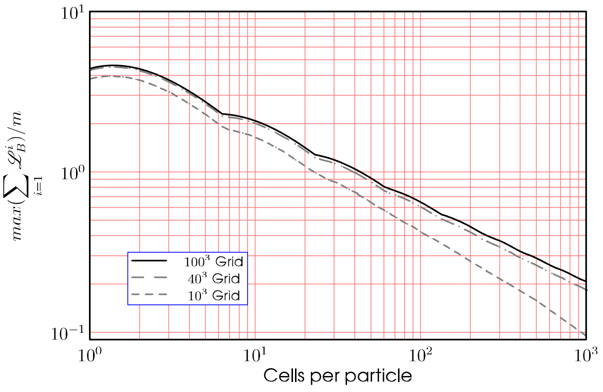


Fig. 12 Median relative maximum size of boundary chains per cell for three dimensions for varying grid resolutions showing quick convergence to a particular line

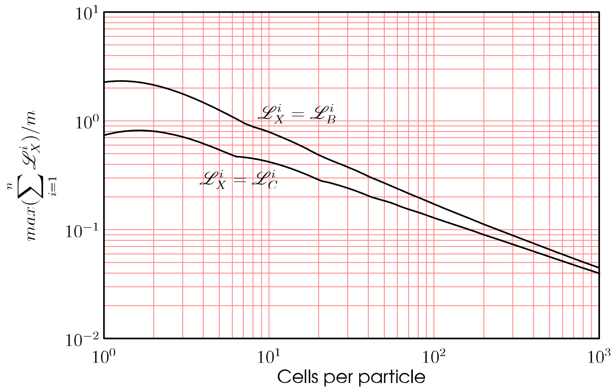


Fig. 13 Median relative maximum size of boundary chains per cell for two dimensions

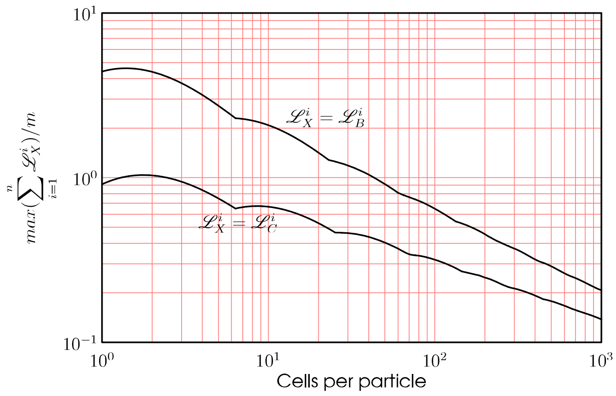


Fig. 14 Median relative maximum size of boundary chains per cell for three dimensions

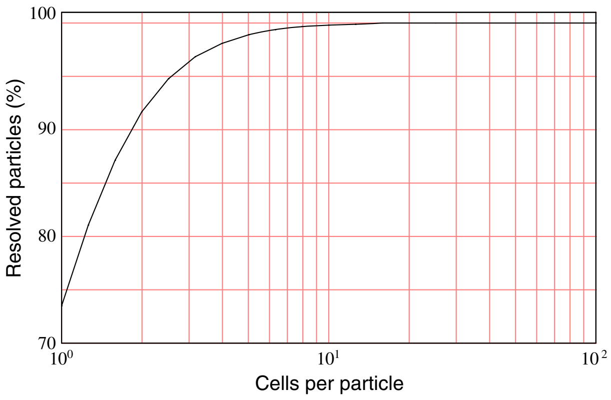


Fig. 15 Median percentage of resolved particles as a function of the number of grid cells per particle for two dimensional Voronoi diagrams

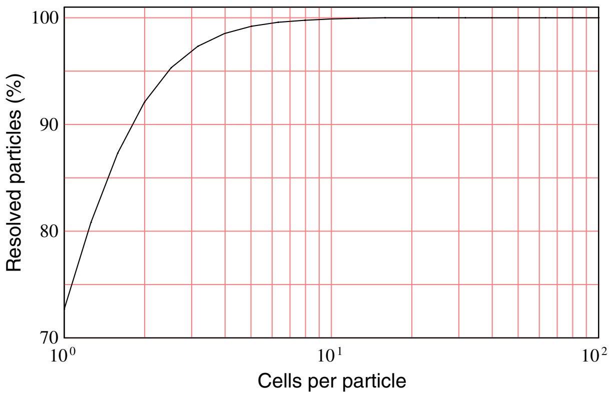


Fig. 16 Median percentage of resolved particles as a function of the number of grid cells per particle for three dimensional Voronoi diagrams

In Figs. 15 and 16 it can be seen that if there are ten or more grid cells per particle on average, then the particles in the discrete Voronoi diagram are almost always resolved. In some cases it may desirable to resolve every particle in the diagram. In this case one would create a grid that has at least ten times the number of cells as there are particles, and then check to see if all the particles are resolved. If any particles were not resolved, Figs. 15 and 16 indicate that one would only need to increase the grid size by a modest amount to ensure that every particle is resolved.

4 Conclusion

The DVC algorithm described is a fast and robust method for the construction of discrete Voronoi diagrams. The method provides a volumetric representation of the Voronoi cells by utilizing a grid which partitions the domain. The underlying grid used to define the Voronoi diagram may be structured or unstructured. The DVC algorithm is well suited for use in large scale numerical computations as its performance is not dependent on computer architecture, is easy to implement and is efficient. Whilst originally devised for constructing integration schemes for the PIC finite element method of [1] in two and three dimensions, the algorithm's flexibility makes it applicable within the context of other numerical schemes. Further applications of the DVC algorithm will be given elsewhere (also see the PhD thesis by [18]).

Acknowledgements We would like to acknowledge the use of AuScope facilities at Monash University. AuScope Ltd is funded under the National Collaborative Research Infrastructure Strategy (NCRIS) an Australian Commonwealth Government Programme. We acknowledge the support of the ARC grants LP0347688 and DP044997.

References

1. Moresi, L., Dufour, F., Mühlhaus, H.B.: A lagrangian integration point finite element method for large deformation modeling of viscoelastic geomaterials. J. Comput. Phys. **184**, 476–497 (2003)
2. Hughes, T.J.R.: The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Prentice-Hall, Englewood Cliffs (1987)

3. Tzionas, P.G., Thanailakis, A., Tsalides, P.G.: Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata. IEEE Trans. Robot. Autom. **13**(2), 237–250 (1997)
4. Tsai, Y.: Rapid and accurate computation of the distance function using grids. J. Comput. Phys. **178**(1), 175–195 (2002). ISSN 0021-9991. doi:10.1006/jcph.2002.7028
5. Bespamyatnikh, S., Segal, M.: Fast algorithms for approximating distances. Algorithmica **33**, 263–269 (2002)
6. Mauch, S.: A fast algorithm for computing the closest point and distance transform. Technical report, Caltech (2000)
7. Rong, G., Tan, T.S.: Jump flooding in GPU with applications to Voronoi diagram and distance transform. In: I3D ’06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, pp. 109–116. ACM, New York (2006). ISBN 1-59593-295-X. doi:10.1145/1111411.1111431
8. Rong, G.: Jump flooding algorithm on graphics hardware and it’s applications. Ph.D. thesis, National University of Singapore (2007)
9. Sud, A., Govindaraju, N., Manocha, D.: Interactive computation of discrete generalized Voronoi diagrams using range culling. In: Proc. 2nd International Symposium on Voronoi Diagrams in Science and Engineering. Hanyang University, October 2005
10. Mirtich, B.: V-clip: fast and robust polyhedral collision detection. ACM Trans. Graph. **17**(3), 177–208, July (1998)
11. Sud, A., Govindaraju, N., Gayle, R., Kabul, I., Manocha, D.: Fast proximity computation among deformable models using discrete Voronoi diagrams. ACM Trans. Graph. **25**(3), 1144–1153 (2006). ISSN 0730-0301. doi:10.1145/1141911.1142006
12. Hu, S.Y., Liao, G.M.: Scalable peer-to-peer networked virtual environment. In: NetGames ’04: Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games, pp. 129–133. ACM, New York (2004). ISBN 1-58113-942-X. doi:10.1145/1016540.1016552
13. Adamatzky, A.I.: Voronoi-like partition of lattice in cellular automata. Math. Comput. Model. **23**(4), 51–66 (1996)
14. Adamatzky, A.I., Holland, O.: Voronoi-like nondeterministic partition of lattice by collectives of finite automata. Math. Comput. Model. **28**(10), 73–93 (1998)
15. Aurenhammer, F.: Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Comput. Surv. **23**(3), 345–405 (1991). ISSN 0360-0300. doi:10.1145/116873.116880
16. Okabe, A., Boots, B., Sugihara, K.: Spatial Tessellations: Concepts and Applications of Voronoi diagrams. Wiley, New York (1992). ISBN 0-471-93430-5
17. Sambridge, M.S., Braun, J., McQueen H.: Geophysical parametrization and interpolation of irregular data using natural neighbours. Geophys. J. Int. **122**, 837–857 (1995)
18. May, D.A.: Ph.D. thesis, Monash University (2008)
19. Vleugels, J., Overmars, M.: Approximating Voronoi diagrams of convex sites in any dimension. Int. J. Comput. Geom. Appl. **8**(2), 201–221 (1998)
20. Kenneth, E., Hoff, I., Keyser, J., Lin, M., Manocha, D., Culver, T.: Fast computation of generalized Voronoi diagrams using graphics hardware. In: SIGGRAPH ’99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 277–286. ACM/Addison-Wesley, New York (1999). ISBN 0-201-48560-5. doi:10.1145/311535.311567