Collision Detection Optimization in a Multi-particle System

Marina L. Gavrilova and Jon Rokne

Dept of Comp. Science, University of Calgary, Calgary, AB, Canada, T2N1N4
marina@cpsc.ucalgary.ca, rokne@cpsc.ucalgary.ca

Abstract. Collision detection optimization algorithms in an event-driven simulation of a multi-particle system is one of crucial tasks, determining efficiency of simulation. We employ dynamic computational geometry data structures as a tool for collision detection optimization. The data structures under consideration are the dynamic generalized Voronoi diagram, the regular spatial subdivision, the regular spatial tree and the set of segment trees. Methods are studies in a framework of a granular-type materials system. Guidelines for selecting the most appropriate collision detection optimization technique summarize the paper.

1 Introduction

A particle system consists of physical objects (particles), whose movement and interaction are defined by physical laws. Studies conducted in the fields of robotics, computational geometry, molecular dynamics, computer graphics and computer simulation describe various approaches that can be applied to represent dynamics of particle systems [9, 8]. Disks or spheres are commonly used as a simple and effective model to represent particles in such systems (ice, grain, atomic structures, biological systems). Granular-type material system is an example of a particle system. When a dynamic granular-material system is simulated, one of the most important and a time consuming tasks is predicting and scheduling collisions among particles.

Traditionally, the cell method was the most popular method employed for collision detection in molecular dynamics and granular mechanics [8]. Other advanced kinetic data structures, commonly employed in computational geometry for solving a variety of problems (such as point location, motion planning, nearest-neighbor searches [7, 11, 1]), were seldom considered in applied materials studies.

Among all hierarchical planar subdivisions, binary space partitions (BSP) are most often used in dynamic settings. Applications of binary space partitions for collision detection between two polygonal objects were considered in [1, 4, 6]. Range search trees, interval trees and OBB-trees were also proposed for CDO [6, 3].

This paper presents an application of the weighted generalized dynamic Voronoi diagram method to solve the collision optimization problem. The idea of employing the generalized dynamic Voronoi diagram for collision detection optimization was first proposed in [2]. The method is studied in general d-dimensional space and is compared against the regular spatial subdivision, the regular spatial tree and the set of segment trees methods. Results are summarized in a form of guidelenes on the selection of the most appropriate collision detection optimization method.

2 Dynamic event-driven simulation algorithm

As of today, most of the research on collision detection in particle systems is limited to consideration of a relatively simple simulation model. The idea is to discretize time into short intervals of fixed duration. At the end of each time interval, the new positions of the moving particles are computed. The state of the system is assumed to be invariant during the time interval between two consecutive time steps. The common problem with such methods is related to choosing the length of the interval. If the duration is too short, unnecessary computations take place while no topological changes occurred. If the duration is too large, some important for analysis of the model events can be omitted. A much more effective approach, the dynamic event-driven simulation of a particle system, relies on discrete events that can happen at any moment of time rather then during fixed time steps. This can be accommodated by introducing an event queue. We employ this scheme and suggest the following classification of events: *collision events*, *predict trajectory events* and *topological events.*

A set of n moving particles in R^d is given. The particles are approximated by spheres (disks in the plane). Collision event occurs when two particles come into contact with each other or with a boundary. A predict trajectory event occurs when the trajectory and the velocity of a particle is updated due to the recalculation of the system state. Between two consecutive predict trajectory events a particle travels along a trajectory defined by a function of time.

Collision detection algorithms optimize the task of detecting collisions by maintaining a set of neighbors for every particle in the set and only checking for collisions between neighboring particles. The algorithm is said to be correct if at the moment of collision the two colliding particles are neighbors of each other (i.e. the collision is not missed). The computational overhead associated with a CDO algorithm is related to the data structure maintenance.

The Event-Driven Simulation Algorithm

1. (Initialization) Construct the topological data structure; set the simulation clock to time $t_0 = 0$; schedule predict trajectory events for all particles and place them in the queue.
2. (Processing) While the event queue is not empty do:
   (a) Extract the next event e_i from the event queue, determine the type of the event (topological event, predict trajectory event or collision event);
   (b) Advance the simulation clock to the time of this event t_e

```
(c) Process event $e_i$ and update the event queue:
    i. if the event is a topological event:
       - modify the topology of the data structure;
       - schedule new topological events;
       - check for collisions between new neighbors and schedule collision
         events (if any);
    ii. if the event is a collision event:
       - update the states of the particles participating in a collision;
       - delete all events involving these particles from the event queue;
       - schedule new predict trajectory events at current time for both
         particles;
    iii. if the event is a predict trajectory event:
       - compute the trajectory of the particle for the next time interval
         $(t_e, t_e + \Delta t]$;
       - schedule the next predict trajectory event for the particle at time
         $t_e + \Delta t$;
       - schedule new topological and collision events for the updated particle
```

Note, that new topological or collision events are never scheduled past the time of the next predict trajectory event of all particles involved.

The following criteria can be introduced to estimate the efficiency of a CDO algorithm: the total number of pairs of closest neighbors; the number of neighbors of a single particle; the number of topological events that can take place between two consecutive collision events (or predict trajectory events); computational cost of a topological event (scheduling and processing); computational cost of the data structure initialization; and space requirements.

3 Dynamic Computation Geometry Data Structures

Consider a problem of optimizing the collision detection for a set of moving particles in the context given above. In a straightforward approach each pair of particles is considered to be neighbors, i.e. there are $\frac{1}{2}n\,(n-1)$ neighbor pairs. For a large system the application of this method is very computationally expensive. Thus, our goal is to reduce the number of neighbors to be considered on each step.

3.1 The Dynamic Generalized Voronoi diagram for CDO

The dynamic generalized Voronoi diagram in Laguerre geometry is the first data structure applied for collision detection optimization.

Definition 1. A generalized Voronoi diagram (VD) for a set of spheres S in R^d is a set of Voronoi regions $GVor\,(P) = \{\mathbf{x}|\ d\,(\mathbf{x}, P) \leqslant d\,(\mathbf{x}, Q)\,, \forall Q \in S - \{P\}\}$, where $d\,(\mathbf{x}, P)$ is the distance function between point $\mathbf{x} \in R^d$ and particle $P \in S$.

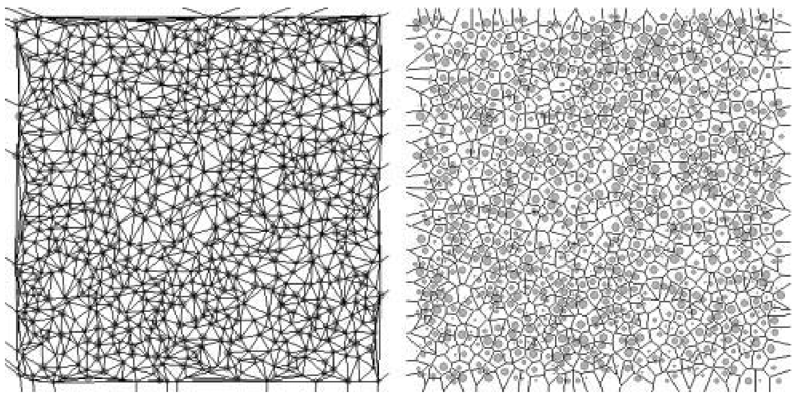

Fig. 1. The generalized Delaunay triangulation (left) and the Voronoi diagram (right) for 1000 particles in Laguerre geometry.

In Laguerre geometry, the distance between a point and a sphere is defined as $d(\mathbf{x}, P) = d(\mathbf{x}, \mathbf{p})^2 - r_P^2$, where $d(\mathbf{x}, p)$ is the Euclidean distance between $\mathbf{x}$ and $\mathbf{p}$ [10]. The generalized Voronoi diagram stores the topological information for a set of particles. Each Voronoi region represents the locus of points that are closer to the particle than to any other particle from set S. The dual to the Voronoi diagram, the Delaunay tessellation, contains the proximity information for the set of particles (see Fig. 1).

The following approach is implemented. The Delaunay tessellation (DT) is constructed in Laguerre geometry. The computation of topological events is incorporated in the Event-Driven Simulation Algorithm. To ensure the algorithms correctness, the following property should be satisfied: if two particles are in contact with each other, then there must be an edge in the Delaunay tessellation incident to these particles. Due to the fact that the nearest-neighbor property in DT in Laguerre geometry is satisfied, the dynamic generalized DT can be used for collision detection optimization.

According to [2], a topological event in the dynamic generalized VD occurs when the topological structure of the VD is modified and the proximity relationships between particles are altered. Handling of a topological event requires flipping a diagonal edge (or a facet) in a quadrilateral of the Delaunay tessellation and scheduling future topological events for the newly created quadrilaterals (a *swap* operation). A topological event occurs when the Delaunay tessellation sites comprising a quadrilateral become co-spherical.

Let $P_i = \{(x_i = x_i(t), y_i = y_i(t)), r_i\}$, $i = 1..d+2$ be a set of spheres with centers $(x_i(t), y_i(t))$ and radii r_i. The computation of the time of a topological event requires solving equation: $F(P_1(t), P_2(t), ..., P_{d+2}(t)) = 0$.

Lemma 1. *The time of the topological event in a Delaunay d-dimensional quadrilateral of* $d+2$ *spheres* $P_i = \{(x_i = x_i(t), y_i = y_i(t)), r_i\}$*, can be found in La-*

guerre geometry as the minimum real root t_0 of the equation

$$F\left(P_1, P_2, ..., P_{d+2}\right) = \begin{vmatrix} x_{11} & x_{12} & ... & x_{1d} & w_1 & 1 \\ x_{21} & x_{22} & ... & x_{2d} & w_2 & 1 \\ ... & ... & ... & ... & ... & ... \\ x_{d+1,1} & x_{d+1,2} & ... & x_{d+1,d} & w_{d+1} & 1 \\ x_{d+2,1} & x_{d+2,2} & ... & x_{d+2,d} & w_{d+2} & 1 \end{vmatrix} \tag{1}$$

where $w_i = x_{i,1}^2 + x_{i,2}^2 + ... + x_{i,d}^2 - r_i^2, i = 1..d+2$ and the following condition is satisfied:

$$CCW\left(P_1, P_2, ..., P_{d+1}\right) = \begin{vmatrix} x_{11} & x_{12} & ... & x_{1,d} & 1 \\ x_{21} & x_{22} & ... & x_{2,d} & 1 \\ ... & ... & ... & ... & ... \\ x_{d+1,1} & x_{d+1,2} & ... & x_{d+1,d} & 1 \end{vmatrix} > 0. \tag{2}$$

Performance Analysis

The detailed performance analysis for the CDO algorithm employing dynamic DT in Laguerre geometry follows. Some of the estimates apply to all CDO employing different data structures, while some are specific for the Delaunay triangulation based approach.

First, consider the planar Delaunay triangulation. During the preprocessing stage the DT is constructed in $O(n \log n)$ using the sweep-plane technique. The space required to store the data structure is $O(n)$. Placing the initial events into the event queue takes $O(n)$ (since they all occur at time $t = 0$).

The upper bound on the number of predict trajectory events at any moment of time in the queue is $O(n)$, since only one predict trajectory event is scheduled for each particle. The upper bound on the number of collision events at any moment of time in the queue is $O(n^2)$, since a possible collision can be scheduled for each pair of particles. It is independent of the CDO data structure. The upper bound on the number of topological events stored in the queue is $O(n)$ at any moment of time, since only one topological event is scheduled for every Delaunay triangulation edge.

The algorithm efficiency also depends on the number of *collision checks* that need to be performed in order to determine if a collision event needs to be scheduled. The number of collision checks that need to be performed after a predict trajectory event is the number of neighbors of the particle. This number is $O(n)$ in the worst case. However, for planar DT, the average number of neighbors of a particle is a constant. The maximum number of collision checks per topological event is equal to the number of new neighbors of a particle after the topological event occurs. For the dynamic DT, this number is exactly one (since one new edge appears in the DT due to a topological event).

Processing a topological event requires scheduling up to five new topological events (one for the new edge and one for each of the four neighboring quadrilaterals). It also requires performing one collision check and scheduling one possible collision event. Thus, the overall complexity of this step is $O(logn)$.

The most time consuming step in processing of a collision event in the worst case is deleting up to $O(n)$ previously scheduled events. If every particle contains a list of references to all events involving this particle, then they can be deleted in $O(n)$ time. Thus, the overall complexity of this step is $O(n)$.

Processing a predict trajectory event requires performing collision checks and scheduling new collision events (which is $O(n)$ in the worst-case). It also requires scheduling new topological events for the particle, which can result in scheduling new topological events for all edges of DT adjacent to this particle ($O(n)$ in total). Thus, the overall complexity of this step is $O(n \log n)$.

The total number of collisions between particles during the simulation cannot be bounded since the simulation time is unlimited. Hence, the overhead associated with the use of a particular CDO algorithm is usually estimated by the maximum number of topological events that can occur between two consecutive collisions. For a planar Delaunay triangulation, the number can be as large as $O\left(n^2\lambda_s(n)\right)$, where $\lambda_s(n)$ is the maximum length of an (n,s)-Davenport-Schinzel sequence, or as low as $O(1)$ for densely packed systems. The above discussion is summarized in Table 1, Section 3.5.

In higher dimensions, only a few estimates differ. The worst-case number of topological events that can happen between two consecutive collisions is $O\left(n^d\lambda_s(n)\right)$, the initialization step takes $O\left(n^{\lceil d+1/2\rceil}\right)$ (using an incremental construction technique), the space required is $O\left(n^{\lceil d/2\rceil}\right)$.

3.2 The Regular Spatial Subdivision

The regular spatial subdivision is a data structure that is traditionally used for collision detection optimization. The performance of the method strongly depends on the ratio between the maximum size of the particle and the diameter of the cell. Our contribution is in establishing the condition under which the number of the particles in the cell is a constant, thus guaranteeing a successful application of this method for CDO.

The space is subdivided into axis-parallel hypercubes in R^d. These are generically called cells in the sequel. A particle is said to reside in a cell if its center belongs to the cell. Each cell contains a list of particles that currently reside in it. The set of neighbors of a particle comprises all particles residing in the same or any of the $3^d 1$ neighboring cells. To ensure correctness, the size of a cell must be greater or equal to the diameter of the largest particle. Then, if two particles are in contact, they are guaranteed to reside in the same or in the two neighboring cells. Each particle $P_i = (\mathbf{p}_i, r_i)$ is represented by a pair consisting of the coordinates of its center $\mathbf{p}_i = \mathbf{p}_i(t)$ and the radius r_i. Assume that the size of the simulation domain is such that there are k cells in each direction. Consider a d-dimensional box with a diameter l as a simulation domain. The size of a cell must exceed the diameter of the largest particle. Thus, k is defined as the diameter of the simulation domain divided by the diameter of a largest particle $M = \max_{P_i \in S}(2r_i)$, i.e. $k = \lceil l/M \rceil$. The diameter of the smallest particle is denoted by $m = \min\limits_{P_i \in S}(2r_i)$.

Assumption 1. The ratio $\gamma = M/m$ between the maximum and the minimum diameter is invariant of n.

Lemma 2. *Under Assumption 1, the maximum number n_c of particles within each cell is bounded by a constant.*

A topological event in the regular spatial subdivision occurs when the center of a particle moves from one cell to another. The time of a topological event can be determined exactly by computing the time when the center of the particle passes the boundary of a cell.

Performance analysis

The space required to store the data structure is $O\left(k^d + n\right)$. The regular spatial subdivision can be constructed by first allocating k^d cells and then placing each of the particles into the appropriate cell based on their coordinates at the moment $t = 0$. The cells are stored in a multi-dimensional array and are accessed directly in $O(1)$.

For each particle only one topological event can be scheduled at any particular moment of time. Therefore, the upper bound on a number of topological events stored in the queue is $O(n)$ at any moment of time. Upper bounds on collision and predict trajectory event are invariant of the data structure used.

Collision checks after topological event are performed between particles that reside in neighboring cells. Since the topological event occurs when a particle moves into one of these neighboring cells, collision checks with particles from some of these cells were computed previously. Thus, only particles in 3^{d-1} *new* neighboring cells must be checked for collisions. The total number of collision checks after topological event is the number of new neighbors of a particle and is $O(1)$ under Assumption 1. Therefore, the total number of collision checks per predict trajectory event is also a constant.

Processing a topological event requires scheduling one new topological event (move to a new cell). It also requires performing $O(1)$ collision checks with new neighbors and scheduling the detected collision events. Thus, the overall complexity of this step is $O(logn)$.

Processing a predict trajectory event requires performing collision checks and scheduling new collision events. Since each cell contains only a constant number of particles (according to Lemma 1), only a constant number of collision events will be scheduled. It also requires scheduling one new topological event. Thus, the overall complexity of this step is $O(logn)$. Following the same arguments as for the dynamic DT, processing of the collision event takes $O(n)$.

Finally, since a particle can cross maximum k cells before it collides with the boundary, the number of topological events between two collisions is $O(nk)$.

Note 2. The efficiency of this method strongly depends on the distribution of particle diameters. For the algorithm to perform well, the maximum number

n_c of particles that can fit into a cell must be smaller than the total number of particles n.

3.3 The Regular Spatial Tree

This approach is a modification of the regular spatial subdivision method that reduces the memory overhead at the expense of increased access time. We propose the following approach. The non-empty cells are stored in an AVL tree according to the lexicographical ordering on their coordinates. Each node of the tree is associated with a cell in the d-dimensional Euclidean space and implicitly with a non-empty set of particles $\{P_{i_1}, P_{i_2}, ..., P_{i_l}\}$, whose centers belong to the cell.

The method reduces the storage to $O(n)$, since the number of occupied cells cannot exceed the total number of particles in the system. On the other hand, each cell access now requires $O(\log n)$ time. All the complexity estimates obtained for the regular spatial subdivision method hold with the following exception. Any operation involving modifications of the data structure (such as moving a particle from one cell into another) will now require $O(\log n)$ time. Hence, each topological event requires $O(\log n)$ operations. The initial construction of the data structure takes $O(n \log n)$ time, independent of the dimension.

3.4 The Set of Segment Trees

This is an original method proposed for collision detection optimization. We maintain a set of trees of intersecting segments, obtained by projecting the bounding boxes of particles onto the coordinate axes. The particles are said to be neighbors if their bounding boxes intersect. The algorithm is correct since if two particles are in contact, then their bounding boxes intersect.

A segment tree, represented as an AVL tree, is maintained for each of the coordinate axis. The size of each tree is fixed, since the total number of particles does not change over time. For every particle, associated list of its neighbors is dynamically updated. A topological event occurs when two segment endpoints on one of the axes meet. This indicates that the bounding boxes of the two corresponding particles should be tested for intersection. Positive test identifies that the particles became neighbors, thus their neighbor lists are updated and collision check is performed.

As the particles move, it is necessary to maintain the sequence of segment endpoints in sorted order for each of the coordinate axes. When two neighboring endpoints collide, they are exchanged in the tree. Note that we do not rebalance the tree, but exchange references to segment endpoints.

Performance analysis

The segment tree is constructed initially by sorting the segment endpoints in $O(n \log n)$ time. The space required is $O(n)$.

The upper bound on the number of topological events stored in the event queue is $O(n)$ at any moment of time, since every segment endpoint can only collide with either of the neighboring endpoints of another segment, and there are $2d$ endpoints for every particle. Upper bounds on collision and predict trajectory event are the same as in Section 3.1.

At most one collision check is performed per topological event. Note that a half of the collisions between segment endpoints, when two segments start to intersect, might result in collision checks between particles. The other half correspond to topological events, when two segments stop intersecting (no collision checks are required for these events, though the neighbor lists are updated). The number of collision checks per predict trajectory event is estimated as follows.

Lemma 3. *Under Assumption 1, the total number of collision checks per predict trajectory event is $O(1)$.*

Processing a topological event requires scheduling up to two new topological event (one for each new neighboring segment endpoint). Thus, the overall complexity of this step is $O(logn)$. Processing a predict trajectory event requires scheduling up to $4d$ new topological events (two for every segment endpoint). Only a constant number of collision events will be scheduled. Thus, the complexity of this step is $O(logn)$. The overall complexity of processing a collision event is $O(n)$.

Lemma 4. *If the particles move along the straight-line trajectories, then the upper bound on the number of topological events that can take place between two consecutive collisions is $O(n^2)$.*

3.5 Summary of Performance Analysis

The complexities of the presented algorithms for the planar case are now summarized. The following notations are used:
A the upper bound on the number of neighbors of a particle
B maximum number of neighbors appearing due to a topological event
C time per topological event (excluding collision checks)
D time per predict trajectory event
E maximum number of topological events between two collisions
F initialization
G space

4 Conclusion

The analysis of algorithm performance is summarized as follows.

1. The worst-case number of topological events that can happen between two collisions is the largest for the Delaunay tessellation method. This method should only be used in particle systems with high collision rate.

Table 1. Algorithms performance in d-dimensions

Algorithm	A	B	C	D	E	F	G
Dynamic DT	$O(n)$	1	$O(\log n)$	$O(\log n)$	$O(n^2\lambda_s(n))$	$O\left(n^{\lceil d+1/2\rceil}\right)$	$O\left(n^{\lceil d/2\rceil}\right)$
Subdivision	$O(1)$	$O(1)$	$O(\log n)$	$O(\log n)$	$O(nk)$	$O(n+k^d)$	$O(n+k^d)$
Spatial tree	$O(1)$	$O(1)$	$O(\log n)$	$O(\log n)$	$O(nk)$	$O(n\log n)$	$O(n)$
Segment tree	$O(1)$	0 or 1	$O(\log n)$	$O(n)$	$O(n^2)$	$O(n\log n)$	$O(n)$

2. The regular spatial subdivision is the most space consuming method and should not be used if the memory resources are limited.
3. The regular spatial subdivision and the regular spatial tree methods perform worst for densely packed granular systems.
4. The Delaunay tessellation based method is the only method which performance is independent of the distribution of radii of the particles and their packing density.
5. In order to implement the regular subdivision method the size of the simulation space and the size of the largest particle in the system should be known in advance. This information is not required for other data structures considered.

References

1. Agarwal, P., Guibas, L., Murali, T. and Vitter, J. Cylindrical static and kinetic binary space partitions, in Proceedings of the 13th Annual Symposium on Computational Geometry (1997) 39–48.
2. Gavrilova,M., Rokne,J. and Gavrilov, D. Dynamic collision detection algorithms in computational geometry, 12th European Workshop on CG, (1996) 103–106.
3. Gottschalk,S., Lin,M. and Manocha,D. OBBtree: A hierarchical data structure for rapid interference detection, Computer Graphics Proc., (1996) 171–180.
4. Held,M., Klosowski,J. and Mitchell,J. Collision detection for fly-throughs in virtual environments, 12th Annual ACM Symp. on Comp. Geometry (1996) V13–V14
5. Hubbard, P. Approximating polyhedra with spheres for time-critical collision detection, ACM Transaction on Graphics, **15(3)** (1996) 179–210.
6. Kim, D-J., Guibas, L., Shin, S-Y. Fast collision detection among multiple moving spheres, IEEE Transactions on Visualization and Computer Graphics, **4(3)** (1998).
7. Lee, D.T., Yang, C.D., and Wong, C. K. Rectilinear Paths among Rectilinear Obstacles, ISAAC: 3rd Int. Symp. on Algorithms and Computation (1996)
8. Milenkovic,V. Position-based physics: Simulating a motion of many highly interactive spheres and polyhedra, Comp. Graph., Ann. Conf. Series (1996) 129–136.
9. Mirtich, B and Canny, J. Impulse-based simulation of rigid bodies, in Symposium on Interactive 3D Graphics, (1995) 181–188.
10. Okabe, A., Boots, B. and Sugihara, K. Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, John Wiley Sons, England (1992).
11. van de Stappen, A.V., M.H. Overmars, M. de Berg, and J. Vleugels. Motion planning in environments with low obstacle density. Discrete Computational Geometry 20 (1998) 561–587.