Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 226 (2007) 1710–1730
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Three-dimensional boundary detection for particle methods

Aamer Haque [a,*], Gary A. Dilts [b]

[a] *ASC/Alliances Center for Astrophysical Thermonuclear Flashes, University of Chicago, RI-464, Chicago, IL 60637, United States*
[b] *Los Alamos National Laboratory, MS D413, Los Alamos, NM 87544, United States*

Received 29 August 2006; received in revised form 2 May 2007; accepted 8 June 2007
Available online 22 June 2007

Abstract

The three-dimensional exposure method for the detection of the boundary of a set of overlapping spheres is presented. Like the two-dimensional version described in a previous paper, the three-dimensional algorithm precisely detects void opening or closure, and is optimally suited to the kernel-mediated interactions of smoothed-particle hydrodynamics, although it may be used in any application involving sets of overlapping spheres. The principle idea is to apply the two-dimensional method, on the surface of each candidate boundary sphere, to the circles of intersection with neighboring spheres. As the algorithm finds the exact solution, the quality of detection is independent of particle configuration, in contrast to gradient-based techniques. The observed CPU execution times scale as $O(MN^{\epsilon})$, where M is the number of particles, N is the average number of neighbors of a particle, and ϵ is a problem-dependent constant between 1.6 and 1.7. The time required per particle is comparable to the amount of time required to evaluate a three-dimensional linear moving-least-squares interpolant at a single point.

Keywords: Particle methods; Meshless methods; Meshfree methods; Smooth particle hydrodynamics; Computational geometry

1. Introduction

In recent years there has been much development of meshfree methods for computational mechanics. Among these are the smoothed-particle hydrodynamics (SPH) [1,2], element-free Galerkin (EFG) [3], reproducing kernel particle method (RKPM) [4], and moving-least-squares-particle hydrodynamics (MLSPH) [5,6] methods. Common to all of these is the replacement of a conventional mesh composed of non-overlapping cells, zones or elements with a scattered set of overlapping disks or spheres, each supporting a kernel function for local weighting of information. We refer to such a disk or sphere as a "particle", and the broad class of "meshfree" methods as particle methods, in deference to the original meshfree method, SPH.

* Corresponding author. Tel.: +1 312 953 7085; fax: +1 773 834 3230.
E-mail addresses: ahaque@flash.uchicago.edu (A. Haque), gad@lanl.gov (G.A. Dilts).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2007.06.012

Particle methods have a natural advantage over meshed methods for problems in which topologically discontinuous deformations such as void creation and collapse, fracture, spallation, fragmentation, splashing and folding occur. Finite difference or finite element methods need expensive finely-resolved meshes to capture the detail of these dynamic phenomena. On the other hand, implementation of boundary conditions is not so clear-cut with particle methods as it is with meshed methods. One must first locate the points that comprise the boundary. With meshed methods, this is straightforward, but with meshfree methods, it is problematic.

Randles and Libersky [2] have suggested using the sums of the gradients of SPH kernels. Ideally, these kernel gradients sum to zero for interior particles. Any particle for which the sum of the kernels is not zero is presumably an exterior particle. This method gives correct results when the particles are uniformly spaced, but for non-uniform spacing the kernel sums are far from ideal, even unpredictable, and a useful specification of the trigger level for boundary detection remains elusive. Furthermore, if the SPH kernels are corrected as in MLS so that the interior kernel gradient sums are exactly zero in the interior [5], the correction spills over the boundary a little bit, and boundary particles are indistinguishable from interior particles.

Dilts [6] has proposed a purely geometric two-dimensional boundary detection algorithm, dubbed the "exposure method", such that in two-dimensions the "exact" boundary is always found. We draw a circle of radius h_i for each particle i, where h_i is the smoothing length of the kernel centered at particle i. The circle associated with particle i will simply be referred to as circle i. Assume that the neighbors of every particle have been predetermined by one of the usual techniques (KD tree, quad-tree, etc.) Now consider the neighbor particles of particle i and draw their corresponding circles. For every neighbor circle j that intersects circle i we find the arc that circle j covers on circle i. If the union of the set of arcs from neighboring circles completely covers circle i, then particle i is an interior particle. However, if circle i is not completely covered then particle i is a boundary particle. The coverage is determined by applying a quick sort to left endpoints of the arcs, and comparing the right endpoints of the sorted set. The operation count of this procedure is $O(MN\log N)$, where M is the number of particles and N is the average number of neighbors of a particle. For details, see Ref. [6]. The boundary so determined is "exact" because in SPH, typically symmetrized kernels yield pair interactions that appear and disappear precisely when the radius-h circles touch or do not touch, respectively. The exposure method finds exactly those particles which are not completely bathed in interacting neighbors. The exposure boundary is "exact" also because it is precisely what would be seen if a physical model of the particle configuration were constructed.

In this paper, we propose an extension of the two-dimensional algorithm of Dilts [6] to three dimensions. A candidate *boundary circle* with a set of *surface arcs* created by intersections with neighboring circles is replaced by a candidate *boundary sphere* with a set of *surface circles* created by intersections with neighboring spheres. The chief idea is to apply the two-dimensional boundary detection scheme to the set of intersection circles on the surface of the candidate boundary particle. If any arc of an intersection circle is exposed, then the candidate is a boundary sphere. This criterion produces a boundary identification exactly the same as would be determined by looking at the outer surface of a three-dimensional physical model of the particle configuration. The three-dimensional exposure method thus produces the exact solution to the problem.

2. Computational details

Let S_i denote the sphere of radius h_i (the particle's smoothing length) centered at particle i. Assume as in the two-dimensional case that all particle neighbors have been determined by some means. Let C_{ij} denote the oriented circle on sphere i given by the intersection of spheres S_i and S_j. Note that $C_{ij} \neq C_{ji}$ because these circles are assigned a different orientation, as explained in Section 2.1. In words, the algorithm proceeds as follows: sphere S_i is intersected with all the neighboring spheres S_j and the circles of intersection C_{ij} are drawn on sphere S_i. If these circles of intersection cover the surface of sphere S_i, then particle S_i is an interior particle. If sphere S_i is not completely covered, then particle S_i is a boundary particle. The determination of when a sphere is covered by a set of circles on its surface is not as simple as in the case of two-dimensional disks and arcs. We describe below a technique to apply the two-dimensional exposure method of Dilts [6] to each neighbor circle C_{ij} on the surface of sphere S_i. If any part of any neighbor circle C_{ij} is exposed, then particle i is a boundary particle. These ideas are illustrated in Fig. 1.

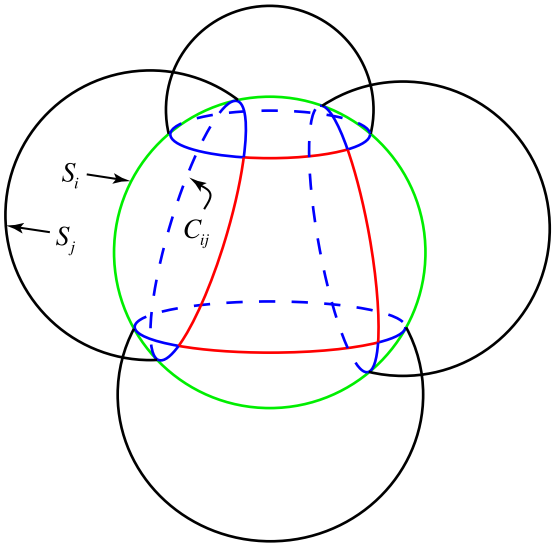

Fig. 1. Green circle represents candidate boundary sphere S_i. Black arcs represent portions of neighboring spheres S_j. Circles of intersection C_{ij} are in blue. Red arcs are those portions of circles of intersection which are not covered. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The following pseudo-code describes the high-level organization of the algorithm:

find_boundary_3D
loop over all particles i
  loop j over neighbors of i
    if S_j contains S_i **then**
      particle i is interior
      continue with next i
    end if
    if S_j does not intersect S_i **then continue** with next j
    $C_{ij} = \mathit{sphere\_intersection}(S_i, S_j)$.
    if S_j is a known interior particle **then** mark C_{ij} covered.
  end loop
  check_sphere_coverage(S_i)
end loop
check_sphere_coverage(S_i)
if there are no circles on S_i **then**
  S_i is a boundary particle
  exit the algorithm.
end if
Sort the C_{ij} by largest to smallest $C_{ij}^{(s)}$
loop over all circles j
  Construct interaction list for C_{ij}
  Set $\mathcal{L}_{ij} := [0, 2\pi)$
end loop
loop over all circles j in sorted order
  if C_{ij} is covered **then continue** with next j
  loop k over the interaction list of C_{ij}
    circle_intersection(C_{ik},C_{ij})
    circle_intersection(C_{ij},C_{ik})
    Remove C_{ij} from the interaction list of C_{ik}

```
    end loop
  end loop
  if for every j, C_ij is covered then
    S_i is an interior particle
  else
    S_i is a boundary particle
  end if
  circle_intersection(C_ik, C_ij)
  if C_ik is parallel to C_ij then
    if C_ik covers C_ij by relation 17 then
      return covered
    else
      return uncovered
    end if
  end if
  Determine number of points of intersection of C_ik and C_ij.
  if there are 2 points of intersection then
    Compute A_ijk by Eq. (57)
    Update L_ij by Eq. (62)
    if L_ij = ∅ then
      return covered
    else
      return uncovered
    end if
  else
    if C_ik covers C_ij by relation (17) then
      return covered
    else
      return uncovered
    end if
  end if
```

The rest of this section will provide the mathematical details of the three major functions

sphere_intersection
check_sphere_coverage
circle_intersection

of this high-level description in more detail.

2.1. Intersection of two spheres

In computing the intersection of neighboring spheres S_j with the candidate boundary sphere S_i, five possible cases can arise. Let $\mathbf{r}_i = (x_i, y_i, z_i)$ be the center of sphere S_i and likewise $\mathbf{r}_j = (x_j, y_j, z_j)$ is the center of S_j. Define $\Delta_{ij}\,\mathbf{r} = \mathbf{r}_j - \mathbf{r}_i = \langle \Delta_{ij}x, \Delta_{ij}y, \Delta_{ij}z\rangle$, and $\Delta_{ij} = \|\Delta_{ij}\mathbf{r}\|$, where $\|\mathbf{a}\| = \|\langle a_x, a_y, a_z\rangle\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$ denotes the euclidean norm. The five cases are:

(1) $\Delta_{ij} > h_i + h_j \Rightarrow S_j$ and S_i do not intersect.
(2) $h_j \geqslant \Delta_{ij} + h_i \Rightarrow S_j$ contains S_i.
(3) $h_i \geqslant \Delta_{ij} + h_j \Rightarrow S_i$ contains S_j.
(4) $\Delta_{ij} = h_i + h_j \Rightarrow S_j$ and S_i intersect at one point.

(5) $\Delta_{ij} < h_i + h_j$ and —$h_i - h_j$— $< \Delta_{ij} \Rightarrow$ Surfaces of S_j and S_j intersect at circle C_{ij}.

Case 4, where the spheres intersect at one point, is considered the same as if there spheres did not intersect because a single point of intersection covers no area on the surface of a sphere. If the spheres intersect at more than one point, then the following quantities associated with circle C_{ij} from Fig. 2 are needed:

$$\cos\theta_{ij} = \frac{\Delta_{ij}^2 + h_i^2 - h_j^2}{2\Delta_{ij}h_i}, \tag{1}$$

$$\lambda_{ij} = \frac{h_i\cos\theta_{ij}}{\Delta_{ij}}. \tag{2}$$

The center, $\mathbf{c}_{ij}$ and radius, $C_{ij}^{(r)}$ of C_{ij} are found from

$$\mathbf{c}_{ij} = \langle c_{ij}^{(1)}, c_{ij}^{(2)}, c_{ij}^{(3)} \rangle = \mathbf{r}_i + \lambda_{ij}\Delta_{ij}\mathbf{r}, \tag{3}$$

$$C_{ij}^{(r)} = \sqrt{h_i^2 - (h_i\cos\theta_{ij})^2} = h_i\sin\theta_{ij}. \tag{4}$$

These quantities are labeled in Fig. 2.

The circles of intersection lie on the surface of sphere S_i and are not necessarily co-planar, as shown in Fig. 4. It is useful to assign each circle an outward unit normal vector $\mathbf{n}_{ij}$, and two unit axis vectors $\hat{\mathbf{x}}_{ij}$ and $\hat{\mathbf{y}}_{ij}$ (labeled in Fig. 3) in the following manner:

$$\mathbf{n}_{ij} = \frac{1}{\Delta_{ij}}\Delta_{ij}\mathbf{r}, \tag{5}$$

$$\hat{\mathbf{x}}_{ij} = \begin{cases} \frac{\hat{\mathbf{x}}\times\mathbf{n}_{ij}}{\|\hat{\mathbf{x}}\times\mathbf{n}_{ij}\|} & \text{if } \hat{\mathbf{y}}\times\mathbf{n}_{ij} = \mathbf{0}, \\ \frac{\hat{\mathbf{y}}\times\mathbf{n}_{ij}}{\|\hat{\mathbf{y}}\times\mathbf{n}_{ij}\|} & \text{if } \hat{\mathbf{y}}\times\mathbf{n}_{ij} \neq \mathbf{0}, \end{cases} \tag{6}$$

$$\hat{\mathbf{y}}_{ij} = \frac{\mathbf{n}_{ij}\times\hat{\mathbf{x}}_{ij}}{\|\mathbf{n}_{ij}\times\hat{\mathbf{x}}_{ij}\|}, \tag{7}$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the global unit vectors in positive X and Y directions, respectively.

2.2. Checking coverage of a sphere

After the all the circles of intersection C_{ij} on sphere S_i have been found, we determine if any are not fully covered by a modified version of the 2D arc method from Ref. [6]. If so, then S_i represents a boundary particle.

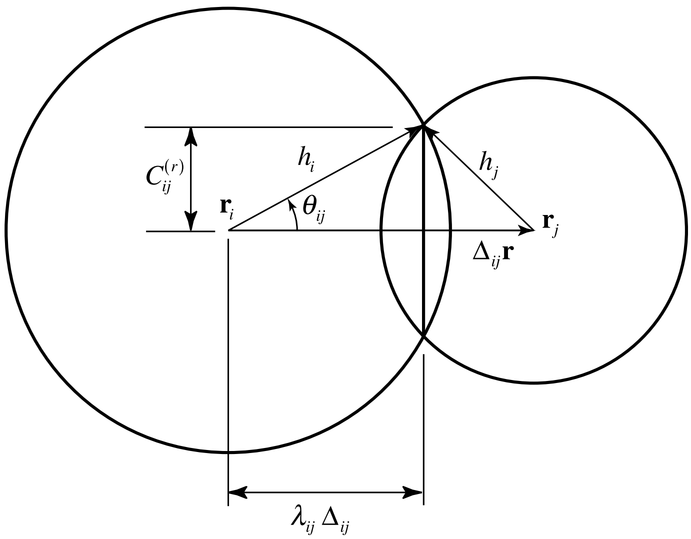

Fig. 2. Nomenclature for the intersection of two spheres.

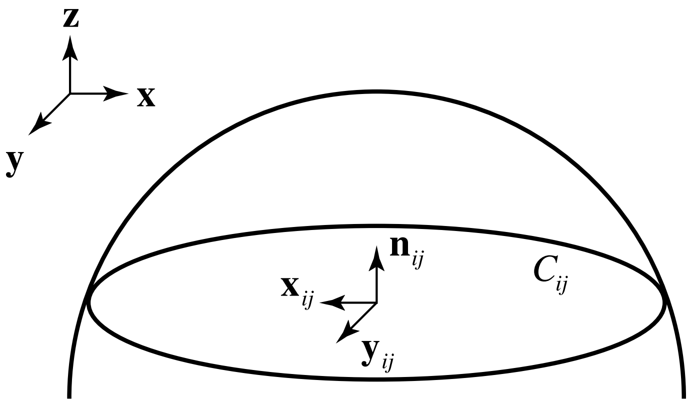

Fig. 3. Local coordinate system for a circle on the surface of a sphere.

If all the circles C_{ij} for the sphere S_i are covered, then sphere S_i is covered and represents an interior particle. If a sphere has no circles on it, then its corresponding particle is isolated and is thus a boundary particle.

The 2D arc method checks coverage of a given circle by seeing if the union of the arcs of intersection with all the other circles on the surface of sphere S_i contains the given circle. The efficiency of this process is enhanced by borrowing an idea from Iwai et al. [7]. Before computing the circle intersections, we assign a size $C_{ij}^{(s)}$ to each circle, representing the angle subtended about the origin of S_i, and sort the circles C_{ij} by $C_{ij}^{(s)}$. By considering the largest circles first we enhance the probability of completely covering up more circles early on. Once a circle is determined to be covered, it is not required to compute any more intersections for it. Since circle intersections are the expensive part of this algorithm, the overall cost is reduced by having to consider fewer than the nominal N_i^2 intersections, where N_i is the number of neighbors of S_i. The angle $C_{ij}^{(s)}$ estimates how much of the surface of sphere i is covered by C_{ij} and is given by:

$$C_{ij}^{(s)} = \begin{cases} \tan^{-1}\left|\frac{C_{ij}^{(r)}}{\lambda_{ij}\Delta_{ij}}\right| & \text{if } \mathbf{n}_{ij}\cdot\mathbf{d}_{ij} > 0, \\ 2\pi - \tan^{-1}\left|\frac{C_{ij}^{(r)}}{\lambda_{ij}\Delta_{ij}}\right| & \text{if } \mathbf{n}_{ij}\cdot\mathbf{d}_{ij} \leqslant 0, \end{cases} \tag{8}$$

where $\mathbf{d}_{ij} = \mathbf{c}_{ij} - \mathbf{r}_i$.

We assign spherical coordinates to the surface of sphere S_i by placing the poles at the points of minimum and maximum global z coordinates. After sorting, for each circle C_{ij}, an interaction list of circles that could possibly intersect C_{ij} is formed by searching for circles whose latitude–longitude bounding boxes overlap the bounding box for C_{ij}. This further reduces the total number of circle intersections to less than the nominal N_i^2.

The latitude and longitude of the circle centers are given by

$$\text{lat}(C_{ij}) = \sin^{-1}\left(\frac{c_{ij}^{(3)} - z_i}{\lambda_{ij}\Delta_{ij}}\right), \tag{9}$$

$$\text{lon}(C_{ij}) = \begin{cases} 0 & \text{if } (c_{ij}^{(1)} - x_i)^2 + (c_{ij}^{(2)} - y_i)^2 = 0, \\ \gamma_{ij} & \text{if } c_{ij}^{(1)} - x_i \geqslant 0 \quad \text{and } c_{ij}^{(2)} - y_i \neq 0, \\ 2\pi - \gamma_{ij} & \text{if } c_{ij}^{(1)} - x_i < 0 \end{cases} \tag{10}$$

where

$$\gamma_{ij} = \cos^{-1}\left(\frac{c_{ij}^{(1)} - x_i}{\sqrt{(c_{ij}^{(1)} - x_i)^2 + (c_{ij}^{(2)} - y_i)^2}}\right). \tag{11}$$

The latitude takes on values in $[0,\pi]$ where 0 represents the south pole (minimum value of z) and π is at the north pole (maximum value of z). The values for longitude are $[0,2\pi)$. If the center of a circle is located at one of the poles, then there is an ambiguity in its longitude which is removed by setting the value to zero.

The minimum and maximum values of latitude and longitude of all points on the circle C_{ij} is given by

$$\mathrm{minlat}(C_{ij}) = \max\left\{0, \mathrm{lat}(C_{ij}) + \frac{\pi}{2} - C^{(s)}_{ij}\right\}, \tag{12}$$

$$\mathrm{maxlat}(C_{ij}) = \min\left\{\pi, \mathrm{lat}(C_{ij}) + \frac{\pi}{2} + C^{(s)}_{ij}\right\}, \tag{13}$$

$$\mathrm{minlon}(C_{ij}) = \begin{cases} \mathrm{circ}(\mathrm{lon}(C_{ij}) - C^{(s)}_{ij}), & \mathrm{lat}(C_{ij}) \geqslant C^{(s)}_{ij} - \frac{\pi}{2}, \\ 0 & \text{otherwise}, \end{cases} \tag{14}$$

$$\mathrm{maxlon}(C_{ij}) = \begin{cases} \mathrm{circ}(\mathrm{lon}(C_{ij}) + C^{(s)}_{ij}), & \mathrm{lat}(C_{ij}) \leqslant \frac{\pi}{2} - C^{(s)}_{ij}, \\ 2\pi & \text{otherwise}, \end{cases} \tag{15}$$

where

$$\mathrm{circ}(\alpha) = \begin{cases} \alpha + 2\pi & \text{if } \alpha < 0, \\ \alpha & \text{if } \alpha \geqslant 0 \quad \text{and } \alpha < 2\pi, \\ \alpha - 2\pi & \text{if } \alpha > 2\pi. \end{cases} \tag{16}$$

The above formulas enforce the requirement that the circle's range of latitude is $[0,\pi/2]$ and the longitude range is $[0,2\pi]$.

Suppose C_{ik} is in the interaction list of C_{ij}. Once we compute the intersection of C_{ik} with C_{ij}, we remove C_{ij} from the interaction list of C_{ik} so that this pairwise intersection is not computed twice.

2.3. Circle intersection algorithm

For the modified arc method, each circle C_{ij} is checked for intersection with the other circles C_{ik}, accounting for the fact that these circles may not lie in the same plane. If the union of the resulting intersection arcs reconstitutes C_{ij} completely, then circle C_{ij} is covered. Alternatively, if the intersections of the complements of the intersection arcs is empty, then the circle is covered. This leads to a more efficient method for determining circle coverage, described below.

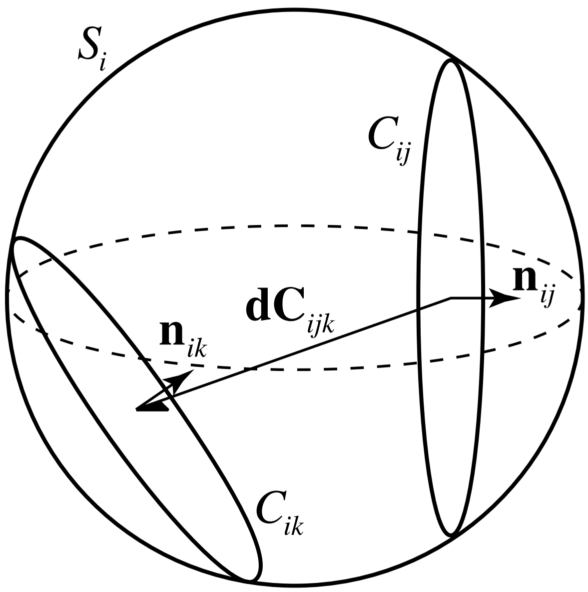

Fig. 4. How one circle can cover another circle without intersecting it.

2.3.1. Intersection of two circles – preliminaries

If $\mathbf{n}_{ik} \times \mathbf{n}_{ij} = 0$ then C_{ik} is parallel to C_{ij} and a check is done to see if C_{ik} covers C_{ij} using

$$\mathbf{dC}_{ijk} \cdot \mathbf{n}_{ik} \leqslant 0 \Rightarrow C_{ij} \text{ is covered,} \tag{17}$$

where $\mathbf{dC}_{ijk} = \mathbf{c}_{ik} - \mathbf{c}_{ij}$. Observe that this check must be done even when the circles do not intersect, as shown in Fig. 4.

If the circles are not parallel, then their planes intersect, in which case we find the line of intersection L_{ijk} of the two planes. Knowledge of L_{ijk} allows us to determine if the circles themselves intersect, and if so, what are the points of intersection. The equations of the planes of the two circles in point-direction form are:

$$\begin{aligned} &\mathbf{n}_{ij} \cdot (\mathbf{r} - \langle c_{ij}^{(1)}, c_{ij}^{(2)}, c_{ij}^{(3)} \rangle) = 0, \\ &\mathbf{n}_{ik} \cdot (\mathbf{r} - \langle c_{ik}^{(1)}, c_{ik}^{(2)}, c_{ik}^{(3)} \rangle) = 0, \end{aligned} \tag{18}$$

where $\mathbf{r} = \langle x, y, z \rangle$. A parametric equation for L_{ijk} is given by

$$\mathbf{r} = \alpha \mathbf{u}_{ijk} + \mathbf{p}_{ijk}, \tag{19}$$

where α is a scalar parameter, $\mathbf{u}_{ijk}$ is a direction vector, and $\mathbf{p}_{ijk}$ is a point on the line L_{ijk}. Observe that $\mathbf{u}_{ijk} = \mathbf{n}_{ij} \times \mathbf{n}_{ik}$.

The solution for $\mathbf{p}_{ijk}$ can be found by two methods. The first is by determining where L_{ijk} intersects one of the global principal coordinate planes $x = 0$, $y = 0$, or $z = 0$. The second is by intersecting the planes containing the centers of sphere S_i and the circles C_{ij} and C_{ik}. It turns out the first method is faster but more complicated to code, while the second method is slower but simpler to code. We include a discussion of both methods here and leave the choice to the reader. Each method for $\mathbf{p}_{ijk}$ has its own associated method for finding the intersection points with the two circles, which we also detail. The goal of the next two subsections is to find angles α_1 and α_2 in circle C_{ij}'s angle coordinates which represent the points of intersection of circles C_{ij} and C_{ik}.

If the circles intersect at less than two points, then we must check if circle C_{ik} covers C_{ij} by virtue of its 3D orientation in space using relation (17).

The computations of the next two subsections are described with reference to circle C_{ij}, but also apply to circle C_{ik} with no substantive changes.

2.3.2. Intersection of two circles – method 1

Throughout this section, we will define numerous intermediate quantities which should have subscripts of *ij*, *ik* or *ijk*, but which we will eliminate for clarity. Any code implementation of these ideas must be cognizant of these dependencies.

If we let

$$\langle a_1, b_1, c_1 \rangle = \mathbf{n}_{ij}, \tag{20}$$

$$d_1 = -\mathbf{n}_{ij} \cdot \mathbf{c}_{ij}, \tag{21}$$

then from Eq. (18) an equation for the plane of circle C_{ij} is

$$\langle a_1, b_1, c_1 \rangle \cdot \mathbf{r} + d_1 = 0. \tag{22}$$

Similarly, if we let

$$\langle a_2, b_2, c_2 \rangle = \mathbf{n}_{ik}, \tag{23}$$

$$d_2 = -\mathbf{n}_{ik} \cdot \mathbf{c}_{ik}. \tag{24}$$

then an equation for the plane of circle C_{ik} is

$$\langle a_2, b_2, c_2 \rangle \cdot \mathbf{r} + d_2 = 0. \tag{25}$$

The following pseudo-code describes how to find the point $\mathbf{p}_{ijk}$ where L_{ijk} crosses one of the global principal coordinate planes $x = 0$, $y = 0$, or $z = 0$.

if $u_z = 0$ (L_{ijk} does not cross the X–Y plane), **then**
 if $u_x = 0$ (L_{ijk} does not cross Y–Z plane), **then**
  $p_x = (c_2 d_1 - d_2 c_1)/(a_2 c_1 - c_2 a_1)$,
  $p_y = 0$,
  $p_z = (a_1 d_2 - d_1 a_2)/(a_2 c_1 - c_2 a_1)$,
 else (L_{ijk} crosses Y–Z plane)
  $p_x = 0$,
  $p_x = (c_2 d_1 - d_2 c_1)/(-c_2 b_1 + b_2 c_1)$,
  $p_z = (-b_2 d_1 + d_2 b_1)/(-c_2 b_1 + b_2 c_1)$,
else (L_{ijk} crosses X–Y plane)
 $p_x = (b_2 d_1 - d_2 b_1)/(-b_2 a_1 + a_2 b_1)$,
 $p_y = (a_1 d_2 - d_1 a_2)/(-b_2 a_1 + a_2 b_1)$,
 $p_z = 0$.

To determine the points of intersection, we change coordinates from 3D global x, y and z to 2D coordinates x and y local to circle C_{ij}. Then, we solve the 2D problem of finding the intersection between a circle and line in a plane. We make the following definitions:

$$m_x = \mathbf{u}_{ijk} \cdot \hat{\mathbf{x}}_{ij}, \tag{26}$$

$$m_y = \mathbf{u}_{ijk} \cdot \hat{\mathbf{y}}_{ij}, \tag{27}$$

$$\mathbf{r}_1 = \mathbf{p}_{ijk} - \mathbf{c}_{ij}, \tag{28}$$

$$x_0 = \mathbf{r}_1 \cdot \hat{\mathbf{x}}_{ij}, \tag{29}$$

$$y_0 = \mathbf{r}_1 \cdot \hat{\mathbf{y}}_{ij}. \tag{30}$$

If $m_x = 0$, define $a = 1$, $b = 0$, and $c = x_0$, otherwise define $a = -m_x/m_y$, $b = 1$, and $c = y_0 + ax$. The equation of the projected line thus takes the canonical form $ax + by = c$. The equation of the circle C_{ij} in local coordinates is simply $x^2 + y^2 = r^2$, where $r = C_{ij}^{(r)}$ is the radius.

There are six possible cases for the intersection of the line L_{ijk} and circle C_{ij}:

(1) $b = 0$ and $r^2 - \frac{c^2}{a^2} < 0 \Rightarrow$ no intersection.
(2) $b = 0$ and $r^2 - \frac{c^2}{a^2} = 0 \Rightarrow$ vertical line, one point of intersection:

$$x_1 = \frac{c}{a}, \quad y_1 = 0. \tag{31}$$

(3) $b = 0$ and $r^2 - \frac{c^2}{a^2} = 0 \Rightarrow$ vertical line, two points of intersection:

$$x_1 = \frac{c}{a}, \quad y_1 = \sqrt{r^2 - \frac{c^2}{a^2}}, \tag{32}$$

$$x_2 = \frac{c}{a}, \quad y_2 = -\sqrt{r^2 - \frac{c^2}{a^2}}. \tag{33}$$

(4) $b \neq 0$ and $b^4 r^2 - b^2 c^2 + a^2 r^2 b^2 < 0 \Rightarrow$ no intersection.
(5) $b \neq 0$ and $b^4 r^2 - b^2 c^2 + a^2 r^2 b^2 = 0 \Rightarrow$ not a vertical line, one point of intersection:

$$x_1 = \frac{ac}{a^2 + b^2}, \quad y_1 = \frac{-a x_1 + c}{b}. \tag{34}$$

(6) $b \neq 0$ and $b^4 r^2 - b^2 c^2 + a^2 r^2 b^2 > 0 \Rightarrow$ not a vertical line, two points of intersection:

$$x_1 = \frac{ac + \sqrt{b^4 r^2 - b^2 c^2 + a^2 r^2 b^2}}{a^2 + b^2}, \quad y_1 = \frac{-a x_1 + c}{b}, \tag{35}$$

$$x_2 = \frac{ac - \sqrt{b^4 r^2 - b^2 c^2 + a^2 r^2 b^2}}{a^2 + b^2}, \quad y_2 = \frac{-a x_2 + c}{b}. \tag{36}$$

If there is an intersection, the solutions for x_1, y_1, x_2, and y_2 given above for the points of intersection are converted into angles:

$$\alpha_1 = \begin{cases} \cos^{-1}\left(\frac{x_1}{r}\right) & \text{if } y_1 \geqslant 0, \\ 2\pi - \cos^{-1}\left(\frac{x_1}{r}\right) & \text{if } y_1 < 0, \end{cases} \tag{37}$$

$$\alpha_2 = \begin{cases} \cos^{-1}\left(\frac{x_2}{r}\right) & \text{if } y_2 \geqslant 0, \\ 2\pi - \cos^{-1}\left(\frac{x_2}{r}\right) & \text{if } y_2 < 0. \end{cases} \tag{38}$$

If $\alpha_1 > \alpha_2$, then we swap α_1 and α_2. This ordering is required for the ambiguity resolution algorithm described in Section 2.3.4.

2.3.3. Intersection of two circles – method 2

Consider the plane containing the center of the sphere S_i and the centers of the circles C_{ij} and C_{ik}. Its intersection with sphere S_i is portrayed by the large circle in Fig. 5 denoted by C_{ijk}. Circle C_{ijk} and line L_{ijk} intersect at a single point, which in this method constitutes our solution for $\mathbf{p}_{ijk}$. Fig. 6 shows a 2D layout of circle C_{ijk}, with circles C_{ij} and C_{jk} shown edge-on as line segments. We make some preliminary definitions for circle C_{ijk} as follows:

$$\mathbf{d}_{ij} = \mathbf{c}_{ij} - \mathbf{r}_i, \tag{39}$$

$$\mathbf{d}_{ik} = \mathbf{c}_{ik} - \mathbf{r}_i, \tag{40}$$

$$\mathbf{n}_{ijk} = \frac{\mathbf{d}_{ij} \times \mathbf{d}_{ik}}{\|\mathbf{d}_{ij} \times \mathbf{d}_{ik}\|}, \tag{41}$$

$$\hat{\mathbf{x}}_{ijk} = \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|}, \tag{42}$$

$$\hat{\mathbf{y}}_{ijk} = \mathbf{n}_{ijk} \times \hat{\mathbf{x}}_{ijk}. \tag{43}$$

From Fig. 6 the point of intersection $\langle x, y\rangle$ in coordinates local to circle C_{ij} can be computed by setting $\langle x_1, y_1\rangle = (\mathbf{d}_{ik} \cdot \hat{\mathbf{x}}_{ij}, \mathbf{d}_{ik} \cdot \hat{\mathbf{y}}_{ij})$ and

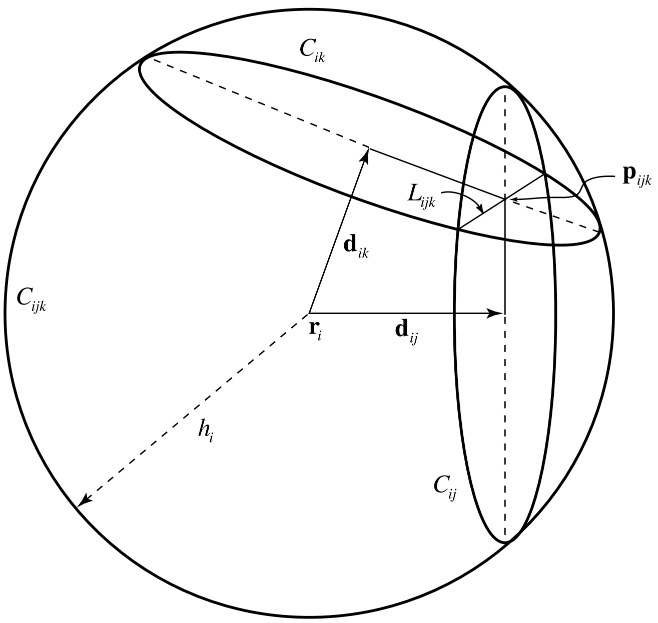

Fig. 5. Nomenclature for circle intersection method 2. The point $\mathbf{p}_{ijk}$ is the intersection of the planes of circles C_{ijk}, C_{ij} and C_{ik}.

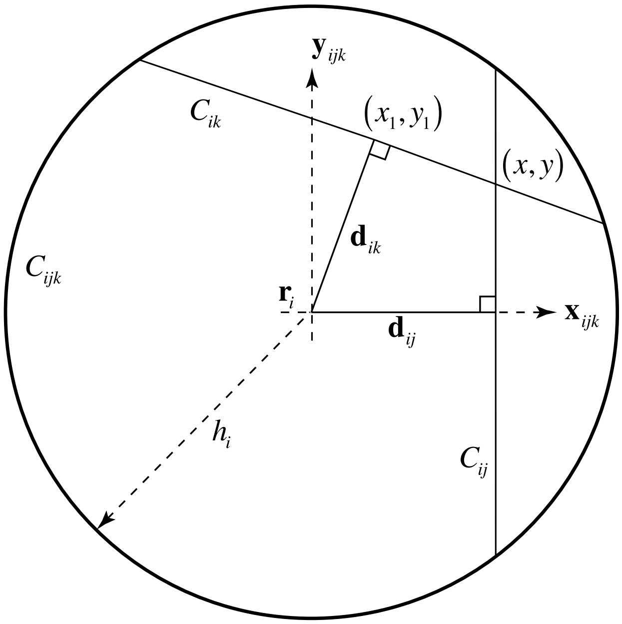

Fig. 6. Finding the intersection point for circle intersection method 2 in the plane of circle C_{ijk}.

$$x = \|\mathbf{d}_{ij}\|, \tag{44}$$
$$y = \frac{-x_1}{y_1}(x - x_1) + y_1. \tag{45}$$

We assume y_1 is non-zero above because the circles C_{ij} and C_{ik} are assumed not to be parallel. This point of intersection is then expressed in global coordinates by

$$\mathbf{p}_{ijk} = x\hat{\mathbf{x}}_{ijk} + y\hat{\mathbf{y}}_{ijk} + \mathbf{r}_i. \tag{46}$$

We find its local coordinates in the plane of circle C_{ij} using

$$\mathbf{d}_{ijk} = \mathbf{p}_{ijk} - \mathbf{c}_{ij}, \tag{47}$$
$$\mathbf{p}_{ij} = \langle x, y \rangle = \langle \mathbf{d}_{ijk} \cdot \hat{\mathbf{x}}_{ij}, \mathbf{d}_{ijk} \cdot \hat{\mathbf{y}}_{ij} \rangle. \tag{48}$$

To find the angles of the intersection points, consider the plane of circle C_{ij}, shown in Fig. 7. The angles ω and θ in the diagram are:

$$\omega = \cos^{-1}\left(\frac{r}{C_{ij}^{(r)}}\right), \tag{49}$$
$$\theta = \begin{cases} \cos^{-1}\left(\frac{x}{r}\right) & \text{if } y \geqslant 0, \\ 2\pi - \cos^{-1}\left(\frac{x}{r}\right) & \text{if } y < 0, \end{cases} \tag{50}$$

where $r = \sqrt{x^2 + y^2}$. The angles of intersection are then just:

$$\alpha_1 = \theta - \omega, \quad \alpha_2 = \theta + \omega. \tag{51}$$

The derivation above cannot be used if the line L_{ijk} passes through the center of circle C_{ij}. In this exceptional case, the direction of the line of intersection is determined and the points of intersection with circle C_{ij} are computed with the aid of Fig. 8:

$$\mathbf{u} = \langle u_x, u_y \rangle = \langle \mathbf{u}_{ijk} \cdot \hat{\mathbf{x}}_{ij}, \mathbf{u}_{ijk} \cdot \hat{\mathbf{y}}_{ij} \rangle, \tag{52}$$
$$\alpha_1 = \begin{cases} \cos^{-1}\left(\frac{u_x}{\|\mathbf{u}\|}\right) & \text{if } u_y \geqslant 0, \\ 2\pi - \cos^{-1}\left(\frac{u_x}{\|\mathbf{u}\|}\right) & \text{if } u_y < 0, \end{cases} \tag{53}$$
$$\alpha_2 = \mathrm{mod}(\alpha_1 + \pi, 2\pi). \tag{54}$$

We order the angles of intersection as in method 1: **if** $\alpha_1 > \alpha_2$, **then** swap α_1 and α_2.

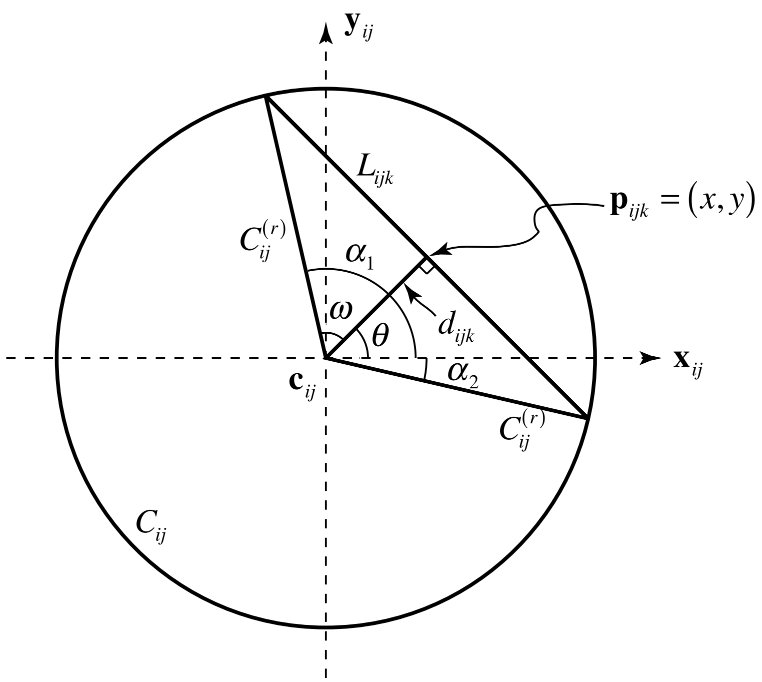

Fig. 7. Finding α_1 and α_2 in the plane of circle C_{ij} for circle intersection method 2.

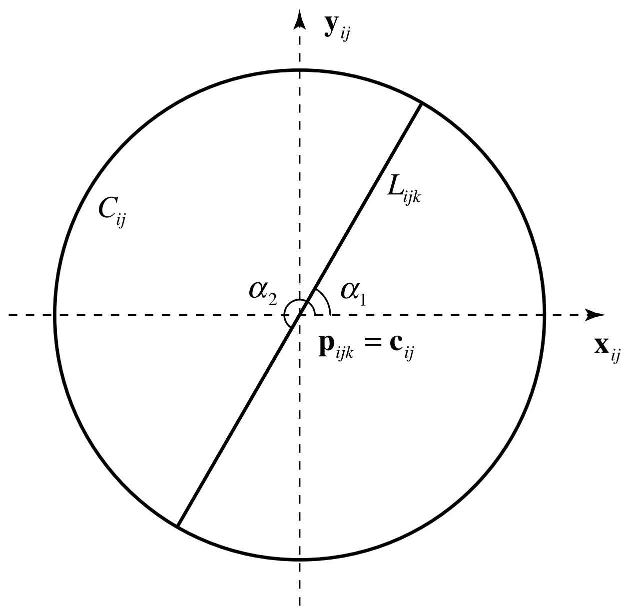

Fig. 8. Definition of α_1 and α_2 when L_{ijk} passes through the center of circle C_{ij}.

2.3.4. Resolving ambiguity in circle intersections

If there are two points of intersection for C_{ij} and C_{ik} we must determine which portion of circle C_{ij} is covered by projecting the normal vector of C_{ik} into the plane of circle C_{ij} (Fig. 9). This projection points to the part of circle C_{ij} that is covered and is given by:

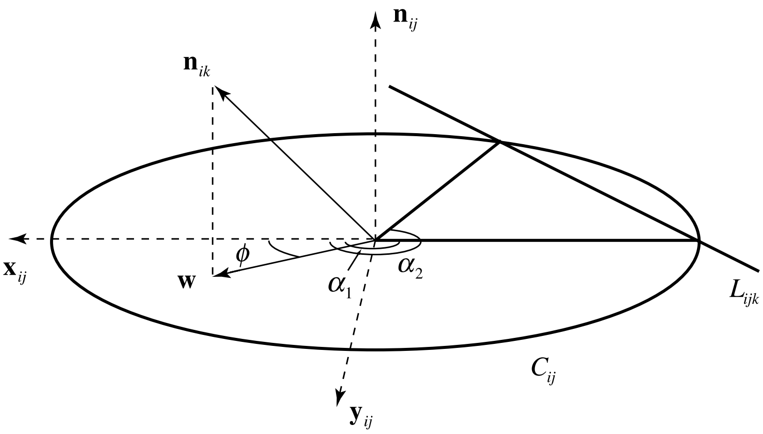

Fig. 9. Line L_{ijk} divides circle C_{ij} into two parts. The projection $\mathbf{w}$ of $\mathbf{n}_{ik}$ into the plane of circle C_{ij} points to the covered part.

$$\mathbf{w} = \langle w_x, w_y \rangle = \langle \mathbf{n}_{ik} \cdot \hat{\mathbf{x}}_{ij}, \mathbf{n}_{ik} \cdot \hat{\mathbf{y}}_{ij} \rangle. \tag{55}$$

The angle of the projected normal vector of C_{ik} is

$$\phi = \begin{cases} \cos^{-1}\left(\frac{w_x}{\|\mathbf{w}\|}\right) & \text{if } w_y \geqslant 0, \\ 2\pi - \cos^{-1}\left(\frac{w_x}{\|\mathbf{w}\|}\right) & \text{if } w_y < 0. \end{cases} \tag{56}$$

Let A_{ijk} denote the arc on circle C_{ij} due to its intersection with C_{ik}. This arc is assigned an interval on the real line according to

$$A_{ijk} = \begin{cases} [\alpha_1, \alpha_2] & \text{if } \phi \in [\alpha_1, \alpha_2], \\ [\alpha_2, \alpha_1 + 2\pi] & \text{if } \phi \notin [\alpha_1, \alpha_2]. \end{cases} \tag{57}$$

2.3.5. Checking coverage of a circle

The circle C_{ij} is covered by its arcs if:

$$[0, 2\pi) \subset \bigcup_k A_{ijk}. \tag{58}$$

If all the arcs are computed beforehand, an algorithm for making this determination is given in [6] which uses the quicksort algorithm to order the arcs by their left endpoint. The resulting sorted list of arcs is then checked for gaps between right endpoints and left endpoints.

We provide here an alternate method of determining coverage using a dynamic linked list that represents the *complement* of union of the set of arcs known at any point in time. Thus we can stop mapping arcs to a circle when we know that the circle is fully covered, and it is not necessary to compute all arcs beforehand. The linked list is initialized to the whole interval $\mathcal{C} = [0, 2\pi)$, and when it is empty, the circle is covered. As the process of finding circle intersections and mapping arcs is time-consuming, the linked-list technique for arc complements can provide substantial savings in CPU resources. Mathematically, the linked list represents the intersection of the complements of the arc intervals:

$$\mathcal{L}_{ij} = \bigcap_k (\mathcal{C} - \widetilde{A}_{ijk}), \tag{59}$$

where $\widetilde{A}_{ijk}$ is the interval A_{ijk} adjusted to fit inside $\mathcal{C}$:

$$A_{ijk} = [\alpha_1, \alpha_2], \tag{60}$$

$$\widetilde{A}_{ijk} = \begin{cases} A_{ijk} & \text{if } \alpha_2 < 2\pi, \\ [0, \alpha_2 - 2\pi] \cup [\alpha_1, 2\pi) & \text{if } \alpha_2 \geqslant 2\pi. \end{cases} \tag{61}$$

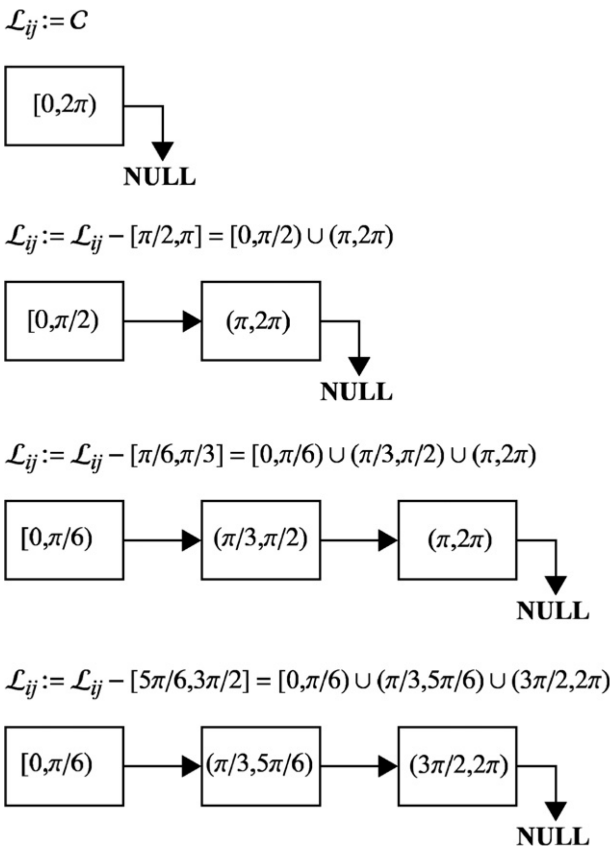

Fig. 10. Example of updating linked list to remove arcs of intersection.

Mapping arcs to the circle C_{ij} is thus described by successive intersections in Eq. (59):

$$\mathcal{L}_{ij} := \mathcal{L}_{ij} \cap (\mathcal{C} - \widetilde{A}_{ijk}) = \mathcal{L}_{ij} - \widetilde{A}_{ijk}. \tag{62}$$

From this we see that an update for a new arc will consist of "cutting out" some intervals and parts of intervals contained in the previous linked list iteration. An example of updating the list is given in Fig. 10. Observe that the linked list is always ordered, in the sense that it always "points to the right". The condition that circle C_{ij} is covered by arcs (Eq. (58)) is that the linked list be empty:

$$\mathcal{L}_{ij} = \emptyset \iff C_{ij} \text{ is covered.} \tag{63}$$

Each update to the linked list requires finding intervals within it that contain the left and right endpoints of the new arc. A naive search procedure for finding these intervals would produce a running time of $O(N^2)$, where N is the number of arcs. A binary search procedure would reduce this to $O(N\log N)$, which is the same as for the quicksort method of [6]. In this case we expect on average that the linked list method will be faster than the quicksort method since the linked list method terminates as soon as the list becomes empty.

3. Examples and timing

The above boundary detection algorithm was implemented in the SPHINX hydrodynamic code [8] and tested on a cube (Fig. 11), a sphere (Fig. 12), and two cylinders (Fig. 13). Each problem was run for five time

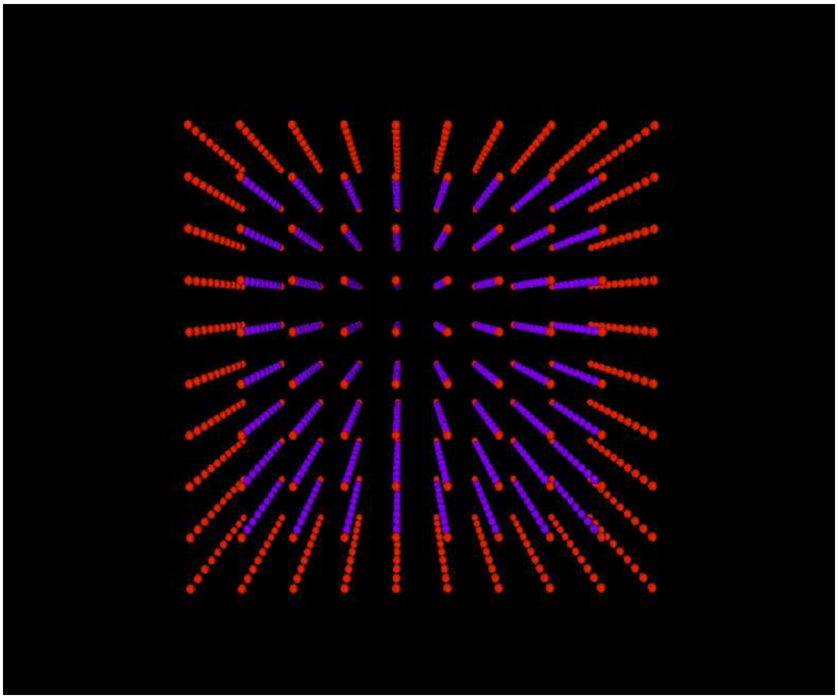

Fig. 11. Results of boundary detection on a cube. Red points are on boundary, blue are in interior. Spacing was $1 - h$. Only center points are shown. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

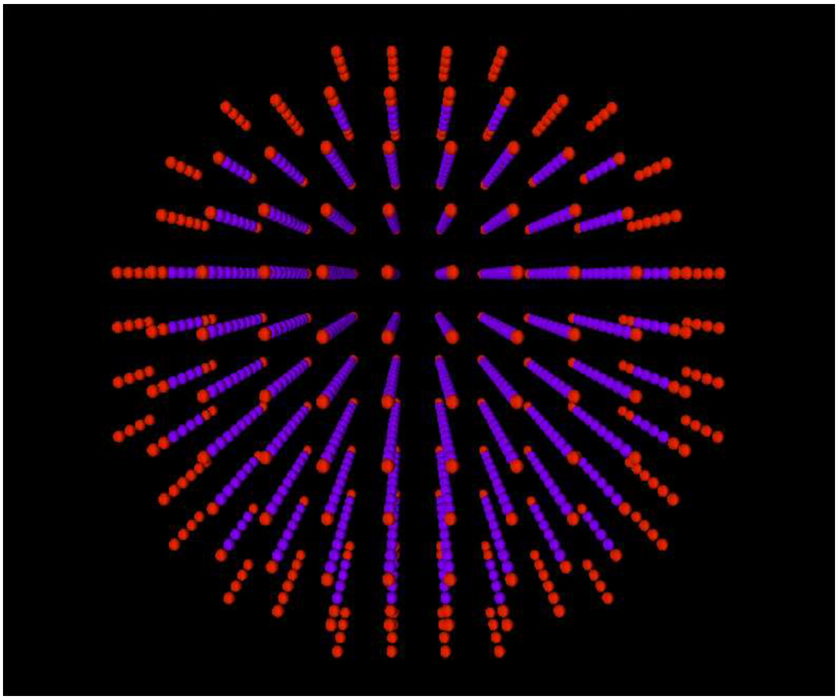

Fig. 12. Results of boundary detection on a sphere. Red points are on boundary, blue are in interior. Spacing was $1 - h$. Only center points are shown. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

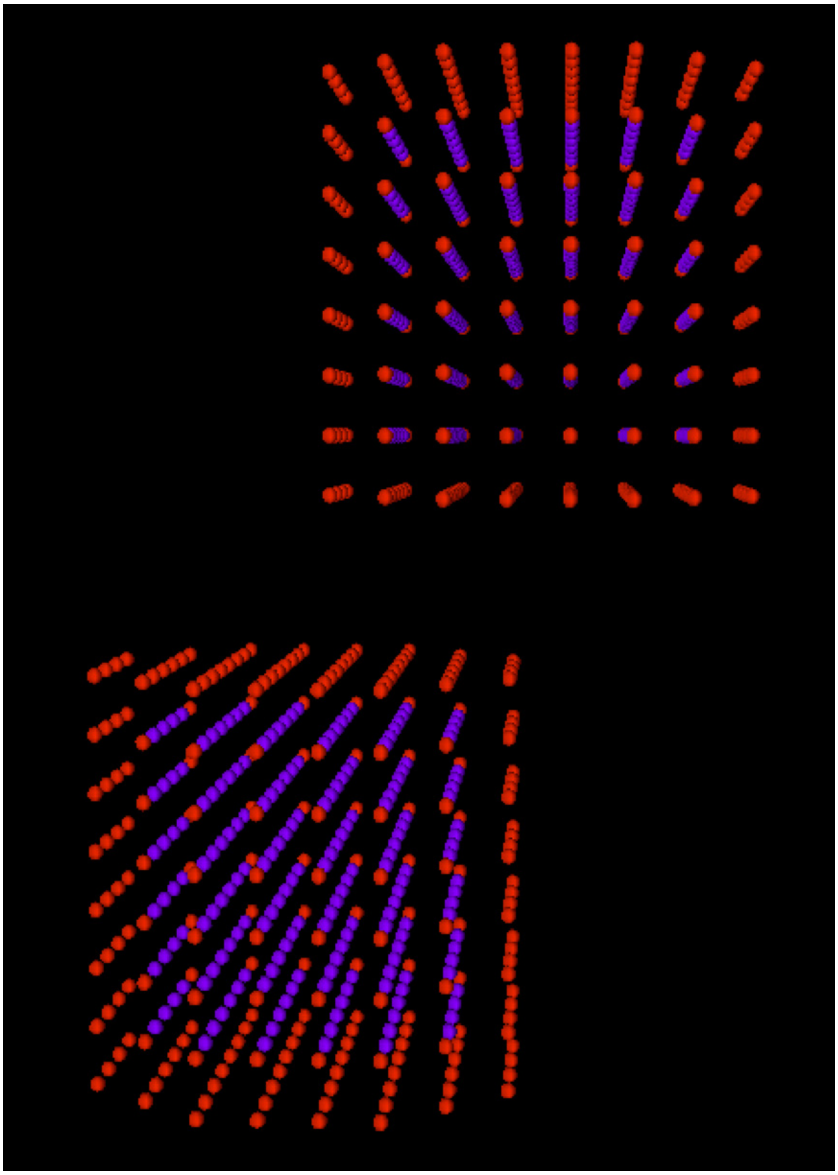

Fig. 13. Results of boundary detection on two cylinders. Red points are on boundary, blue are in interior. Spacing was $1-h$. Only center points are shown. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

steps in order to average out the timings for each application of the algorithm. The three-dimensional exposure method correctly determines the boundary in all cases.

An upper bound for run time of the algorithm is $O(MN^2)$, where M is the total number of particles and N is the average number of neighbors per particle, and is less favorable than $O(MN\log N)$ obtained for two dimensions. The plot in Fig. 14 confirms that the algorithm is linear in the total number of particles. The two cylinders test case was used, increasing the number of particles while keeping the number of neighbors per particle approximately the same. The run time was averaged over five time steps.

In Fig. 15 the average time finding the boundary for each particle is plotted against the average number of neighbors per particle. The run times vary with the shape of the object, due to the different surface to volume

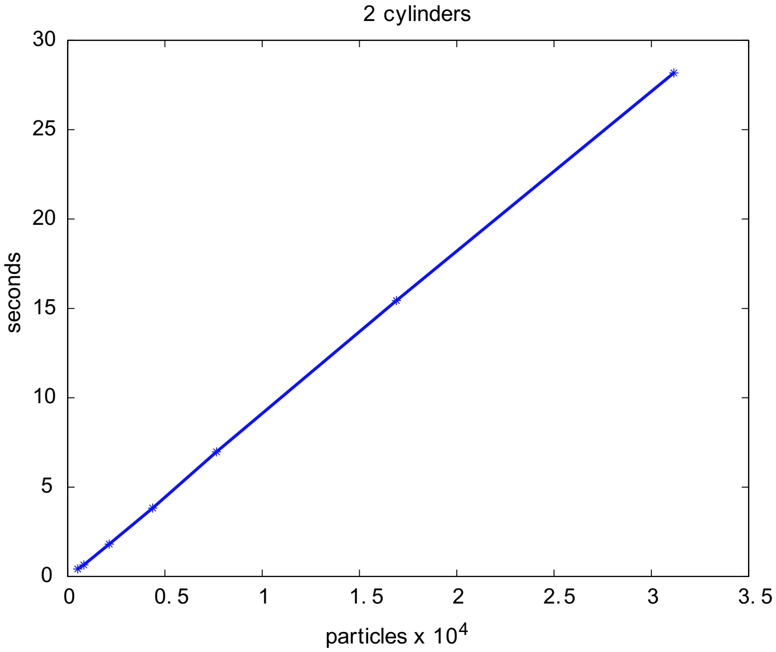

Fig. 14. Plot of run time vs. number of particles with number of neighbors held constant.

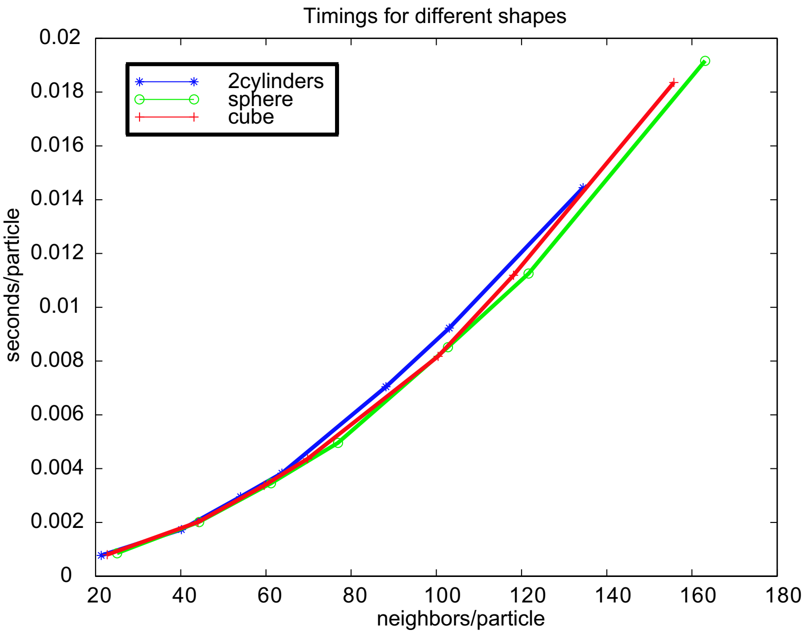

Fig. 15. Run time for different shapes.

Table 1
Exponents

Shape	λ ($\times 10^{-6}$)	ϵ
2 cylinders	4.81	1.62
Sphere	3.77	1.67
Cube	4.33	1.64

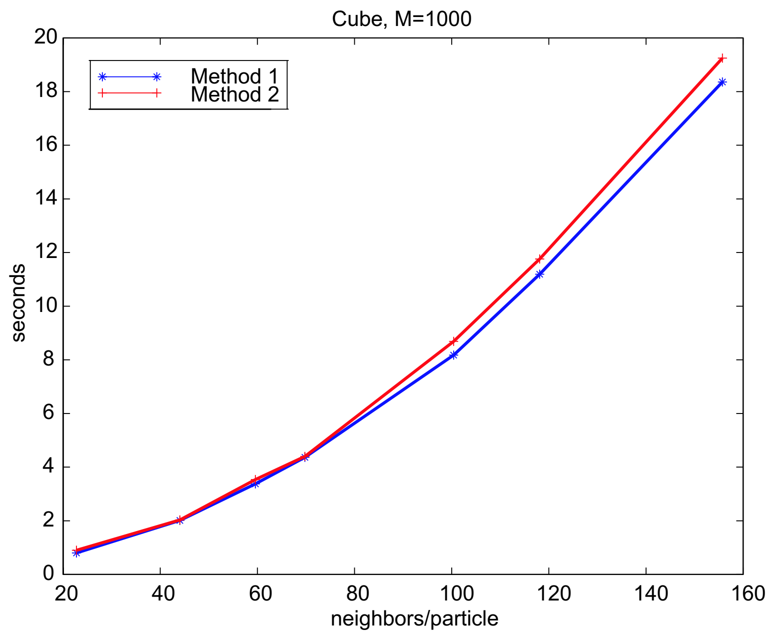

Fig. 16. Circle intersection method 1 vs. method 2.

ratios of the shapes. The different numbers of neighbors were obtained by varying the smoothing length. The run times were averaged over the number of particles because the different shapes contain different numbers of particles and are modeled by λN^{ϵ}, where N is the number of neighbors and λ and ϵ are constants. Table 1 gives a least squares fit of the data in Fig. 15 for λ and ϵ. The observed values of $\epsilon \in [1.6, 1.7]$ are better than the predicted value of $\epsilon = 2$, due probably to the measures introduced in Sections 2.2 and 2.3.5 for reducing the number of circle intersections computed.

The run times for the two methods of computing the intersections of circles described in Sections 2.3.2 and 2.3.3 are compared in Fig. 16. Method 1 is asymptotically slightly faster than method 2. The runs used 1000 particles while varying the average number of neighbors per particle and again averaging over 5 time steps.

A time sequence from a ball and plate impact simulation similar to that presented in Ref. [6] is shown in point-cloud representation in Fig. 17. The red points are boundary points selected by the 3D exposure method and the blue points are interior. In Fig. 18 a cut-away of the set of spheres of radius 0.5-h from a single time-step in this simulation is shown. The ability to dynamically detect void opening and closure is shared with that demonstrated by the two-dimensional method of Ref. [6].

The technique for computing geometric boundary normals given in Ref. [6] also applies to the present three-dimensional algorithm. The geometric normal is given by

$$\mathbf{n}(\mathbf{x}_i) = -\sum_{j \notin \mathcal{B}} \nabla \phi_j^S(\mathbf{x}_i), \quad i \in \mathcal{B} \tag{64}$$

$$\phi_i^S(\mathbf{x}) = W_i(\mathbf{x}) \Big/ \sum_{j \notin \mathcal{B}} W_j(\mathbf{x}), \tag{65}$$

where $\mathcal{B}$ is the set of all boundary particle indices and W_i is the smoothing kernel centered at particle i. Fig. 19 shows a time step of the same ball-and-plate simulation rendered in a surface representation where a disk is drawn perpendicular to the above normal and displaced from the center of the sphere a distance of 0.5 h. Interior particles are not shown.

The time to make the boundary identification using the new 3D exposure method in these

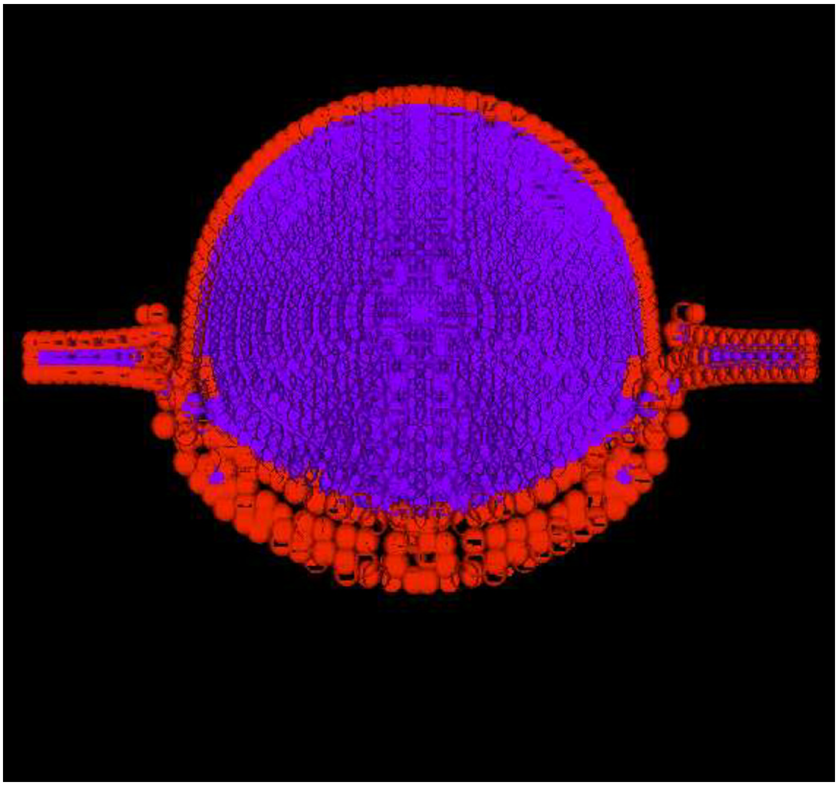

Fig. 18. Cut-away of 3D ball-on-plate impact simulation.

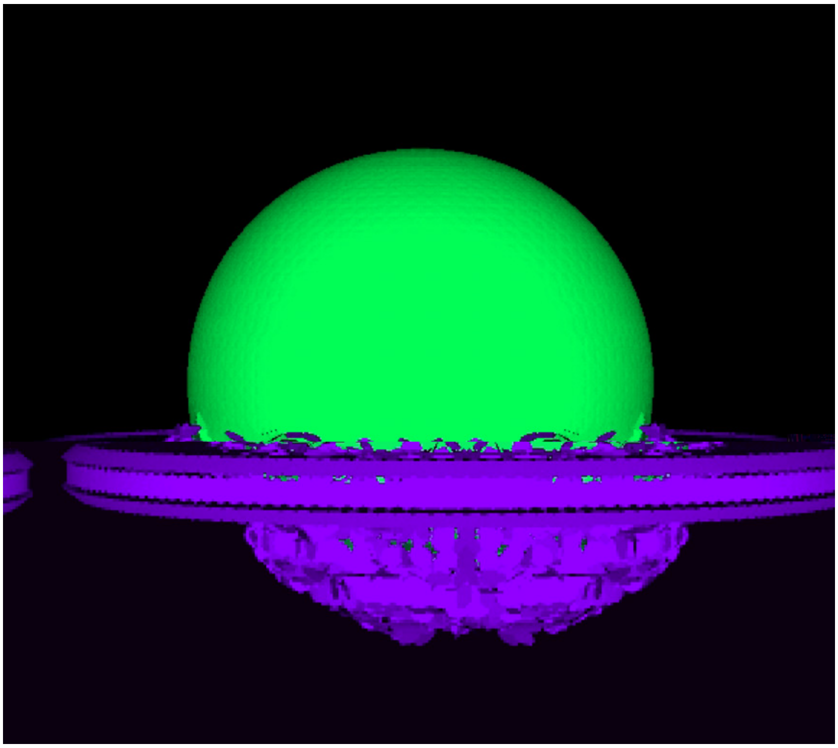

Fig. 19. Representation of boundary normals by hexagonal disks.

Acknowledgement

This work was supported by the U.S. Department of Energy under contract W-7405-ENG-36.

References

[1] L. Libersky, A. Petschek, T. Carney, J. Hipp, F. Allahdadi, High strain lagrangian hydrodynamics, J. Comp. Phys. 109 (109) (1993) 67–75.
[2] P. Randles, L. Libersky, Smoothed particle hydrodynamics: Some recent improvements and applications, Comp. Meth. Appl. Mech. Eng. 139 (1996) 375–408.
[3] T. Belytschko, Y. Lu, L. Gu, Element-free Galerkin methods, Int. J. Numer. Meth. Eng. 37 (1994) 229–256.
[4] Y.F.Z.W.K. Liu, S. Jun, Reproducing kernel particle methods, Int. J. Numer. Meth. Eng. 20 (8-9) (1995) 1081–1106.
[5] G.A. Dilts, Moving-least-squares-particle hydrodynamics I, consistency and stability, Int. J. Numer. Meth. Eng. 44 (8) (1999) 1115–1155.
[6] G.A. Dilts, Moving-least-squares-particle hydrodynamics II: conservation and boundaries, Int. J. Numer. Meth. Eng. 48 (10) (2000) 1503–1524.
[7] T. Iwai, C. Hong, P. Greil, Fast particle pair detection algorithms for particle simulations, Int. J. Modern Phys. C 10 (5) (1999) 823.
[8] L.A. Crotzer, G.A. Dilts, C.E. Knapp, K.D. Morris, R.P. Swift, C.A. Wingate, Sphinx Manual Version 11.0 Tech. Rep. LA-13436-M, Los Alamos National Laboratory, 1998.