LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-214271

Surface detection, meshing and analysis during large molecular dynamics simulations

L. M. Dupuy, R. E. Rudd

August 3, 2005

Modelling and Simulations in Materials Science and Engineering

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Surface detection, meshing and analysis during large molecular dynamics simulations

Laurent M. Dupuy and Robert E. Rudd

Lawrence Livermore National Laboratory, University of California, L-415, Livermore, CA 94551 USA

E-mail: dupuy2@llnl.gov, robert.rudd@llnl.gov

Abstract. New techniques are presented for the detection and analysis of surfaces and interfaces in atomistic simulations of solids. Atomistic and other particle-based simulations have no inherent notion of a surface, only atomic positions and interactions. The algorithms we introduce here provide an unambiguous means to determine which atoms constitute the surface, and the list of surface atoms and a tessellation (meshing) of the surface are determined simultaneously. The algorithms have been implemented and demonstrated to run automatically (on the fly) in a large-scale parallel molecular dynamics (MD) code on a supercomputer. We demonstrate the validity of the method in three applications in which the surfaces and interfaces evolve: void surfaces in ductile fracture, the surface morphology due to significant plastic deformation of a nanoscale metal plate, and the interfaces (grain boundaries) and void surfaces in a nanoscale polycrystalline system undergoing ductile failure. The technique is found to be quite robust, even when the topology of the surfaces changes as in the case of void coalescence where two surfaces merge into one. It is found to add negligible computational overhead to an MD code, and is much less expensive than other techniques such as the solvent-accessible surface.

Keywords: surface detection, surface meshing, atomistic simulation, molecular dynamics

Submitted to: *Modelling Simulation Mater. Sci. Eng.*

1. Introduction

Surfaces and interfaces play an important role in many processes in Physics, Chemistry and Materials Science [1]. Surfaces are extremely important in nanoscience because of the relatively large surface-area-to-volume ratio in these ultra-small systems. They are essential to passivation and functionalization in which surfaces are modified to eliminate unwanted activity and add specific kinds of activity, respectively. They are also important to catalytic activity of nanoparticles, the binding of protein-ligand complexes, and the filtering properties of nanoporous materials, just to name a few of the many nanoscale processes of current interest in which surfaces play an essential role. Interfaces, too, are the subject of intensive study. These interfaces include grain boundaries in which two different orientations of the same crystalline material join, epitaxial interfaces in which dissimilar materials come together in a way commensurate with their crystal structures, other thin film/substrate interfaces, phase boundaries, and even cell walls. Interfaces are central to such currently hot topics such as spintronics, nanostructured materials, and the bio-nano interface. Atomistic simulation of these systems often provides information crucial to the understanding of their behavior, from synthesis to functioning to failure. At each of these stages, the surfaces and interfaces may evolve. The morphological evolution of surfaces during epitaxial deposition culminating in the self-assembly of quantum dots is an excellent example. Similarly, the dynamics of grain boundaries during the deformation of nanostructured materials through dislocation emission and grain boundary sliding is another case that has attracted attention recently.

In our work, voids associated with ductile fracture are of interest, and here too, the dynamical evolution of the void surface is the focus of attention. The tracking and analysis of surfaces and interfaces in atomistic simulation is non-trivial. It is desirable to have a technique to identify the atoms in a surface or interface quickly, and to tessellate the surface with triangles and/or higher order polygons. This tessellation is the first step in many analyses, and the ability to compute a tessellation quickly, automatically and robustly on a parallel supercomputer opens the door to many kinds of analysis on the fly, yielding much more precise results without the need to write extensive data to the disk for post-processing. It also facilitates comparison with continuum calculations are larger length scales, paving the way for more robust hierarchical multiscale models.

In this article we present a new algorithm to detect and analyze surfaces during MD simulations. Based on a simple yet robust geometrical construction, it locates atoms on a surface and meshes the surface simultaneously, thus expediting its analysis. This method has several desirable qualities. (i) It introduces an unambiguous geometrical criterion for surface atoms; (ii) Surface detection and tessellation are performed at the same time; (iii) The analysis can be performed automatically, on the fly during the course of a simulation; (iv) The required computations are fast, and the method takes advantage of commonly used MD data structures as well as parallelism, thereby adding no significant computational overhead; (v) It is suitable for convoluted surfaces (non-convex, etc.). Here we develop our algorithm for monatomic simulations, but it could

be extended to simulations with more than one kind of atom.

The paper is organized as follows. In section 2 we review existing surface detection and surface meshing algorithms. In section 3 we discuss the principle of our algorithm. Details of the algorithm in single-processor simulations and its extension to simulations with multiple processors in parallel are presented in section 4. Section 5 includes three different applications of this method. The first example concentrates on void analysis during dynamic fracture of single crystal of a ductile metal with an emphasis on surface, volume and shape analysis of the voids. The second example shows a surface analysis of a nanoscale plate (an unsupported thin film) under stress. The third application demonstrates how this method can be extended to analyze not only surfaces but also interfaces in atomistic simulations through the example of dynamic fracture of polycrystalline copper.

2. Analysis techniques: A brief overview

As discussed above, extracting the surface atoms and meshing the surface are the two essential components of surface analysis. First we briefly describe some of the existing techniques to detect these atoms. Second we review the existing meshing techniques.

One of the simplest methods to analyze surfaces is to look for regions with low densities. For example, a system can be partitioned quickly into boxes, and surfaces can be identified with the transition from empty boxes to boxes with full atomic density [2]. This technique provides a rather crude identification of the surface atoms and the properties of the surfaces such as the area. Often the noise in this kind of analysis is sufficiently large to obscure the signal.

Other approaches are based on the fact that atoms in solids tend to get organized in well-defined structures. Only a few of them are involved in topological defects (including surfaces). These defects are key to understanding the deformation of crystalline materials, and for this reason several analysis techniques have been developed to extract the atoms associated with defects from the considerably larger total number of atoms in the system. Generally all of the atoms are considered sequentially and their local configuration is analyzed. One of the simplest methods is based on the potential energy of each atom, where relatively high energy is associated with defects including surfaces [3, 4, 5]. This method is computationally inexpensive since classical MD codes are typically set up to calculate the energy of each atom at each time step already. Other methods are based on geometrical measures of the local configuration. Among them, the centrosymmetry deviation method [6] takes advantage of the fact that in many common crystallographic structures each atom is a center of inversion symmetry for its neighbors; i.e. for any neighbor, there is another neighbor such that the atom is exactly half-way between the two. Defects break this symmetry, and the deviation is easy to quantify. The bond-orientational order method [7] is based on the orientation of the atoms with respect to each other. Other techniques include the common-neighbor analysis [8] or the medium-range-order analysis [9]. It should be noted that these methods require an

analysis of all of the atoms in the system, and they do not provide a mesh. Also, they cannot by themselves differentiate between atoms belonging to separate surfaces.

Another surface detection technique is a generalization of ray tracing. The technique searches outward from a central point (or points) for the closest atom within a solid angle element [10]. The technique only searches a subset of the atoms in the system, so it is fast, but it is not well suited for convoluted surfaces.

Meshing techniques have been developed for a large variety of applications. Powerful techniques were developed for the purpose of finite-element calculations as the accuracy of results depends directly on the quality of the mesh [11]. These methods include both 2D [12, 13, 14] and 3D [15, 16] algorithms for surfaces. It should however be noted that these methods generally require the knowledge of a parametric equation describing the surface, something not available in our case. Among others, the 2D Delaunay method is a well-defined technique that constructs an optimum mesh from a given collection of points. By projecting the positions (3D) of the pre-detected surface atoms from a given point onto a plane, Seppälä et al. [10] were able to use the Delaunay method to obtain a 3D mesh. The applicability of this method is limited by the need for a minimally distorting local projection to a plane. The extension of Delaunay triangulation to non-planar surfaces has been investigated by Schröder and coworkers [17].

Several meshing techniques have been developed to visualize and analyze molecules. Molecular biology is probably one of the leading fields driving this domain. These techniques include the *van der Waals surface*, which is the surface obtained from the radii and coordinates of the atoms and the *solvent-accessible surface*, which is defined as the surface of the molecule that can actually be reached by molecule of a certain size, as well as other variants [18]. These surfaces are generally defined as isosurfaces of a parameter generated as a sum over all of the atoms of Gaussian functions representing the radius or the exclusion volume of each atom. The mesh is calculated subsequently either analytically [18] or using a 3D grid [19]. Although these methods allow the construction of nice meshes as well a thorough analysis of molecules, they add a significant computation overhead to the MD simulations and are therefore not well suited to our purpose.

3. Principle of the surface detection and meshing

Here we propose a method of surface detection and tessellation that is optimized for large-scale atomistic simulations. It is based on a simple geometrical construction. We describe the principle of the method heuristically here, and then present it more precisely in the following section. Consider the system composed of spheres representing atoms all of which have a radius equal to the atomic radius. Since the atomic forces are not infinitely stiff in general, some of these spheres may overlap slightly. Now consider another sphere brought through the vacuum into contact with the spheres of the surface. The surface atoms may then be detected by rolling that sphere across the surface,

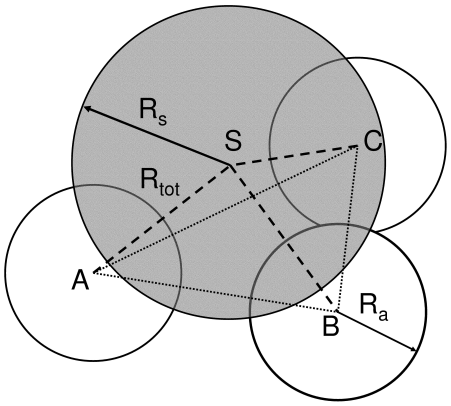

Figure 1. A detection sphere of radius R_s (shaded sphere) is used to probe the surface of atoms of radius R_a (white spheres). The algorithm looks for all of the *stable positions* defined as a position where the center S of the detection sphere is at a distance $R_{tot} = R_a + R_s$ from the center of at least three atoms (here atoms A, B and C) and no other atom is located at a distance smaller than R_{tot} from S. A mesh of the surface is implicitly defined as the polygon defined by the atoms in contact with the sphere (here the triangle ABC).

contacting but not penetrating into the surface spheres. Whenever the sphere contacts three or more atoms simultaneously, the centers of those atoms form the vertices of a polygon in the surface tessellation. We refer to this location as a *stable position*. For our purposes, we choose the surface to run through the centers of the surface atoms, although we note that there is some ambiguity in the position of the surface due to the question of how far the atoms extend into the vacuum. The principle of our surface detection method is sketched in figure 1.

More precisely, let us consider a configuration at a one instant of a MD simulation containing N identical atoms. All the atoms are assumed to be represented by spheres of radius R_a. We define a *detection sphere* of radius R_s. Using this detection sphere, the algorithm looks for all its stable positions as the sphere moves on the surface. A stable position is mathematically defined as a position of the detection sphere where its center S is at a distance R_{tot}, where $R_{tot} = R_a + R_s$, from the centers of each of at least three atoms (these atoms are said to be *in contact* with the detection sphere) and no other atom's center is located at a distance strictly smaller than R_{tot} from S. Once such a stable position is located, the atoms in contact with the sphere are identified as surface atoms and the polygonal surface between their centers is defined as a surface element (see figure 1).

A key aspect of the method is being able to find all the stable positions related to a given surface. This search must be exhaustive and non-redundant. These goals are achieved by basing the search on the edges of the surface elements. Since the surface is closed, every edge of a given surface element is necessarily the edge of another surface

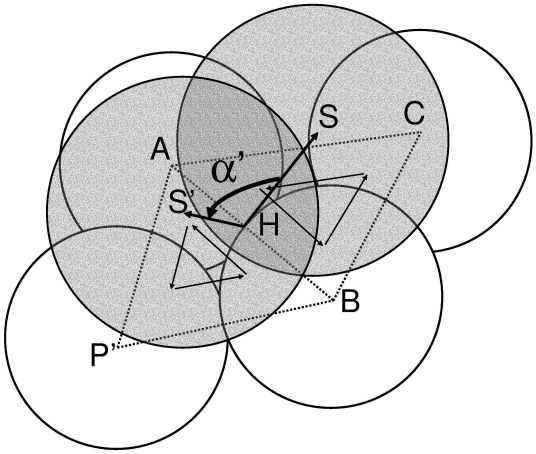

Figure 2. Once a stable position S for the detection sphere (shaded sphere) is detected (here between atoms A, B and C) other stable positions S' can be found by rolling sequentially that sphere across all the edges of the surface element (here segment $[AB]$). For each segment, another stable position S' exists necessarily and defines another element of mesh (here the triangle $AP'B$). The transition between S and S' is a rotation α' around this segment.

element. For definiteness, let us consider a situation where a stable position S was detected as represented in figure 2. We consider a particular segment $[AB]$ of this surface element. Then we know that a unique associated stable position S' can be found by moving the detection sphere around the axis AB while maintaining contact with atoms A and B. Mathematically S and S' are both located on a circle whose center is the middle of $[AB]$ and whose axis is AB. The 1D path between the two stable positions S and S' can be characterized by the rotation angle α' around AB. The algorithm proceeds as follows. It calculates the possible stable positions S'' and determines the rotation angle α'' for all of the neighbors of A and B, where $0 < \alpha'' < 2\pi$. The new stable position S' corresponds to the candidate with the smallest rotation angle α'. The corresponding surface element is added to the list, and its edges are added and removed from the list of unpaired edges, as appropriate. Once a first stable position is obtained, an exhaustive surface detection can be performed by analyzing successively all of the edges and iterating this process until no unpaired edge remains. This algorithm satisfies the goals that the search be exhaustive and avoid redundancy.

4. Details of the algorithm

In this section we present the details of the algorithm starting with the case of a single processor simulation. We assume that the positions of all of the atoms are known and that the code is equipped with an efficient data storage system that allows rapid retrieval of the atoms located in the neighborhood of any given position. The specific input parameters for the surface analysis are R_a, the radius of the atoms, and R_s, the

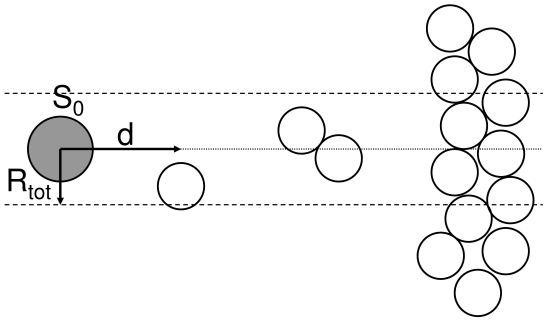

Figure 3. Starting from a position S_0, the detection sphere look for a first surface element in direction **d**. Specifically the algorithm finds successively all the atoms in the path of the detection sphere (i.e. atoms whose center is at a distance less than R_{tot} from its path). Isolated atoms, atoms with less than two neighbors and atoms that do not allow a stable position may be found in the path of sphere and are not considered.

radius of the detection sphere, although only the sum of the two, R_{tot}, affects which surface atoms are detected.

4.1. Finding the first stable position

The first step consists of finding an initial stable position. In our algorithm we assume that we know an initial position S_0 for the detection sphere such that it is not in contact with any atom. We further assume that we know a direction **d** in which to search for a surface. We do not believe these assumptions significantly limit the applicability of the method since the basic geometry of the system is known, either from the initial configuration or from an analysis at previous time steps. The question of surface nucleation through interfacial debonding or cavitation can be addressed with some of the techniques mentioned in section 2.

The detection of the first stable position is performed by finding successively atoms in the path of the detection sphere as it moves from S_0 in direction **d** (see figure 3), until one is found that is part of the surface. It may happen that the sphere encounters an isolated atom or pair of atoms prior to contacting the surface. The connectivity of an atom A is therefore assessed as follows. (i) We look for all of the neighbors of atom A within a distance $2R_{tot}$. By construction, all of the atoms in contact with the detection sphere are indeed at a distance R_{tot} from its center; therefore the distance between the atoms is necessarily smaller than $2R_{tot}$. To be part of the surface, the atom must have at least two neighbors. (ii) Among this collection of neighbors, there must be at least one triplet of atoms $\{A, B, C\}$ satisfying two criteria to be valid as a surface element. The three atoms must be sufficiently close and exposed to form a surface element, as

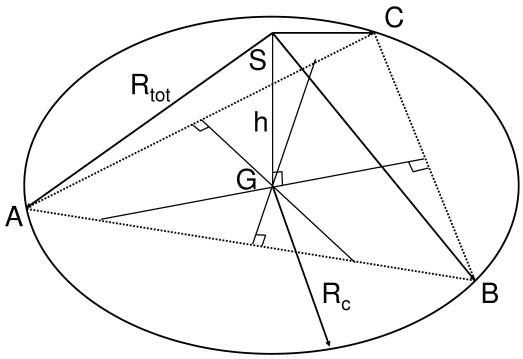

Figure 4. Geometric construction to calculate the position of the detection sphere S. The sphere is located a distance h out of the plane and directly above the center G of the circle that circumscribes the triangle ABC. The circumradius is R_c and the distance from S to A, B and C is R_{tot}.

described precisely below. For definiteness we take the candidate surface element to be oriented toward S_0, by reordering the vertices if necessary. In particular, a properly oriented element satisfies

$$\mathbf{AS_0} \cdot (\mathbf{AB} \times \mathbf{AC}) > 0. \tag{1}$$

If this is not the case, B and C are inverted. The first criterion ensures that the detection sphere can actually be in contact simultaneously with A, B and C. From a mathematical point of view, the circumradius defined by these points must be strictly smaller than R_{tot} (see figure 4); that is,

$$R_{tot} > R_c = \frac{AB\ AC\ BC}{2\|\mathbf{AB} \times \mathbf{AC}\|}. \tag{2}$$

Once this test is passed, the position of the stable position S can be calculated as follows. First we calculate the circumcenter G of the triplet as

$$\mathbf{AG} = \frac{(AB^2 - \mathbf{AB} \cdot \mathbf{AC})\, AC^2\, \mathbf{AB} + (AC^2 - \mathbf{AB} \cdot \mathbf{AC})\, AB^2\, \mathbf{AC}}{2\|\mathbf{AB} \times \mathbf{AC}\|^2}. \tag{3}$$

The distance h between the reference sphere and (ABC) is given by

$$h = \sqrt{R_{tot}^2 - R_c^2}. \tag{4}$$

The stable position of the detection sphere is given by

$$\mathbf{AS} = \mathbf{AG} + h\frac{\mathbf{AB} \times \mathbf{AC}}{\|\mathbf{AB} \times \mathbf{AC}\|}. \tag{5}$$

The second criterion is that no atom's center is located at a distance strictly less than R_{tot} from S. More than three atoms may simultaneously be in contact with the detection sphere, which is referred to as a *degenerate* configuration. In the case of a non-degenerate configuration, the *first surface element* is immediately defined as the

triangle between the three atoms. The oriented element is stored for use to start the iteration over free edges and generate the rest of the surface. Specifically, the surface element orientation was chosen to satisfy $\mathbf{AS}\cdot(\mathbf{AB}\times\mathbf{AC})>0$, and with this convention we store the *oriented* segments $[AB]$, $[BC]$ and $[CA]$, along with the stable point S in the form {[segment], stable position}. The case of the degenerate element (with more than three vertices) is analyzed in section 4.3.

4.2. Finding all the other stable positions

Finding the other surface elements is straightforward once a first surface element has been detected, since there is necessarily a unique stable position directly accessible by the detection sphere for each segment. They can all be retrieved by iteratively analyzing the segments already stored in memory until this list is exhausted. Let us consider an oriented segment $[AB]$ belonging to a surface element $\{A,B,C\}$ characterized by its stable position S (see figure 2). The procedure is referred to as *treating* this segment. We proceed as follows:

- We detect all of the neighbors P'' of *both* A and B within a distance $2R_{tot}$ of each point. Note that atom C must be included in this list.
- The detection sphere must be in contact simultaneously with A, B and P'', therefore we verify that the circumradius R_c between these three points is strictly smaller than R_{tot}. It can be written
$$R_c=\frac{AP''\;P''B\;BA}{2\|\mathbf{AP''}\times\mathbf{AB}\|}. \tag{6}$$
- For each possible point P'', we calculate the position of the circumcenter G'' as
$$\mathbf{AG''}=\frac{(AP''^2-\mathbf{AP''}\cdot\mathbf{AB})\,AB^2\,\mathbf{AP''}+(AB^2-\mathbf{AP''}\cdot\mathbf{AB})\,AP''^2\,\mathbf{AB}}{2\|\mathbf{AP''}\times\mathbf{AB}\|^2}. \tag{7}$$
The distance h'' between S'' and $(AP''B)$ is
$$h''=\sqrt{R_{tot}^2-R_c^2}. \tag{8}$$
The corresponding stable position S'' is given by
$$\mathbf{AS''}=\mathbf{AG''}+h''\frac{\mathbf{AP''}\times\mathbf{AB}}{\|\mathbf{AP''}\times\mathbf{AB}\|} \tag{9}$$
We calculate the rotation angle α'' of the detection sphere around $[BC]$ as it moves from S to S'' as
$$\cos\alpha''=\frac{\mathbf{HS}\cdot\mathbf{HS''}}{HS\;HS''} \tag{10}$$
$$\sin\alpha''=\frac{(\mathbf{HS}\times\mathbf{HS''})\cdot\mathbf{AB}}{HS\;HS''\;AB}, \tag{11}$$
where H is the mid-point of $[AB]$.
- Among all these points P'', we select the atom P' that *minimizes the rotation angle* [i.e. $\alpha'=min(\alpha'')$]. It is another surface atom. The corresponding S'' is

the corresponding new stable position S'. Several points P' may simultaneously minimize the rotation angle α' and correspond again to a degenerate configuration. It can be analyzed using the same procedure as in the detection of the first surface element (see the next section for details).

To avoid multiple storage of the same surface elements, a few additional steps are necessary. (i) In the list of detected segments, we put a flag on $\{[AB], S\}$ indicating that it has been treated. (ii) We add $\{[BA], S'\}$ in the list of segments directly with the same *treated* flag. (iii) Before storing $\{[AP'], S'\}$ and $\{[P'B], S'\}$ we make sure these segments are not already in the list. We note that both the oriented segment as well as the stable position must be tested to determine whether or not a segment has been treated. A segment may indeed be associated to more than two surface elements (e.g. a monatomic sheet of atoms). However these surface elements necessarily pair off and each pair is uniquely characterized by their stable positions.

Keeping track of the orientation of the segments is crucial to detect the atoms that actually belong to the same surface and to orient each surface element of the mesh. To be consistent, the orientation of segment $[BC]$ in the new detected surface element $\{CBA'\}$ must be the opposite of its orientation in the previous surface element as presented in figure 2. For example, it allows unambiguous determination of the position of the detection sphere S'' given by the stable position equation (9). The symmetric position (i.e. taking the opposite value of h'') is forbidden as it would require the detection sphere to cross the surface. Note that once the orientation of the first surface element is chosen, the orientations of all of the subsequent elements are fixed (but need not point toward S_0 for a convoluted surface).

4.3. Degenerate elements

A degenerate configuration is defined as when four or more atoms are simultaneously in contact with the detection sphere for a given stable position. As we saw earlier, such a situation may arise both during the detection of the first surface element and the detection of the remaining surface elements. In our experience, such a situation is rather unlikely during dynamic simulations. Therefore, as it requires more analysis than the case of three contact atoms, it is convenient to do the analysis in a separate routine.

There are two distinct degenerate cases (see figure 5). In the first case, all the contact atoms are coplanar; in the second case they are not. There can also be a mixture between these two families with different planes, which may contain four or more atoms.

Let us consider a list of M atoms ($M > 3$) that are in contact with the detection sphere S at position S. At this stage, whether considering the first surface element or later ones, the stable position S of the detection sphere is known. The first step of the analysis of degeneracy is to find all the atoms that are coplanar. This task can be performed by making a loop over all of the triplets of points $\{A, B, C\}$. Each triplet of points defines necessarily a plane since they cannot be aligned. The other atoms P that

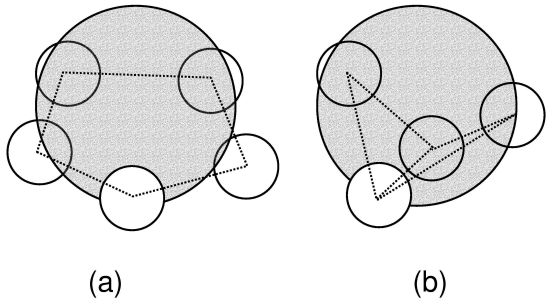

Figure 5. There are two types of degenerate situations. (a) More than three coplanar atoms may simultaneously be in contact with the contact sphere. (b) More than three non-coplanar atoms may be in contact with the sphere, hence defining two or more surface elements.

belong to the same plane can be detected using the criterion $\mathbf{AP} \cdot (\mathbf{AB} \times \mathbf{AC}) = 0$.

For each distinct plane, the edges of the surface element as well as their orientation must now be defined. Here we take advantage of the convexity of each polygonal surface element. The polygon is necessarily convex since all its vertices (atoms) are located at a distance R_{tot} from S. We proceed as indicated in figure 6. First we arbitrarily select two reference atoms A and B. We calculate the unit vector $\mathbf{n}$ normal to the plane verifying $(\mathbf{n} \cdot \mathbf{AS}) > 0$. For each atom (including B but not A), the rotation angle γ_i around $\mathbf{n}$ between $\mathbf{AB}$ and AP_i is calculated as

$$\cos\gamma_i = \frac{\mathbf{AB} \cdot \mathbf{AP_i}}{AB\ AP_i} \tag{12}$$

$$\sin\gamma_i = \frac{\mathbf{AB} \times \mathbf{AP_i}}{AB\ AP_i} \cdot \mathbf{n}. \tag{13}$$

The atoms P_i are consequently sorted by increasing value of γ_i. The orientation of the surface is given by $A \rightarrow P_1 \rightarrow P_2 \rightarrow \ldots P_{d-1} \rightarrow A$. The corresponding oriented segments are stored in memory as described in the previous section.

4.4. Expediting the analysis

Rapid retrieval of both the possible contact points and the stored segments is crucial. Using the storage structure of the MD code is therefore extremely useful. Consider the common MD data structure [20] resulting from dividing and storing the system in rectangular parallelepipeds (cells) whose edge lengths l are equal to or slightly larger than the cutoff radius r_c of the interatomic potential. All the neighbors required to calculate the various contributions to the energy and the forces exerted on one particular atom are therefore located in either the same cell as that atom or its 26 neighboring cells. We can take advantage of this existing structure to determine the atoms in contact with the detection sphere and to store the oriented segments. It is also worth noting that in the present algorithm, all of the surface atoms are necessarily detected within

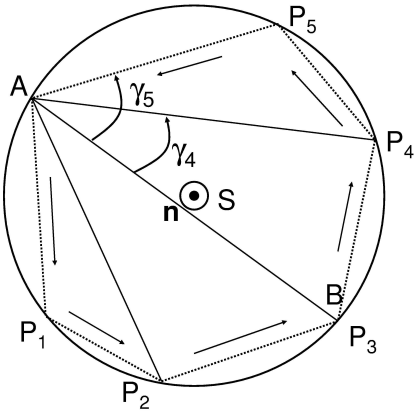

Figure 6. Analysis of a facet containing more than three atoms. All the atoms being located on a circle, the edge of the facet can be determined by sorting the rotation angles γ_i around between vector **AB** and the vectors $\mathbf{AP_i}$.

a distance $2R_{tot}$ from a given atom. It is therefore convenient to choose the radius of the atoms and the detection sphere such that $2R_{tot} < l$. The search for the next surface atoms can then be restricted to a given cell and its 26 neighbors. As many interatomic potentials include the second or third nearest neighbor atom shells, this restriction allows reasonable values for the atomic radius as well as a comparable but somewhat smaller radius for the detection sphere; in the case of a nearest neighbor potential such as the popular Stillinger-Weber potential, the algorithm is generalized to search over more neighboring cells.

4.5. Generalization to parallel processing

MD simulations are often amenable to parallelization, using a domain decomposition to map the atoms to different processors. Many parallel MD codes exist, typically achieving good (approximately linear) scaling through the use of MPI [21] for message passing between the processors and a fairly democratic structure, with little or no master processor operation [22]. This parallelization enables simulation of systems with many millions of atoms. Although there are other strategies, we assume that the system is partitioned in a Cartesian domain decomposition, in which the atoms are distributed between the processors in rectangular parallelepiped domains. Within each processor, the sub-system is divided in rectangular cells with edge lengths l, as described above for the single processor case. In order to calculate the energy and the forces at the boundaries, each processor also stores the positions of the atoms on other processors in the cells in contact with its own cells, information that must be passed at each time step of the simulation. Such a structure makes it easy to implement a parallel version of the surface detection algorithm by choosing the radius of the atoms and the sphere such that $2R_{tot} < l$, as before. The neighbors required to construct a surface element

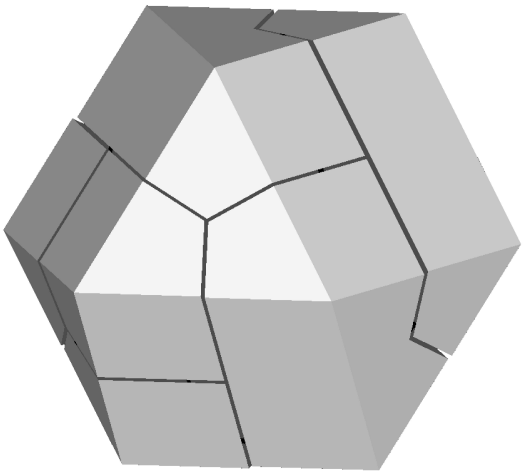

Figure 7. Parallel analysis of the surface of an unrelaxed vacancy in an otherwise perfect FCC crystal. The nodes of the surface are the 12 nearest neighbors of the missing atom. The missing atom was located at the intersection of 8 processors. The surface elements detected in each processor are slightly disjointed to demonstrate the uniqueness of the decomposition. Three degenerate surfaces, each of them containing 4 atoms, can also be seen on this figure.

from a segment whose first atom lies *within* the processor are indeed necessarily in the processor or in the boundary cells. Hence the segment can be treated without any additional interprocessor communication.

In order to obtain an exhaustive and unique detection of the surface, we proceed as in the single processor case with the following modifications. (i) Each processor only treats the oriented segments $\{[BC], S\}$ whose starting point B is *in* that processor. A segment is not to be *treated* if its starting point is in the boundary layer. (ii) A segment whose starting point is in the boundary layer may be legitimately *detected* as part of a surface element. This segment must later be sent to the appropriate processor where it will be treated. (iii) A given surface element may belong to different processors and is necessarily detected in all these processors. In order to construct a unique, non-overlapping and well-oriented mesh, we divide this surface as follows. Each subsurface is delimited by the atoms that are actually in the local processor, the middle of the segments that exit or enter the local processor and the center of gravity of the surface. This construction is illustrated in figure 7. It guarantees the uniqueness of the mesh.

5. Applications and validation

5.1. Void analysis during dynamic fracture simulations

We first consider the example of void growth mechanisms during dynamic fracture [23], with a particular interest in simulations at the atomistic level [2, 10, 24, 25, 26]. Understanding the nucleation, growth and coalescence of voids in ductile crystalline

materials is essential to understand the fracture of these materials and often hinges on a thorough quantitative analysis of void surfaces. Useful information includes the position of the void center, surface area, volume and shape. This analysis requires both the detection of the surface atoms and the construction of a mesh between these atoms in order to perform various surface integrals on the fly. These integrals can be assessed as a discrete sum over all the facets i of the mesh as

$$\int f(r)\, dS = \sum_{i}^{\text{facets}} \sum_{j}^{\text{atoms of i}} f(r_j)\, S_i(r_j)\, w_j^i, \tag{14}$$

where S_i is the surface of the facet and w_j^i is the weight of the atom j in that facet. Using the decomposition into subsurfaces presented in the previous section, we weight the atoms by the ratio between their subsurface and the surface of the facet. This decomposition is particularly adapted to parallel computing as the contribution of each individual processor can be calculated as a sum over all the facets detected locally. As some facets may be detected in multiple processors, the weight of each atom is taken as non-zero only in the processor where it is specifically located to avoid double counting. The surface area of the void can therefore be calculated as

$$S = \sum_{i}^{\text{facets}} \sum_{j}^{\text{atoms of i}} S_i(r_j)\, w_j^i. \tag{15}$$

For a triangular element, $S_i = \frac{1}{2}\,|\mathbf{AB} \times \mathbf{BC}|$, and all elements contribute positive increments to the total surface area. The volume of the void can be calculated from an arbitrary reference point P as

$$V = \sum_{i}^{\text{facets}} \sum_{j}^{\text{atoms of i}} V_i(r_j)\, w_j^i, \tag{16}$$

where V_i is the volume of the general pyramid defined by P and the facet i. For a triangular element, $V_i = \frac{1}{6}\mathbf{AP} \cdot \mathbf{AB} \times \mathbf{BC}$. V_i may be positive or negative depending on whether the facet is oriented toward P or not. For higher order polygonal facets, the area and volume may be computed using a decomposition into triangles.

Following the work of Seppälä et al. [10], the shape of a void can also be well characterized using an analysis of the spherical harmonics of the void from its center. It can be written in parametric form as

$$r^2\,(\theta,\phi) = \overline{r^2} \sum_{l=0}^{\infty} \sum_{m=-l}^{l} Q_{l\bar{m}} Y_{lm}\,(\theta,\phi)\,, \tag{17}$$

where the mean square radius is calculated as a surface integral as

$$\overline{r^2} = \frac{1}{4\pi} \int r^2\,(\theta,\phi)\; d\Omega = \frac{1}{4\pi} \sum_{i}^{\text{facets}} \sum_{j}^{\text{atoms of i}} r_j^2 \Omega_i(r_j)\, w_j^i, \tag{18}$$

where Ω_i is the solid angle of the facet i as seen from the center of the void. The multipole moments are calculated as

$$Q_{lm} = \frac{1}{\overline{r^2}} \int Y_{lm}\,(\theta,\phi)\, r^2\,(\theta,\phi)\; d\Omega \tag{19}$$

$$= \frac{1}{\overline{r^2}} \sum_{i}^{\text{facets}} \sum_{j}^{\text{atoms of i}} Y_{lm}(\theta_j, \phi_j)\, r_j^2 \Omega_i(r_j)\, w_j^i. \tag{20}$$

In the present example, classical molecular dynamics simulations are performed on a single-crystal initially containing 6 912 000 copper atoms in a cubic box with {100} faces. The forces were calculated using the embedded atom method (EAM) interatomic potential for copper due to Oh and Johnson [27]. Periodic boundary conditions are used in all three directions. After being brought to thermal equilibrium at 300 K, two spherical voids with a radius 2.17 nm are cut in the system at a distance 12.2 nm from each other and a dilatational strain is applied at a constant strain rate of 10^9 s^{-1}. Details about the simulations have been published elsewhere [26].

Surface analysis is performed starting from the initial position of the center of each void every 10 time steps. The cutoff radius of the embedded-atom potential used in this simulation [27] is 1.9 times the nearest neighbor distance r_e for a perfect (face centered cubic) crystal at 0 K. This cutoff allows us to choose an atomic radius of $R_a = 0.5r_e$, as well as a comparable detection sphere radius of $R_s = 0.445r_e$. Figure 8 presents the analysis of one void after 80 ps. The surface atoms and corresponding mesh are presented respectively in the top left and right pictures. Using the multipole decomposition, it is possible to store the shape of the void using a limited number of parameters. On the bottom left picture we reconstructed the shape of the void using the center of the void and the multipole moments Q_{lm} up to $l = 4$. Comparison with the entire set of surface atoms in figure 8(d) shows that this void geometry can be stored with a good precision using as few as 27 parameters as presented in the bottom right picture. This description is also a step toward a multiscale modelling of void growth as it can be used to generate a mesh suitable for finite-element calculations. In this system there were 3177 atoms detected on the void surface, with a total of 7706 facets. The analysis took 1.14 CPU-seconds to complete not including I/O on a single processor system (2.4 GHz Xeon processor). For comparison, a single MD step took 155 CPU-seconds. This example has also been run with a parallel code using up to 96 processors. While analysis shows a significant reduction of the wall-clock time for the surface analysis, a direct comparison of the total CPU-time with the single-processor analysis may be ambiguous as the results are extremely surface dependent. Indeed, the surface may not be distributed among all the processors, and therefore not all the processors may be involved in the analysis.

As we pointed out in section 3, R_{tot} is the only arbitrary parameter that is introduced in this algorithm. We use this first example to study more precisely its influence on physical quantities such as the surface area, the volume and the multipole moments of the void as well as on the number of facets that are detected by the algorithm. The results are presented in figure 9. R_{tot} is expressed as in units of nearest neighbor distance r_e. We note that no stable position can be found for values of R_{tot} smaller than $0.6r_e$. This percolation threshold can be estimated with a fairly good precision as the smallest circumradius between three atoms in a perfect crystal at 0 K.

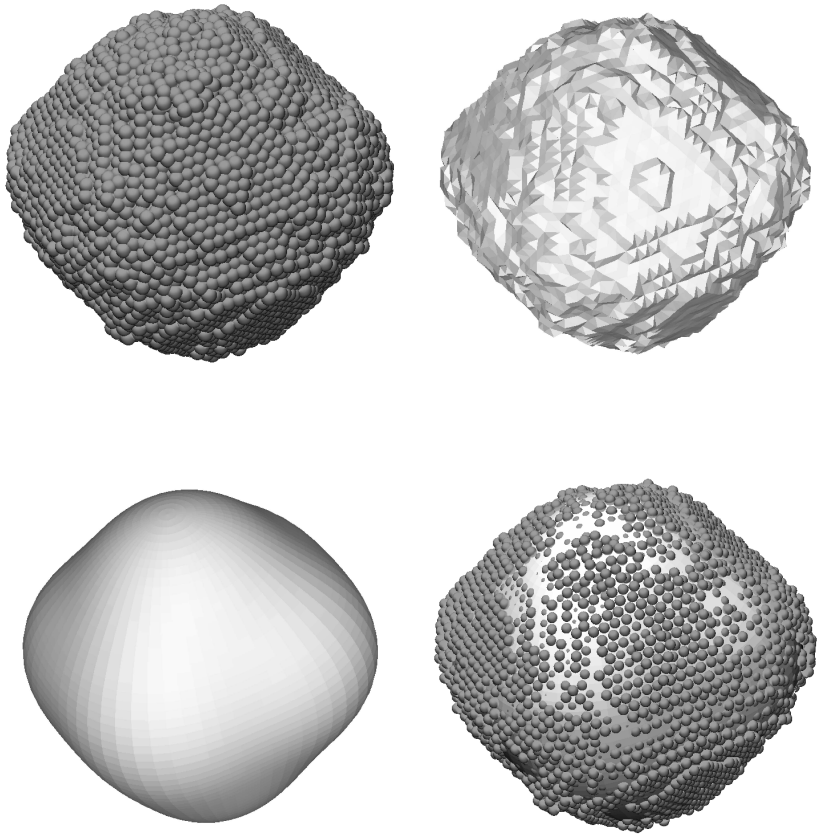

Figure 8. Four different views of the same void expanding in a copper single crystal during dynamic loading. Top left: surface atoms. Top right: mesh as detected by the sphere. Bottom left: reconstructed mesh using spherical harmonics. Bottom right: the atoms and the reconstructed mesh superimposed.

This threshold is given by $r_e/\sqrt{3}$ for a face-centered cubic lattice. At the other end of the scale, although there are no mathematical limitations for the largest R_{tot}, large values of R_{tot} imply testing more atoms for each oriented segment that is considered and reduce significantly the computational efficiency of the algorithm. We note that the influence of R_{tot} on measured quantities is significant close to the percolation threshold but it is limited when the radius of the detection sphere is close to the radius of the atoms i.e. $R_s = R_a = r_e/2$. Thus for both physical and numerical reasons it best to choose R_{tot} approximately equal to r_e. We have chosen the void with its curved and rough surfaces as a stringent test of the variation with R_{tot}, and we find that the volume of the void and Q_4 do not depend significantly on this parameter as a 1% change in R_{tot} causes respectively less than 0.05% and 0.07% change in these quantities. Unsurprisingly the surface of the void and the number of facets are slightly more sensitive to R_{tot} (resp. 0.2% and 0.5% change for 1% change in R_{tot}) as these parameters are more sensitive to the roughness of the void.

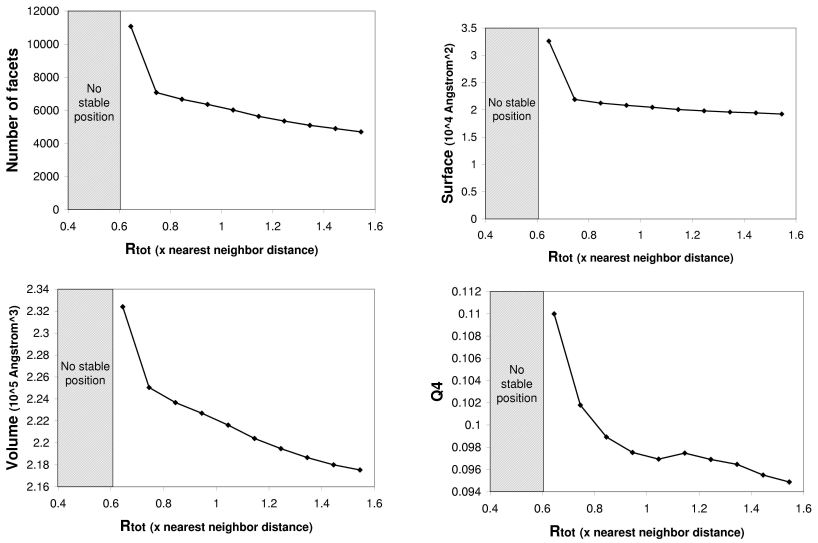

Figure 9. Influence of R_{tot} on the number of facets, surface area, volume and multipole moment Q_4. These quantities were computed for the void shown in figure 8, a challenging case because of the high curvature and roughness of the void surface.

Another compelling feature of the algorithm is the ability to detect coalescence between the two initially separate voids. The concept of coalescence is straightforward in continuum models. It takes place the moment when the cavities of the two voids merge to form a single void. In the context of molecular dynamics where the atoms are defined as points, such a definition is ambiguous. By providing the concept of *surface*, our method provides a natural definition of coalescence as the moment when it detects the surface of the two voids as a *single* surface. In other terms, the two voids are said to coalesce when the detection sphere can move unobstructed from one void to the other. The voids are thus interconnected by a hole, whose size is larger than the diameter of the sphere. In figure 10, we present the surface of the two voids as they coalesce after 91 ps. This surface was detected as a single surface with a starting point located in one of the original voids. This example also gives us the opportunity to motivate a choice for the radius of the detection sphere. Here it is seems legitimate to define coalescence from a physical point of view as the moment when an *atom* can freely move from one void to another, thus implying that the radius of the detection sphere should be equal to the radius of the atoms. The EAM potential cutoff does not include fourth nearest neighbor atoms, so we have chosen the detection sphere as large as permitted by the optimal neighbor cell size. Alternately, in the case of a gas or a liquid moving around our surface, we can choose the radius of the detection sphere to be equal to the size of these atoms or molecules. Note that this choice is similar to the concept of solvent-accessible surface [18].

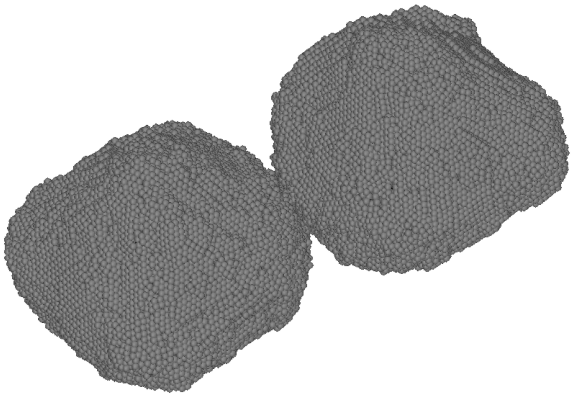

Figure 10. View of the surface atoms of two initially distinct voids as they coalesce to form a *single* surface. This surface was detected with a *unique* starting point S_0 located in one of the original voids. This example demonstrates both the ability of the present method to detect void coalescence and to treat convoluted surfaces.

5.2. Surface analysis of a nanoscale plate under tension

The deformation of external surfaces is also of interest in the study of the plasticity of metals. The formation of surface steps due to dislocation motion in deformation experiments is often used to infer glide system activity [28]. The behavior of thin films under stress [28] and the fatigue and ultimate failure of metals [29] are closely related to this phenomenon. In atomistic simulations once again tessellation of the surface facilitates a quantitative analysis of topographical evolution. It is also worth noting that recent developments of discrete dislocations codes allow them to treat surfaces including dislocations arriving on surfaces and/or direct modelling of the surface topology [29, 30, 31]. The continuum analysis of atomistic surface deformation opens the door to a direct comparison between discrete dislocation dynamics results and MD simulations.

In the present case, we have examined the surface topography that results from the deformation of a nanoscale plate under biaxial tension. The copper plate was initially 91.3 nm × 91.3 nm × 6.5 nm (140×140×10 fcc unit cells with a total of 784 000 atoms) and free from defects such as dislocations. Periodic boundary conditions were imposed along the two larger dimensions, whereas the plate had free surfaces in the third dimension. The same interatomic potential and code parameters were used as in the void simulations, except that the calculation was conducted on a single CPU. It was brought to thermal and mechanical equilibrium (zero stress) at room temperature and subjected to a constant strain rate expansion of the periodic box at a fixed strain rate. The strain rate for this simulation was 10^{10} s^{-1}, which is very high, corresponding to the upper limit of what may be achieved with powerful lasers such as the National Ignition Facility (NIF). Eventually the biaxial stress increased to the point that it overcame the barrier to surface nucleation of dislocations, and partial dislocations were observed

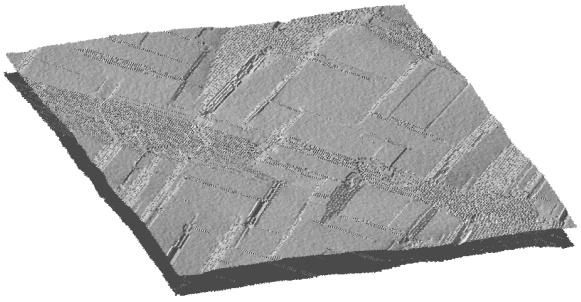

Figure 11. Surface of a nanoscale plate showing slip steps as a result of plastic deformation. The tessellated surface is shown at the point of a total biaxial strain of $\varepsilon_{xx} = \varepsilon_{yy} = 8\%$. The slip steps indicate points at which dislocation were either nucleated at the surface or absorbed into the surface.

to nucleate at one surface and quickly traverse the plate to be absorbed by the other surface, leaving a surface step on both surfaces and a stacking fault spanning the plate. The trailing partial dislocation often is not nucleated in nanoscale systems due to the energetics of the stacking fault. For a recent analysis of trailing partial nucleation in a somewhat different nanoscale system, see van Swygenhoven et al. [32].

At the point at which the surface was analyzed, the system had undergone a biaxial tensile strain of $\varepsilon_{xx} = \varepsilon_{yy} = 8\%$. Substantial dislocation activity had taken place. The surface roughening due to the formation of slip steps is clear from the tessellation of the surface shown in figure 11. In this case there were 51 173 and 51 122 atoms detected on the two surfaces, with a total of 111 044 and 106 200 facets, respectively. The analysis took 17.5 CPU-seconds to complete not including I/O on a single-processor system.

5.3. Dynamic fracture of polycrystalline copper

In this last section we give an example of how it is possible to extend our method beyond the cases of void and free-surface analysis. Material *interfaces* are also of interest and with a slight generalization accessible to our simulation techniques. Crystallographic defects such as material interfaces can be detected using a variety of diagnostic tools such as centrosymmetry deviation parameters [6], potential energy per atom [3] or other order parameters [7]. Our method does not provide an alternate detection method, however by combining it with these well-characterized techniques it is possible to provide a mesh for these defects that may be useful for a quantitative analysis. Such an analysis can be performed by detecting the interface between atoms having different properties. Based on a threshold on some of these properties, the algorithm can be used to build a mesh at the interface between atoms that pass this threshold and the others.

Here we consider the example of the grain boundary and porosity structure in dynamic fracture of polycrystalline copper. The simulations that produced the configuration analyzed here have been published previously, and the interested reader

is directed to that article for details [24]. Briefly, a cubic box containing 1.1 million atoms divided into four different grains with different crystallographic orientations was constructed. After thermal equilibrium was established at 300 K, a uniform triaxial strain rate of 10^9 s^{-1} was imposed on the system to induce an increasing tensile stress. The voids were found to nucleate preferentially at grain boundaries and dislocation emission from the void surface was also reported. The snapshot used for the proof-of-principle analysis here corresponds to Figure 1(b) in Reference [24].

Figure 12 displays the surface of the three voids that are detected in this snapshot starting from three MD cells that did not contained any atoms. We also use our algorithm in conjunction with the centrosymmetry deviation parameter to detect grain boundaries. Specifically, we filter out the atoms with a low value of the centrosymmetry deviation parameter, which are not associated with any defect, as well as the atoms related to stacking faults, which have a particular centrosymmetry value [6]. The latter are not considered to prevent the detection of stacking faults as grain boundaries. Starting from the interior of each grain, we are able to mesh and analyze the surface of each grain. We note that only three different grains are found. This reduction from the initial four indicates that locally no crystallographic misorientation is detected between two of the initially distinct grains. In this system there were 7097, 13 113 and 19 517 atoms detected on the grain boundaries, with a total of 14 362, 35 411 and 52 833 facets. The analysis took 74 CPU-seconds to complete not including I/O on a single processor system. This result highlights the rotation of the grains, as well as the ability of our method to detect this type of grain coalescence.

6. Conclusion

We have presented a new algorithm that simultaneously detects and tessellates surfaces during MD simulations. Tessellation is the first step in many kinds of analysis of the geometry. Based on a simple geometric construction this algorithm provides a robust and efficient criterion for *surfaces* and *surface atoms*. It can detect convoluted surfaces and has been demonstrated to work for closed surfaces even when the topology changes (two surfaces coalesce into one). It does not involve a particular parametrization of the surface nor preferential axes. Taking advantage of the structure of MD codes including parallelism, it allows a full analysis of surfaces without having to save the positions of all of the atoms to disk for post-processing. Several applications including shape analysis, coalescence detection, and the evolution of both internal and external surfaces have been presented. Possible links with finite-element and discrete dislocation dynamics code have been noted. Finally we have shown how to apply this method to analyze material interfaces at the atomistic level.

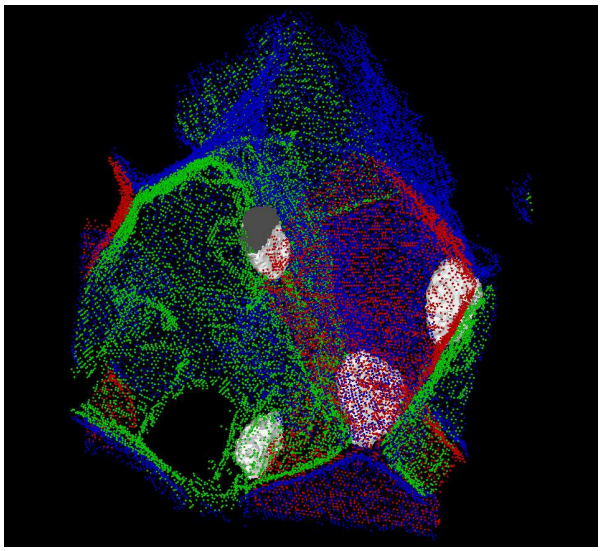

Figure 12. Dynamic fracture of a polycrystal. Three voids are detected in the system (mesh represented). Due to periodic boundary conditions the void on the left is cut in two parts, the grey area corresponding to its interior. Three different grains are detected in the system and are represented as atoms of different colors (mesh not represented). Analysis shows that the voids are preferentially nucleated at grain boundary junctions.

Acknowledgments

The MD code, MD3D, used in this article was written initially by J. Belak and later parallelized by E. T. Seppälä, and we gratefully acknowledge their work. This work was performed under the auspices of the US Department of Energy by the University of California, Lawrence Livermore National Laboratory, under Contract No. W-7405-Eng-48.

References

[1] Sutton A P and Balluffi R W 1995 *Interfaces in Crystalline Materials* (Oxford: Clarendon)
[2] Belak J 1998 *J. Comp. Aid. Mat. Design* **5** 193–6
[3] Abraham F F 1997 *Europhys. Lett.* **38**, 103–6
[4] Abraham F F, Brodbeck D, Rudge W E and Xu X 1997 *J. Mech. and Phys. Solids* **45** 1595–1605
[5] Zhou S J, Beazley D M, Lomdahl P S and Holian B L 1997 *Phys. Rev. Lett.* **78** 479–82
[6] Kelchner C L, Plimpton S J and Hamilton J C 1998 *Phys. Rev. B* **58** 11085–8
[7] Steinhardt P J, Nelson D R and Ronchetti M 1983 *Phys. Rev. B* **28** 784–805
[8] Honeycutt J D and Andersen H C 1987 *J. Phys. Chem.* **91** 4950–63
[9] van Swygenhoven H, Farkas D and Caro A 2000 *Phys. Rev. B* **62** 831–838
[10] Seppälä E T, Belak J and Rudd R E 2004 *Phys. Rev. B* **69**, 134101
[11] Ciarlet P G 1978 *The finite element method for elliptic problems* (Amsterdam: North Holland)
[12] Bowyer A 1981 *Computer J.* **24** 162–6
[13] Lo S H 1985 *Int. J. Numer. Meth. Eng.* **21** 1403–26
[14] Sloan S W *Adv. Eng. Software* **9** 34–55
[15] Yerry M A and Shephard M S 1984 *Int. J. Numer. Meth. Eng.* **20** 1965–90

[16] Weatherill N P and Hassan O 1994 *Int. J. Numer. Meth. Eng.* **37** 2005–39
[17] Rossi F and Schröder P 2005 "Combinatorial mesh optimization and empty circumcircles," unpublished
[18] Gabdoulline R R and Wade R C 1996 *J. Mol. Graphics* **14** 341–53
[19] Laskowski R A 1995 *J. Mol. Graphics* **13** 323–30
[20] Allen M P and Tildesley D J 1987 *Computer Simulations of Liquids* (Oxford: Clarendon)
[21] Gropp W, Lusk E and Skjellum A 1994 *Using MPI: portable parallel programming with the Message Passing Interface* (Cambridge: MIT Press)
[22] Plimpton S 1995 *J. Comp. Phys.* **117** 1–19
[23] Davison L, Grady D E, and Shahinpoor M (eds) 1996 *High-Pressure Shock Compression of Solids II: Dynamic Fracture and Fragmentation* (Berlin: Springer-Verlag)
[24] Rudd R E and Belak J 2002 *Comp. Mat. Science* **24**, 148–153
[25] Seppälä E T, Belak J and Rudd R E 2004 *Phys. Rev. Lett.* **93**, 245503
[26] Seppälä E T, Belak J and Rudd R E 2005 *Phys. Rev. B* **71** 064112
[27] Oh D J and Johnson R A 1988 *J. Mater. Res.* **3** 471–8
[28] Coupeau C, Girard J C and Grilhé J 1998 *J. Vac. Sci. Technol. B* **16** 1964–70
[29] Déprés C, Robertson C F and Fivel M C 2004 *Mater. Sci. Eng. A* **387–389** 288–91
[30] Weygand D, Friedman L H, Van der Giessen E and Needleman A 2002 *Modelling Simul. Mater. Sci. Eng.* **10** 437–68
[31] Tang M, Cai W, Xu G and Bulatov V V 2005 submitted to *J. Mech. Phys. Solids*
[32] van Swygenhoven H, Derlet P M and Frøseth A G 2004 *Nature Mater.* **3**, 399–402