2.31
ATOMISTIC VISUALIZATION

Ju Li
Department of Materials Science and Engineering,
Ohio State University, Columbus, Ohio, USA

Visualization plays a critical role in materials modeling. This is particularly true for atomistic modeling, in which there is a large number of discrete degrees of freedom (DOF): the positions of the atoms. Atomic resolution is therefore the defining feature of atomistic visualization. This, however, does not exclude the possibility of going up in scale – visualizing the coarse-grained continuum fields, or going down – visualizing the electronic structure around a particular atom or cluster of atoms, of a configuration if the need arises.

These discrete DOF in an atomistic simulation do not necessarily satisfy any smoothness condition like the continuum fields. For example the reconstructed atomic structure of a dislocation core in Si is not likely to be describable by a formula or a series expansion. However, this does not mean that there is no order in these DOF. Atomic-level order is ubiquitous in materials, even in amorphous or disordered materials, even in liquids. Finding these order, quantifying them, and then representing them in the best light are the tasks of atomistic visualization. Atomistic visualization is not merely a software engineering problem, it is also inherently a physico-chemical and mechanics problem.

To appreciate the importance of atomistic visualization, one must recognize that in a setup like a large-scale molecular dynamics (MD) simulation, it is not infrequent that the DOF self-organize in ways that the investigator would not have expected before the simulation is carried out. Thus, a main function of atomistic simulation is *discovering* new structures, new kinetic pathways and micro-mechanisms, with atomic resolution. Even though these discoveries often need to be taken with a grain of salt due to the present accuracy of empirical interatomic potentials, large-scale simulation is nonetheless a unique and tremendously powerful tool of identifying key structures and processes. Once a structure or a process is clearly described and understood, it often can be isolated and modeled with a much smaller number of atoms at the first-principles level, allowing one to eventually select the most probable structure

S. Yip (ed.),
Handbook of Materials Modeling, 1051–1068.

or process out of a catalog of possible low-energy structures or processes. This *surveying* mission of large-scale simulation would be impossible without efficient visualization, for the amount of data from a large-scale simulation is truly enormous.

This article is organized as follows. First, a brief survey of the present state-of-the-art in atomistic visualization is given, that includes both tool development and work done using the tools. Special emphasis is put on public-domain visualization tools that the author is familiar with. Then, the design philosophy behind the free atomistic configuration viewer AtomEye is explained. Finally, a recently developed characterization of local atomic structure called the central symmetry parameter [27] is explained.

1. A Brief Survey of Molecular Visualization

At the time this article is written, the state-of-art in atomistic visualization can be experienced in a movie that Farid Abraham *et al*. (IBM) made for a one-billion atom MD simulation of work-hardening, with two notched dislocation sources [1]. The MD simulation was performed for 200 000 time steps on the 12-teraflop, 4096-node ASCI White supercomputer at LLNL, for four days wall-clock time, which generated 25 terabytes of raw data. They were compressed with 30× efficiency to less than 1 terabyte, which would still take about 10 hard drives (weighs ∼1.2 lb each) to store. The movie was made in the post-processing stage by Mark Duchaineau, a computer scientist (LLNL). It has a resolution of 640×480, a file size of 66 megabytes, and lasts 46 s. In terms of file size, the movie is less than a 1/100% of the raw data. Watching the movie takes only a $1/100\%$ of the time it takes the fastest computer in the world to run the simulation. Yet, one gets a very good overview of what went on in the simulation, that entail a plethora of dislocation nucleation, interaction and dynamics, by just watching the movie. Thus, a main purpose of visualization is condensation of information. A crucial trick that enables such high condensation rate of information or data is *selective representation* of atoms. That is, one only renders "interesting" atoms near defects in the atomistic configuration, in this case dislocations and cracks. The "uninteresting" atoms which have bulk order are not rendered and do not cover up the field of view. Here, the "interesting" atoms are determined by a local energy criterion. Later in the article, we are going to illustrate alternative methods of distinguishing "interesting" atoms using some geometrical criteria without knowing the particular interatomic potential used.

As a side note, it was observed personally that the above movie never fails to captivate the audience in seminars and lectures, whether they are experts or not. Thus, aside from sifting and compressing information, atomistic visualization also lowers the barrier of entry for accessing the information.

Century-old methods of scientific visualization such as graphing/charting are still important as ever (they achieve even higher information compression rate). But the new kinds of visualization that come with the Information Age, in the forms of snapshots, movies/animations, and interactive navigation, greatly complement and enhance the traditional methods.

Top-quality atomistic visualization such as above [1, 2] still require the expertise of dedicated computer science professionals. They may also require specialized hardware such as an Immersadesk or CAVE system [3, 4]. However, for day-to-day research, there is an array of visualization software available on personal computers. Commercial modeling packages such as Materials Studio, CAChe, ChemOffice, HyperChem, Spartan, etc. come with powerful visualization front ends, that usually include graphical user interface (GUI)-driven atomic configuration builders as well. And there are also more specialized crystallographic software such as CrystalMaker. But here we are going to focus on free software, or freeware, that are accessible to everyone.

Molscript [5] and Rasmol [6] are two pioneering freewares that have had tremendous impact on visualization, beyond the field of molecular biology from which they originated. According to the Institute for Scientific Information (ISI), from 1991 to 2004 the Molscript paper [5] has been cited more than 10 000 times, making it one of the most cited papers in science. Molscript takes an input file, which specifies the 3-D coordinates of biomolecules and the desired graphics state (such as viewpoint), and renders into publication-quality schematics in vector image formats like PostScript, which can be directly inserted into typesetting program such as LaTeX. Later, photorealistic rasterization program Raster3D [7, 8] and charge-density isosurface plotting program CONSCRIPT [9] were developed that can work in unison with Molscript. Similar to many present-day raytracing programs, Molscript, Raster3D and CONSCRIPT run on the command line and are noninteractive. So, while the qualities the configuration snapshots are excellent, they are less suitable as a configuration navigation and surveying tool. Rasmol, on the other hand, is designed with navigation in mind. One is able to rotate the configuration and change the rendering state interactively. The Rasmol source code, which is freely available starting from the early 1990s, implements advanced features such as shared memory extension for local display, scripting interface, and various fast rendering technique, and advances the knowledge-base of molecular visualization freeware. Other macromolecule visualization tools with similar functions include the Swiss-PdbViewer (Deep View) [10, 11], and MOLMOL [12].

It should be pointed out that there are many detailed differences between molecular visualization of soft matter, specifically proteins, and atomistic visualization of hard matter. For example, in modeling deformation of solids, one can often use the perfect crystal as reference state. This means, in a visualization scheme, collective modes or defects can often be identified by comparing

with crystalline order atom by atom. Configuration changes in hard matter such as defect nucleation and mobility are often accompanied by the breaking and reformation of stiff, nearest-neighbor covalent or metallic bonds. In proteins, there is no crystalline reference state, and conformation changes are usually accomplished by the breaking and reformation of softer, non-nearest-neighbor bonds like hydrogen bonds. And while the concepts of local strain and stress are still useful in proteins [13], quantification/visualization poses perhaps a greater challenge. On the other hand, there are well-recognized local orders in proteins such as α-helices, β-sheets, turns and loops, that do not have direct analogies in hard matter, and require special representations such as ribbons/thick tubes, arrows, and lines/thin tubes.

Historically, the Protein Data Bank (PDB) configuration file format [14] and the Research Collaboratory for Structural Bioinformatics (RCSB) molecular structure database has been a major driving force behind promoting molecular visualization and standardization. No such standards yet exist in materials modeling. However, there are several good reasons *not* to use the PDB format to save one's configurations and for information exchange in atomistic modeling of hard matter, which are:

- *Precision*. PDB format has a fixed precision of 0.001 Å for storing the atomic coordinates. While this is probably sufficient for proteins, for which one usually models at around $T = 300\,\mathrm{K}$ in solution so there is plenty of thermal noise anyway, it is often not precise enough for hard matter.
- *Extensibility*. Since PDB adopts a fix-line format, there is no standard and supported way to add in new properties. For instance, there is no standard option to store atomic velocities.
- *Support for periodic boundary condition (PBC)*. It is very difficult to coax the PDB format to robustly and consistently store atomic configurations satisfying PBC, because the atomic coordinates are saved in direct Cartesian x, y, z coordinates rather than dimensionless reduced coordinates [15]. In order to effect an affine transformation on the supercell, for instance, one needs to modify all atomic coordinates explicitly in PDB, rather than just modifying the 3×3 **H**-matrix [15].

An extensible, arbitrary-precision configuration file format (CFG) and its supporting viewer AtomEye [16] is introduced in the next section, which provides full support for PBC and is most suited for large-scale MD simulations.

We now turn to another area, quantum chemistry, which also had profound influence on atomistic visualization. One deals with a smaller number of atoms in one configuration, usually no more than a few hundred at present, but scalar fields such as orbital wavefunctions need to represented besides the molecular conformation. The pioneering freeware in this field is Molden [17], which renders the orbital wavefunctions, charge density and electrostatic potential

of molecules, as well as their relaxation dynamics, vibrational normal modes and reaction pathways. It works well interactively, but also gives good quality vector graph output for 2-D contours and 3-D isosurfaces. Another freeware with similar functionalities is gOpenMol.

An excellent freeware for visualizing electronic structure in crystals is XCrySDen [18, 19]. One can store the crystal structure plus an arbitrary number of scalar fields defined on a regular grid under PBC in the so-called XSF format, which can be visualized, rotated and numerically manipulated interactively. Isosurfaces and cut-plane contours of the scalar fields can be rendered with a variety of colormap, transparency, and specularity options. Both the onscreen display and the snapshots have outstanding quality, and the controls are highly responsive. XCrySDen also has some tools for analyzing reciprocal-space properties such as interactive selection of **k**-paths in the Brillouin zone for band-structure plots, and visualization of the Fermi surface.

Presently, the most powerful and versatile freeware for visualizing molecular dynamics simulation trajectories is perhaps VMD [20]. It is based on OpenGL, with graphical user interfaces, but also a command line with full scripting capabilities. There is even a special syntax for choosing subsets of atoms for display (includes boolean operators, regular expressions, etc.). Trajectories can be played back, analyzed and easily converted to movies. Sterescopic display is fully supported. VMD can also display volumetric data sets, including electron density maps, electron orbitals, potential maps, and various types of user-generated volumetric data. They can be rendered using "VolumeSlice" or "Isosurface" representations, each of which provides several geometric rendering styles for viewing the data, varying isolevels, slice plane position, etc. 1-D, 2-D, and 3-D textures can be applied onto molecular and volumetric data representations to convey various types of information. VMD also provides the ability to render molecular scenes using external programs such as ray-tracing programs. This feature can be used to attain higher image quality than that is possible using the built-in OpenGL rendering features. There are also many special features for analyzing large biomolecular systems.

Compared to VMD, freeware such as AViz [21] and AtomEye [16], which are dedicated to atomistic visualization of nonbiological systems, are more lightweight. A good idea for beginners is to install and try all three freewares. The design philosophy behind AtomEye [16] is introduced in the next section.

Aside from the specialized tools introduced above, there are general visualization packages such as OpenDX and VTK, that are programmable and extremely powerful. The python interface of VTK, for instance, has been incorporated into Atomic Simulation Environment (ASE), an open-source distribution of python scripts [22] that can wrap around several *ab initio* and molecular mechanics engines (Dacapo, SIESTA, MMTK, etc.). The commercial software package MATLAB is also a very good environment for data visualization. Freeware in this direction include Gnuplot, Grace, Octave, and Scilab.

2. Design of an Efficient Atomistic Configuration Viewer

AtomEye [16] is a memory-efficient and lightweight atomistic configuration viewer, which nonetheless achieves high quality in the limited number of things that it can do. It is based on the observation that when visualizing MD simulation results, most often only the spheres and cylinders, representing the atoms and bonds, need to be rendered in massive quantities. Therefore, special subroutines were developed to render the spheres and cylinders as graphics *primitives*, rather than as composites of polygons. This combined with area-weighted anti-aliasing [23] greatly enhance AtomEye's graphics quality. One can also produce snapshots (in PNG, JPEG or EPS file formats) of a configuration in the desired graphics state at arbitrary resolutions (like 2560×2560) that are greater than the monitor display resolution, to obtain publication-quality figures (Figs. 1–6). Making movie is straightforward with a set of sequentially named configuration files.

AtomEye is an easy-to-use configuration *navigator* with full support for PBC. The user can move the view frustum anywhere inside the atomic configuration (see Figs. 2, 4). This is done by defining an *anchor* point, which can be the position of an atom, the center of a bond, or the center of mass of the entire configuration. Dragging the mouse up or down with the right mouse button

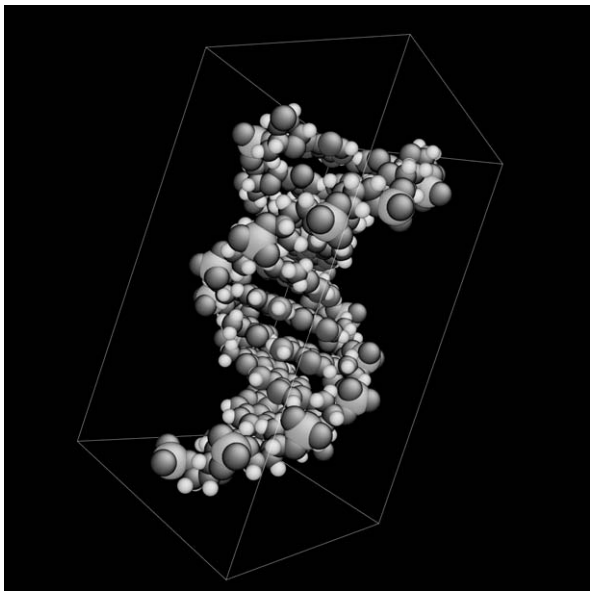

Figure 1. A strand of DNA, visualized in AtomEye.

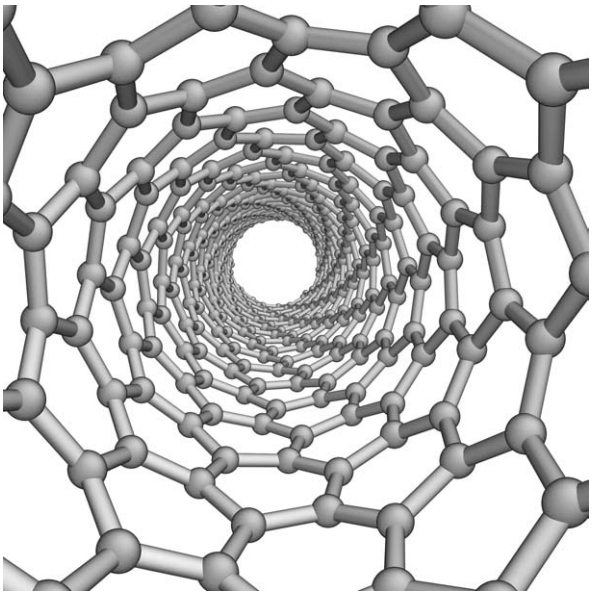

Figure 2. Inside a chiral single-walled carbon nanotube.

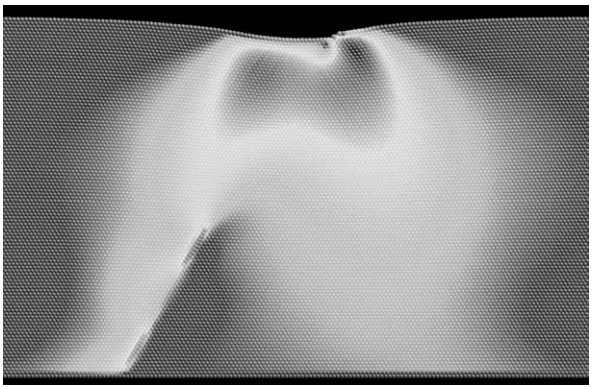

Figure 3. Dislocation emission in a two-dimensional bubble raft under a spherical indentor [24]. The color encoding of atoms is by the auxiliary property of local atomistic von Mises stress invariant.

pressed pulls the viewpoint away or closer from the anchor. Rotation is always done such that the anchor position is invariant in the field of view. At beginning, the anchor is taken to be the center of mass. This allows for global view of the configuration by rotating with mouse or with arrow keys (see below). When one right-clicks on an atom or a bond, the anchor is transferred to that particular atom or bond. So if one is interested in a closer view of a

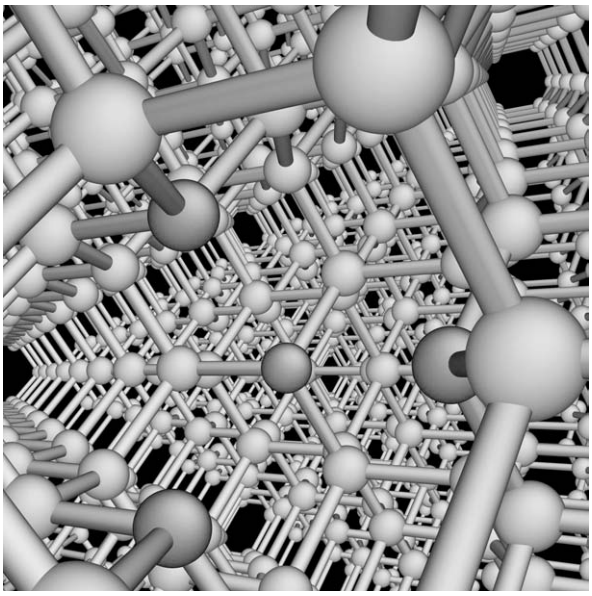

Figure 4. A vacancy defect in silicon. Three-fold coordinated atoms are colored green, while 4-fold coordinated atoms are colored silver.

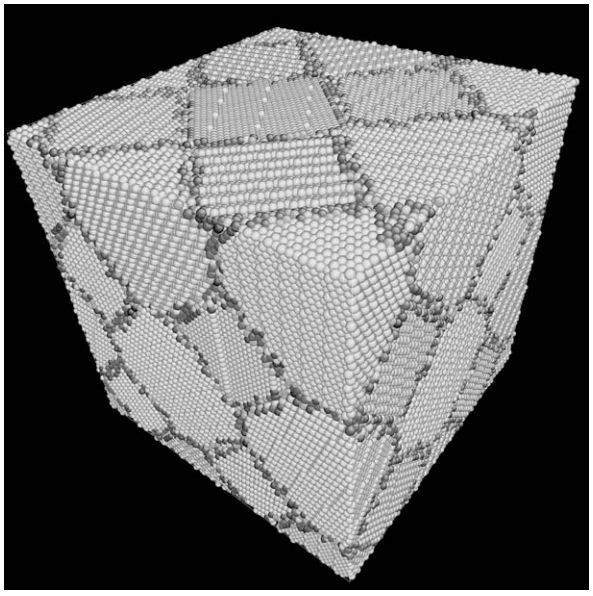

Figure 5. Cu nanocrystal configuration consisting of 424 601 atoms. Atom coloring is by coordination number.

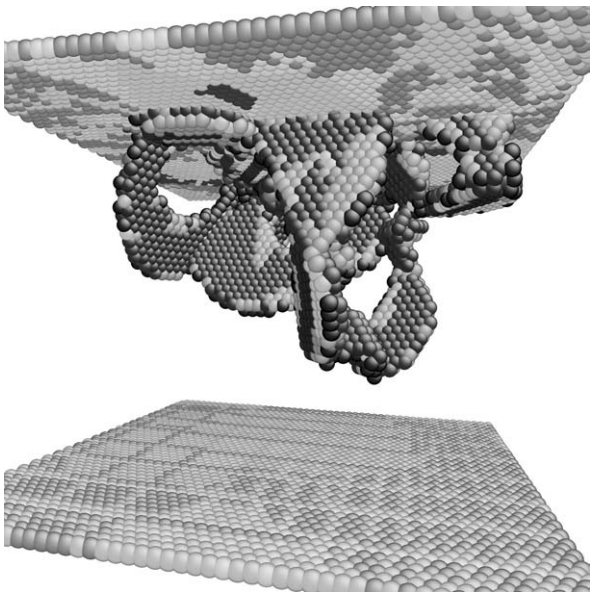

Figure 6. Central symmetry color encoding showing intrinsic stacking faults bounded by partial dislocations during indentation of Cu.

particular atomic local environment, one right-clicks on an atom or bond and then drags the mouse down without releasing the right mouse button. To pull away, simply right-click on a vacuum region and drag the mouse up without releasing the right mouse button. One can always recover the center of mass anchor by pressing key "w".

Rotation by mouse movement is accomplished under the following concepts: there is a glass sphere about half the viewport size hinged at the center of the viewport. The configuration is "frozen" in the glass sphere and corotates with it. After the rotation, there is a compensating translation if necessary, to fix the anchor in the viewport. To rotate, one imagines putting a finger on the glass sphere surface and move the fingertip, which is accomplished by left-clicking in the window and dragging the mouse without releasing the left button. The remainder of the viewport comprises of a flat glass surface parallel to the viewport, left-clicking and dragging which causes the configuration to rotate clockwise or counterclockwise. By pressing the arrow keys $\leftarrow$, $\rightarrow$, $\uparrow$, $\downarrow$, and shift$+\uparrow$, $\downarrow$, the configuration can also be rotated along three orthogonal axes. The rate of rotation is governed by the gearbox value, that controls all rates of changes, and can be varied by pressing the numeric keys 0–9. One can always recover the initial view frustum orientation with x, y, z perfectly aligned, by pressing key "u".

At this point we need to explain the design of the CFG configuration file format which AtomEye supports. (Though there is rudimentary support for the PDB file format [14], PDB is not recommended. See the last section.) In the CFG file, one always assumes that the configuration is under PBC, with a parallelopiped supercell defined by its three edge vectors (not necessarily orthogonal to each other). The reason for enforcing the PBC requirement is that, while it is quite easy to express a cluster configuration as a PBC configuration by putting a large enough PBC box around it, therefore separating the periodic images by vacuum, it is not so easy the other way around. To define a PBC configuration, a minimum of $3N + 9$ real numbers needs to be supplied, where N is the number of atoms. First, one must specify a 3×3 matrix,

$$\mathbf{H} = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix}, \tag{1}$$

in the unit of angstrom (Å), which specifies the supercell size and shape. AtomEye uses a row-based vector notation. That is, the first row of the $\mathbf{H}$ matrix corresponds to the first edge (or basis) vector $\mathbf{h}_1$ of supercell, and similarly for $\mathbf{h}_2$ and $\mathbf{h}_3$:

$$\mathbf{h}_1 \equiv (H_{11}\ \ H_{12}\ \ H_{13}), \quad \mathbf{h}_2 \equiv (H_{21}\ \ H_{22}\ \ H_{23}), \quad \mathbf{h}_3 \equiv (H_{31}\ \ H_{32}\ \ H_{33}). \tag{2}$$

So, for instance, H_{23} is the z-component of the second edge vector of the supercell (in Å). It is recommended that $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ constitute a right-handed system, that is

$$(\mathbf{h}_1 \times \mathbf{h}_2) \cdot \mathbf{h}_3 = \det(\mathbf{H}) > 0, \tag{3}$$

but it is not required.

The atom positions are specified in the CFG file by the socalled reduced coordinates $\{\mathbf{s}_i\}$ instead of the Cartesian coordinates $\{\mathbf{x}_i\}$. Here i runs from 1 to N (in the program it actually runs from 0 to $N - 1$), and both $\mathbf{s}_i$ and $\mathbf{x}_i$ are 1×3 row vectors

$$\mathbf{s}_i \equiv (s_{i1}\ s_{i2}\ s_{i3}), \quad \mathbf{x}_i \equiv (x_i\ y_i\ z_i). \tag{4}$$

s_{i1}, s_{i2}, s_{i3} are called reduced coordinates since

1. They are dimensionless, unlike x_i, y_i, z_i which are in Å.
2. They are all between 0 and 1:

$$0 \le s_{i1} < 1, \quad 0 \le s_{i2} < 1, \quad 0 \le s_{i3} < 1. \tag{5}$$

$\mathbf{x}_i$ and $\mathbf{s}_i$ are related by the matrix-vector product

$$\mathbf{x}_i = \mathbf{s}_i\mathbf{H} = s_{i1}\mathbf{h}_1 + s_{i2}\mathbf{h}_2 + s_{i3}\mathbf{h}_3. \tag{6}$$

Since $\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_3$ are the three edges of the parallelopiped supercell, it is seen that any point inside the supercell corresponds to s_{i1}, s_{i2}, $s_{i3} \in [0, 1)$, and vice versa. Any *image* atom outside of the supercell can be expressed as $(s_{i1} + l, s_{i2} + m, s_{i3} + n)$, in which l, m, n are all integers, which is separated from the original atom $\mathbf{x}_i$ by Cartesian distance $l\mathbf{h}_1 + m\mathbf{h}_2 + n\mathbf{h}_3$.

Knowing $\mathbf{x}_i$ and $\mathbf{H}$, one can also invert Eq. (6) to get $\mathbf{s}_i$

$$\mathbf{s}_i = \mathbf{x}_i \mathbf{H}^{-1}. \tag{7}$$

If any of $s_{i1}, s_{i2}, s_{i3} \notin [0, 1)$, the atom is outside of the supercell (i.e., it is an image atom) and needs to be mapped back to the original supercell, by

$$\mathbf{s}_{i\alpha} \;\rightarrow\; \mathbf{s}_{i\alpha} - \lfloor \mathbf{s}_{i\alpha} \rfloor, \quad \alpha = 1, 2, 3, \tag{8}$$

where $\lfloor \cdot \rfloor$ is the floor function, returning the largest integer not greater than the argument.

The reciprocal vectors of the supercell $\mathbf{g}_1$, $\mathbf{g}_2$, and $\mathbf{g}_3$ are the first, second and third row vectors of the matrix

$$\mathbf{G} \equiv 2\pi (\mathbf{H}^{-1})^{\mathrm{T}}, \tag{9}$$

and satisfy the fundamental relations

$$\mathbf{g}_\alpha \mathbf{h}_\beta^T = 2\pi \delta_{\alpha\beta}, \quad \alpha, \beta \in 1 \cdots 3 \tag{10}$$

Since $\mathbf{g}_1$ is normal to the plane spanned by $\mathbf{h}_2$ and $\mathbf{h}_3$, $\mathbf{g}_2$ is normal to the plane spanned by $\mathbf{h}_1$ and $\mathbf{h}_3$, $\mathbf{g}_3$ is normal to the plane spanned by $\mathbf{h}_1$ and $\mathbf{h}_2$, it is easy to see that the thicknesses of the supercell perpendicular to the three sets of planes are

$$d_1 = \frac{2\pi}{|\mathbf{g}_1|}, \quad d_2 = \frac{2\pi}{|\mathbf{g}_2|}, \quad d_3 = \frac{2\pi}{|\mathbf{g}_3|}, \tag{11}$$

respectively. It can be shown that a sphere of radius R can fit into *one* supercell (without touching any of the six faces) if and only if

$$2R < \min(d_1, d_2, d_3). \tag{12}$$

The above is important because it tells us whether a great simplification in treating image interactions can be taken or not.

To appreciate this, let us suppose two atoms would interact or consider each other their neighbor whenever their distance is less than r_{c}. Given the contents of the supercell, the physical system it represents is an infinite lattice composed of infinitely tiled replicas of the original supercell. Theoretically, to determine how many neighbors an atom $\mathbf{x}_i$ in the original supercell has, one needs to go over all atoms in nearby supercells. It is then possible that both $\mathbf{x}_j + l\mathbf{h}_1 + m\mathbf{h}_2 + n\mathbf{h}_3$ and $\mathbf{x}_j + l'\mathbf{h}_1 + m'\mathbf{h}_2 + n'\mathbf{h}_3$ are neighbors of $\mathbf{x}_i$, which is called *multiple counting*. There is nothing wrong with multiple counting,

but this possibility makes the program more complicated and less efficient. So a natural question is, under what conditions would multiple counting be guaranteed to *not* occur, and one only has *single counting*? In other words, when would any two atoms i and j in the original supercell have *at most one* interaction even when all images of j are taken into account? To figure this out, suppose $\mathbf{x}_i$ is at the center of the original parallelopiped, $\mathbf{s}_i = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$. It is then seen that if and only if

$$2r_c < \min(d_1, d_2, d_3), \tag{13}$$

can single counting be *guaranteed* for atom i, and all possible neighbors are within the original supercell. One then realizes that this criterion does not really care where atom i is. One can always define a shifted supercell (possibly containing some image atoms) with atom i at its center, that has one-to-one correspondence with atoms $1 \cdots N$ in the original supercell. So long as Eq. (13) is satisfied, one only needs to loop over atoms $1 \cdots N$ *once* to find out *all* the neighbor of i, according to the formulas

$$\mathbf{S}_{ij} \equiv (S_{ij1}\ \ S_{ij2}\ \ S_{ij3}), \quad S_{ij\alpha} = S_{i\alpha} - S_{j\alpha} - \left\lfloor S_{i\alpha} - S_{j\alpha} + \frac{1}{2} \right\rfloor, \quad \alpha = 1, 2, 3 \tag{14}$$

$$\mathbf{x}_{ij} \equiv \mathbf{s}_{ij}\mathbf{H}, \quad r_{ij} \equiv |\mathbf{x}_{ij}|. \tag{15}$$

In the engine of AtomEye, condition (13) is assumed to hold, which is often the case for configurations involved in empirical potential calculations. However, configurations in *ab initio* calculations often do not satisfy (13). So, when AtomEye loads in the configuration, if (13) is found to be unsatisfied, the configuration is automatically replicated in the necessary direction(s) to satisfy condition (13).

In the CFG file, the $\mathbf{H}$ matrix can be specified flexibly according to the following formula

$$\mathbf{H} = A\mathbf{H}_0\sqrt{\mathbf{I} + 2\boldsymbol{\eta}}\mathbf{T}, \tag{16}$$

where A, $\boldsymbol{\eta}$ and $\mathbf{T}$ are optional parameters, and $\mathbf{I}$ is the 3×3 identity matrix. A is a scalar and has the meaning of the basic lengthscale of the configuration in Å, and its default value is unity. $\boldsymbol{\eta}$ is a desired Lagrangian strain which is a 3×3 symmetric matrix, and $\sqrt{\mathbf{I} + 2\boldsymbol{\eta}}$ is the affine transformation matrix that achieves $\boldsymbol{\eta}$ without rotation (see Chap. 2.19); by default, $\boldsymbol{\eta}$=0, the zero matrix. Finally, $\mathbf{T}$ is an affine transformation matrix, which can contain a rotational component; by default, $\mathbf{T} = \mathbf{I}$. When A, $\boldsymbol{\eta}$ and $\mathbf{T}$ all take their default values, $\mathbf{H} = \mathbf{H}_0$. So if one does not care about scaling and affine transformations, one can just specify $\mathbf{H}$ by directly specifying $\mathbf{H}_0$ in Å, like

$$\mathbf{H}_0 = \begin{pmatrix} 1.8075 & 1.8075 & 0 \\ 1.8075 & 0 & 1.8075 \\ 0 & 1.8075 & 1.8075 \end{pmatrix}, \tag{17}$$

for FCC Cu primitive cell with equilibrium lattice constant 3.615 Å. However, it is perhaps better to set

$$\mathbf{H}_0 = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}, \tag{18}$$

but set $A = 3.615$. This way, if we want to create a series of configurations with varying lattice parameters, we only need to change *one* number in the CFG file. Optional $\boldsymbol{\eta}$ and $\mathbf{T}$ matrixes are supported for the same reason. If we want to deform the entire configuration, we only need to change one or few parameters in the CFG file. For instance,

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{19}$$

means effecting a simple shear such that $\mathbf{e}_y \rightarrow \mathbf{e}_y + 0.5\mathbf{e}_x$ with $\mathbf{e}_x$ and $\mathbf{e}_z$ unchanged.

The main data block comes after various required and optional declarations. There are N lines in the data block, one line for each atom. The first three entries on each line are the s_{i1}, s_{i2}, s_{i3} of that atom. Depending on the declaration, there may or may not be three numbers following them that contain the velocity information. The CFG file is extensible in the sense that there is a supported way for the user to store extra atomic properties in the CFG file. For example, one may wish to store the instantaneous force on each atom, along with the positions. To do this, one can declare the existence of three *auxiliary* properties

$$\begin{aligned} &\text{auxiliary}[0] = fx[\text{eV}/\text{Å}] \\ &\text{auxiliary}[1] = fy[\text{eV}/\text{Å}] \\ &\text{auxiliary}[2] = fz[\text{eV}/\text{Å}], \end{aligned}$$

that provide indexing (start from 0), property name, and unit information. One then appends the auxiliary property data at the end of the line for each atom. AtomEye can be used to interrogate atom by atom and to graphically represent these auxiliary properties with various threshold and colormap options (see Fig. 3).

The CFG file (with recommended suffix ".cfg") is meant to be readable and editable by people, so it is in plain ASCII format. One can add comments after "#", which is also a way to store nonstandard information. All data values can be specified to an arbitrary number of digits that the user deems necessary. To compensate for the large size, the user may compress the CFG file using gzip (recommended suffix ".cfg.gz") or bzip2 (recommended suffix ".cfg.bz2"). AtomEye can directly load in the compressed files, using an

automatic recognition and decompression scheme. To simplify handling, one CFG file should store one atomistic configuration only. A sequence of configurations should be named like "mdrun00001.cfg.gz", "mdrun00002.cfg.gz", "mdrun00003.cfg.gz", ..., etc., with the starting identifier "mdrun" arbitrary. AtomEye can recognize the above file name patterns automatically to determine a file group, with browsing forward, backward and loop-back capabilities. This greatly facilitates inspecting MD trajectories.

AtomEye presently has three builtin functions to characterize the local atomic environment

1. Coordination number $\{k_i\}$. This counts the total number of first-nearest neighbors each atom i has in a configuration (self excluded). Physically it is of course a fuzzy concept, especially in liquids, where the sharp shell structure of the crystal reference is largely smeared out. Procedure wise, what is done in AtomEye is that there is a default radius value R_u defined for each element species u. An atom i of species u would consider an atom j of species v its first-nearest neighbor if their distance r_{ij} (see Eq. (15)) is less than $r_{\mathrm{c},uv} \equiv R_u + R_v$. By the above definition, and by common sense, this relationship is reciprocal: that is, if atom i considers atom j its first-nearest neighbor, then atom j would also consider atom i its first-nearest neighbor.

   The choice of the R_u default value is based on the following considerations. The first is Slater's empirical atomic radius tabulation based on the equilibrium bond lengths in over 1200 ionic, metallic, and covalent crystals and molecules [25]. The second is that in order for the procedure to be maximally resistant to thermal noise at low T for the ground-state phase perfect crystal, $2R_u$ should be set to approximately halfway between the first and the second atomic shells of the $T = 0$ perfect crystal. (In liquids a similar choice would be to set $2R_u$ to the locate of the minimum between the first and second maxima in the radial distribution function $g(r)$.) This default $r_{\mathrm{c},uv}$ value however does not always work well in practice, and the user can change it.

   In AtomEye, $\{k_i\}$ is used as a versatile characterization of atomic defects. Point defects (Fig. 4), dislocations, grain boundaries (Fig. 5), etc. will often change the coordination number of atoms in their cores, thereby allowing their conformations to be visualized. Often, to see the defects, one also needs to render the "uninteresting" atoms invisible. Here, the uninteresting atoms are identified as those whose k_i remains unchanged from the reference crystal value, such as 12 in FCC crystal. Ctrl+shift+right-click on them will make them invisible.
2. Central symmetry parameter $\{c_i\}$. There are some important defects in crystals, such as stacking faults and twin boundaries, that do not change the coordination number of atoms. But they can be identified by

evaluating the degree of inversion symmetry breaking around each atom. This is explained in the next section. An example is shown in Fig. 6, where intrinsic stacking faults bound by Shockley partial dislocations in FCC crystals are visualized.

3. Local von Mises shear strain invariant $\{\eta_i\}$. A reference-state free measure of local atomic strain shear invariant has been derived for high-symmetry crystals [26].

Furthermore, the user is free to devise his/her own local environment characterization scheme, save the result as an auxiliary property (see Fig. 3) to visualize it later on. One may also define a "color patch" file that accompanies a CFG file to explicitly control how the atoms should be rendered. AtomEye provides a suite of commands to survey and interrogate the configuration. One can find out about atomic properties (auxiliaries included), bond length, bond angle, surface normal, and dihedral angle by right-clicking on the atoms. One may define a large number of simultaneous cutting planes, and shift the configuration under PBC to expose the most interesting features. Finally, one can put down color marking on the atoms in one configuration and track their diffusive or displacive motion in the ensuing configurations, for example during deformation.

3. Central Symmetry Parameter

The central symmetry parameter $\{c_i\}$, $i = 1 \cdots N$ is used to characterize the degree of inversion symmetry breaking in each atom's local environment. Especially, it is useful for visualizing planar faults in FCC and BCC crystals [27]. We illustrate here how it is done.

Define integer constant M to be the maximum number of neighbors for the computation of $\{c_i\}$. For FCC lattice, we may want to use $M = 12$. For BCC lattice, we may want to use $M = 8$. The computer of course does not know whether the configuration is FCC- or BCC-based, so by default it is going to use,

$$M_{\text{default}} \equiv \left\lfloor \frac{k_{\text{most}}}{2} \right\rfloor \times 2, \tag{20}$$

where k_{most} is the most popular coordination number in the set $\{k_i\}$, $i = 1 \cdots N$ of the configuration. The user is able to override the default. But in any case, M must be even as we will be counting *pairs* of atoms.

Now for each atom $i \in 1 \cdots N$, define,

$$\tilde{m}_i \equiv \min(M, k_i). \tag{21}$$

If $\tilde{m}_i = 0$, $c_i \equiv 0$ since an isolated atom should have perfect inversion symmetry. If $\tilde{m}_i = 1$, $c_i \equiv 1$, since a coordination-1 atom has no inversion image

to compare with, so in a sense its inversion symmetry is the most broken. For $\tilde{m}_i \geq 2$, define,

$$m_i \equiv \left\lfloor \frac{\tilde{m}_i}{2} \right\rfloor \times 2, \tag{22}$$

and we use the following procedure to determine c_i.

1. Sort the $j = 1 \cdots k_i$ neighbors of atom i according to their distances $|\mathbf{d}_j|$ to atom i in ascending order. Pick the smallest m_i-set.
2. Take the closest neighbor $\mathbf{d}_1$. Search, among the other $m_i - 1$ neighbors, the one that minimizes,

$$\tilde{D}_j \equiv \left|\mathbf{d}_1 + \mathbf{d}_j\right|^2, \tag{23}$$

and let us define,

$$j' \equiv \arg \min_{j=2..m_i} \tilde{D}_j, \quad D_1 \equiv \tilde{D}_{j'}. \tag{24}$$

3. Throw atoms 1 and j' out of the set, and look for the closest neighbor in the remaining set. Then repeat Step 2 until the set is empty. We then have obtained $D_1, D_2, .., D_{m_i/2}$. Define,

$$c_i \equiv \frac{\sum_{k=1}^{m_i/2} D_k}{2\sum_{j=1}^{m_i} |\mathbf{d}_j|^2}. \tag{25}$$

Equation (25) is dimensionless. In the case of $m_i = 2$, suppose the two neighbors are independently randomly oriented, it is easy to show that the mathematical expectation,

$$E[c_i] = \frac{1}{2}. \tag{26}$$

On the other hand, we can prove that,

$$\max_{\{\mathbf{d}_j\}} c_i = 1, \tag{27}$$

so this matches with the definition of $c_i \equiv 1$ at $\tilde{m}_i = 1$. But when $m_i \gg 2$,

$$E[c_i] < \frac{1}{2}, \tag{28}$$

because of the minimization process. For instance, at the intrinsic stacking fault in FCC lattice ABC|BCA, there is a loss of inversion symmetry in the two layers C|B, and c_i is,

$$c_i = \frac{3 \times 0 + 3 \times (d\sqrt{3}/2 \times 1/3 \times 2)^2}{2 \times 12d^2} = \frac{1}{24} \approx 0.0416, \tag{29}$$

assuming perfect stacking.

The good thing about expression (25) is that according to the Lindemann/ Gilvarry rule [28], a crystal melts when the atomic vibrational amplitudes reach about $\sim$12% of the nearest neighbor distance, so c_i for perfect crystal should be < 0.01 even at finite temperature. Therefore, it is not very difficult to threshold out thermal noise vs a true stacking fault.

4. Outlook

Atomistic visualization will become more widespread as suitable techniques are developed and software tools are refined. In the future, we expect distributed visualization of large data sets, like distributed number-crunching on Beowulf clusters and grid computers, to become prevalent. In this paradigm, the display node takes care of assembling the scenes and user input, while multiple nodes on a fast network perform data readout and render the scenes in the background, for real-time navigation.

References

[1] F.F. Abraham, R. Walkup, H.J. Gao, M. Duchaineau, T.D. De la Rubia, and M. Seager, "Simulating materials failure by using up to one billion atoms and the world's fastest computer: work-hardening," *Proc. Natl. Acad. Sci. USA.*, 99, 5783–5787, 2002.

[2] P. Vashishta, R.K. Kalia, and A. Nakano, "Multimillion atom molecular dynamics simulations of nanostructures on parallel computers," *J. Nanopart. Res.*, 5, 119–135, 2003.

[3] S. Xu, J. Li, C. Li, and F. Chan, "Immersive visualisation of nano-indentation simulation of cu," In: H. Lee and K. Kumar (eds.), *Recent Advances in Computational Science and Engineering*, World Scientific, Singapore. Proceedings of the International Conference on Scientific and Engineering Computation (IC-SEC) ISBN: 1-8609, 2002.

[4] A. Sharma, A. Nakano, R.K. Kalia, P. Vashishta, S. Kodiyalam, P. Miller, W. Zhao, X.L. Liu, T.J. Campbell, and A. Haas, "Immersive and interactive exploration of billion-atom systems," *Presence-Teleoper. Virtual Env.*, 12, 85–95, 2003.

[5] P.J. Kraulis, "Molscript - a program to produce both detailed and schematic plots of protein structures," *J. Appl. Crystallogr.*, 24, 946–950, 1991.

[6] R.A. Sayle and E.J. Milner-White, "Rasmol – biomolecular graphics for all," *Trends Biochem. Sci.*, 20, 374–376, 1995.

[7] E.A. Merritt and M.E.P. Murphy, "Raster3D photorealistic molecular graphics," *Acta Crystallogr. Sect. D-Biol. Crystallogr.*, 50, 869–873, 1994.

[8] E.A. Merritt and D.J. Bacon, "Raster3D: photorealistic molecular graphics," *Methods Enzymol.*, 277, 505–524, 1997.

[9] M.C. Lawrence and P. Bourke, "CONSCRIPT: a program for generating electron density isosurfaces for presentation in protein crystallography," *J. Appl. Crystallogr.*, 33, 990–991, 2000.

[10] N. Guex and M.C. Peitsch, "SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modeling," *Electrophoresis*, 18, 2714–2723, 1997.
[11] N. Guex, A. Diemand, and M.C. Peitsch, "Protein modelling for all," *Trends Biochem. Sci.*, 24, 364–367, 1999.
[12] R. Koradi, M. Billeter, and K. Wuthrich, "MOLMOL: A program for display and analysis of macromolecular structures," *J. Mol. Graph.*, 14, 51–55, 1996.
[13] O. Miyashita, J.N. Onuchic, and P.G. Wolynes, "Nonlinear elasticity, proteinquakes, and the energy landscapes of functional transitions in proteins," *Proc. Natl. Acad. Sci. USA*, 100, 12570–12575, 2003.
[14] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne, "The protein data bank," *Nucleic Acids Res.*, 28, 235–242, 2000.
[15] M. Parrinello and A. Rahman, "Polymorphic transitions in single-crystals – a new molecular dynamics method," *J. Appl. Phys.*, 52, 7182–7190, 1981.
[16] J. Li, "Atomeye: an efficient atomistic configuration viewer," *Model. Simul. Mater. Sci. Engrg.*, 11, 173–177, 2003.
[17] G. Schaftenaar and J.H. Noordik, "Molden: a pre- and post-processing program for molecular and electronic structures," *J. Comput.-Aided Mol. Des.*, 14, 123–134, 2000.
[18] A. Kokalj, "Xcrysden – a new program for displaying crystalline structures and electron densities," *J. Mol. Graph.*, 17, 176, 1999.
[19] A. Kokalj, "Computer graphics and graphical user interfaces as tools in simulations of matter at the atomic scale," *Comput. Mater. Sci.*, 28, 155–168, 2003.
[20] W. Humphrey, A. Dalke, and K. Schulten, "VMD: visual molecular dynamics," *J. Mol. Graph.*, 14, 33–38, 1996.
[21] J. Adler, A. Hashibon, N. Schreiber, A. Sorkin, S. Sorkin, and G. Wagner, "Visualization of md and mc simulations for atomistic modeling," *Comput. Phys. Commun.*, 147, 665–669, 2002.
[22] S.R. Bahn and K.W. Jacobsen, "An object-oriented scripting interface to a legacy electronic structure code," *Comput. Sci. Engrg.*, 4, 56–66, 2002.
[23] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice in C*, 2nd edn., Addison-Wesley, Reading, 1995.
[24] J. Li, K.J. Van Vliet, T. Zhu, S. Yip, and S. Suresh, "Atomistic mechanisms governing elastic limit and incipient plasticity in crystals," *Nature*, 418, 307–310, 2002.
[25] J. Slater, *J. Chem. Phys.*, 39, 3199, 1964.
[26] J. Li, *To be published*, 2005.
[27] C.L. Kelchner, S.J. Plimpton, and J.C. Hamilton, "Dislocation nucleation and defect structure during surface indentation," *Phys. Rev. B*, 58, 11085–11088, 1998.
[28] J. Gilvarry, "The lindemann and gruneisen laws," *Phys. Rev.*, 102, 308, 1956.