Structure, Vol. 13, 473–482, March, 2005, ©2005 Elsevier Ltd All rights reserved. DOI 10.1016/j.str.2005.01.006

Software Extensions to UCSF Chimera for Interactive Visualization of Large Molecular Assemblies

Thomas D. Goddard, Conrad C. Huang, and Thomas E. Ferrin[1,*]
Department of Pharmaceutical Chemistry
University of California, San Francisco
San Francisco, California 94143

Summary

Many structures of large molecular assemblies such as virus capsids and ribosomes have been experimentally determined to atomic resolution. We consider four software problems that arise in interactive visualization and analysis of large assemblies: how to represent multimers efficiently, how to make cartoon representations, how to calculate contacts efficiently, and how to select subassemblies. We describe techniques and algorithms we have developed and give examples of their use. Existing molecular visualization programs work well for single protein and nucleic acid molecules and for small complexes. The methods presented here are proposed as features to add to existing programs or include in next-generation visualization software to allow easy exploration of assemblies containing tens to thousands of macromolecules. Our approach is pragmatic, emphasizing simplicity of code, reliability, and speed. The methods described have been distributed as the Multiscale extension of the UCSF Chimera (www.cgl.ucsf.edu/chimera) molecular graphics program.

Introduction

Advances in electron cryomicroscopy and X-ray crystallography have enabled structures of many cellular machines to be determined. Structures such as ribosomes, virus capsids, molecular motors, cytoskeletal filaments, proteasomes, nucleosomes, chaperonins, transmembrane channels, and pumps, composed of tens to thousands of macromolecules, have been determined to atomic resolution. The inventory of large molecular assemblies is growing rapidly. At the start of 2000, the Protein Data Bank (PDB) (Berman et al., 2000) contained 137 structures with 10 or more chains, while at the start of 2004, there were 517 such structures. Existing software to interactively display and analyze macromolecules is difficult to use on these large assemblies.

A published study of viral RNA bound to the outside of bluetongue virus capsid (Diprose et al., 2002) illustrates several limitations with existing interactive molecular analysis programs when applied to large assemblies (Figure 1). The icosahedral capsid contains 3 million atoms and cannot be opened in existing desktop analysis software due to insufficient memory. The capsid can be described by using only 50,000 atom positions and the 60-fold icosahedral symmetry, but most existing software is not able to read and use these symmetry matrices. Current low-resolution display styles such as ribbons and molecular surfaces are too detailed for making an informative display of this large system, and the excessive detail prevents smooth interactive rotation of the model because of insufficient graphics rendering speed. The atomic contacts of viral RNA with the capsid are of interest for this system. The simplest algorithm for finding these contacts, calculating distances between all pairs of atoms, requires several minutes of computation and thus is not fast enough for interactive exploration of the contacts. In summary, analysis of this large system runs into problems of limited computer resources that require better data representations and algorithms and new features such as low-resolution display styles.

Virus capsids are the largest structures in the PDB. All other PDB structures of cellular machinery are significantly smaller in number of atoms, and they can be opened in current-generation molecular analysis programs. Yet, software designed for small complexes is cumbersome to use on these larger systems. An example is provided by a recent paper describing the interactions of 27 ribosomal proteins with the 23S ribosomal RNA (Klein et al., 2004) (Figure 2). Each of the intermolecular interfaces between the 27 proteins and the 101 helices of the 23S RNA are individually analyzed and illustrated in that work, and differences between two different organisms are noted. Interactively exploring this large set of interfaces would benefit from the ability to switch views from the full assembly to individual proteins and neighboring helices with minimum effort. Low-resolution depiction of the whole assembly and methods of treating the 101 helices comprising the ribosomal RNA as natural structural units are useful capabilities for facile navigation of this assembly.

The virus and ribosome studies described above were carried out without the benefit of interactive analysis software designed for handling these large complexes. Such studies can be done laboriously with software designed for small complexes, as well as with noninteractive software and custom-written analysis programs. For virus capsids, files containing small pieces with just the needed asymmetric units can be created with specialized programs and then explored interactively. For the helices of the 23S rRNA, many exact residue sequence ranges can be entered by hand to specify regions of interest during interactive analysis. Software that provides more efficient ways of exploring large assemblies will speed up the elucidation of the functions of these systems.

Past efforts to develop analysis software for large assemblies have addressed some of the limitations. A study done 6 years ago (Macke et al., 1998), when many fewer experimental structures were available, fo-

*Correspondence: tef@cgl.ucsf.edu
[1]Lab address: http://www.cgl.ucsf.edu

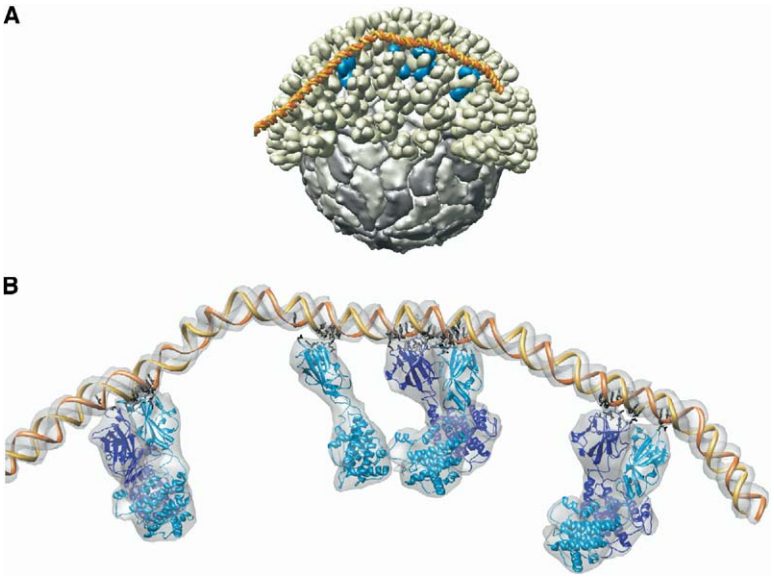

Figure 1. Bluetongue Virus Capsid with Bound Viral RNA

(A) Two protein layers of bluetongue virus capsid with viral RNA bound to the surface (PDB models 2btv and 1h1k). Each molecule is displayed by using a low-resolution surface. RNA contacts seven protein monomers (shown in blue) on the surface. The model contains approximately 3 million atoms (no hydrogens) and 900 molecular components.
(B) Atomic contacts between RNA and capsid proteins within a 5 Å range shown in black.

cused on building molecular assemblies by using symmetry operations and displaying them with variable resolution surfaces using spherical harmonics (Duncan and Olson, 1995). Other work (Bajaj et al., 2004) has explored computer graphics techniques to efficiently display molecular properties at various resolutions by using texture maps on surfaces and volumetric rendering. Another group has developed a web service to apply symmetry matrices to generate atomic coordinates of crystal packings and visualize them (Hussain et al., 2003). Most macromolecular structures are determined by crystallography, so this type of large molecular assembly is common.

The distinctive feature of the work presented here is our focus on software system integration. Many methods exist for analyzing single molecules and small complexes, for example, structure and sequence alignments, molecular dynamics, binding site prediction, electrostatics calculations, hydrogen bond identification, superimposing homologous or mutant structures, testing conformational changes, and fitting experimental density maps. Programs such as UCSF Chimera (Pettersen et al., 2004), PyMol (DeLano, 2002), and VMD (Humphrey et al., 1996) provide many of these capabilities. Analysis of large assemblies requires the same tools to study many local regions in atomic detail. On top of these, better facilities to navigate to small regions of interest, restrict attention to them, and show them in the context of the larger assembly are needed. The main virtue of the tools we have developed and describe here is that they interoperate within a program that provides the rich set of analysis capabilities already created for studying smaller systems. To successfully augment an already complex software package, we have favored simple code, reliable algorithms, and speed. Our aim is to develop the breadth of capabilities necessary for effective analysis of large assemblies, rather than maximal performance of any one software component.

The following sections address the most essential software capabilities needed for exploring large assemblies. We describe how to efficiently represent multimeric molecular complexes in computer memory, an algorithm for creating low-resolution surfaces to represent molecules, an algorithm for finding atomic contacts between large sets of atoms, and a hierarchical

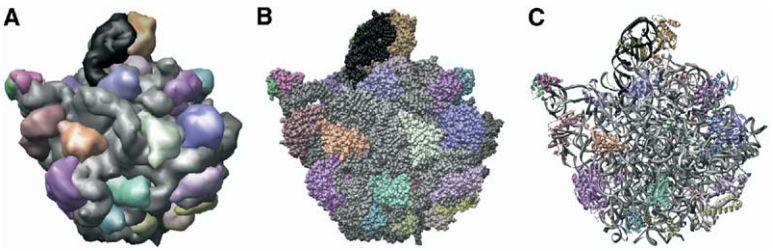

Figure 2. Protein-RNA Contacts in Large Ribosomal Subunit

(A) Arrangement of 27 ribosomal proteins contacting 23S ribosomal RNA (gray) and 5S ribosomal RNA (black) (PDB model 1s72). Each protein and RNA is represented by a low-resolution surface. This coarse view is used to select and focus on individual protein-RNA interactions.
(B) Displaying atoms as spheres creates a pebbly surface that reduces the effectiveness of the three-dimensional cues provided by object illumination techniques.
(C) Ribbon display style permits seeing through the structure. Clear display of protein-RNA interactions requires restricting the view to small pieces of the assembly.

representation of quaternary structure useful for selecting pieces of a large assembly. These components were added to the UCSF Chimera molecular analysis program, primarily targeting visualization and analysis of virus capsids. We also present applications of these tools and discuss additional capabilities useful for interactively exploring large assemblies.

Results and Discussion

Efficient Handling of Multimers

The simplest approach for representing molecular assemblies in a computer is to keep data for every atom in memory. Current molecular graphics software and computer hardware are able to handle structures containing up to a few hundred thousand atoms with this method. For icosahedral and helical virus particles, actin filament networks, and microtubules composed of hundreds or thousands of identical molecules, keeping data for all atoms requires more memory than is available on typical desktop computers. For assemblies containing many copies of structurally identical molecules, it is more efficient to keep atomic data in memory for only unique component molecules, and keep position vectors and orientation matrices for copies of these molecules. The matrices to place molecular components can be read from header information in PDB files. Software that uses this more efficient representation also needs to be able to replicate atomic data in memory to allow the user to make structural modifications to individual molecular components and display different copies of molecules in different ways. The size of systems that can be visualized, and speed in handling structures that already fit in memory, can be significantly increased by not keeping all atom data in memory for multimeric structures.

Keeping data for all atoms of a large molecular assembly requires an impractically large computer memory. Most software for visualizing single molecules or small complexes maintains data for every atom in the structure. Information such as atom name, atom type, a list of covalently connected atoms, Van der Waals radius, B factor, the containing residue, display style, display color, etc. are stored in memory. When the space needed for this data approaches the physical memory size of the computer, there are long delays (minutes) in loading and manipulating the structure. Measuring memory use of the full-featured molecular graphics packages Chimera, PyMol, and VMD shows that 0.5–2 KB of memory is used per atom. Keeping the atomic data in memory for the 3-million-atom bluetongue virus capsid (Figure 1) is impractical on today's desktop computers with typical physical memory sizes of 1 GB.

The PDB provides two types of data files for describing molecular assemblies. The standard PDB files include atomic coordinates for only the asymmetric unit. Matrices describing the biological unit are optionally given in REMARK 350 BIOMT records in the file header. These records specify a rotation matrix followed by a translation to place chains of the asymmetric unit to form the assembly. The rotation and translation are written as a 3 row × 4 column matrix with the rotation in the first three columns and the translation in the last column. Most molecular visualization software is unable to use matrices to show multimeric structures and instead requires coordinates for all atoms of the multimeric form. To meet this need, the PDB provides a second type of file called a biological unit file that contains coordinates for all atoms in the assembly. The biological unit files can be impractically large. For example, bluetongue virus capsid (PDB identifier 2btv) contains approximately 50,000 atoms in the asymmetric unit and 3 million atoms in the biological unit. The biological unit file would be 250 MB in size. For large virus capsids such as this one, the PDB does not provide the biological unit files. For both large and small multimeric structures, it is advantageous for software to read the positioning matrices instead of using all-atom files so that the structurally identical molecular components are easily identified.

To allow different graphical depictions or structural modifications of individual molecular components, software must be able to duplicate atomic data as needed. When copies of a molecular chain are displayed with different colors, or display styles (spheres, sticks, ribbons, etc.), or with different sets of atoms hidden, then separate copies of the graphical information can be created in memory. If modifications of the structure are allowed, such as changing the conformation, adding hydrogens, or making covalent modifications, then nongraphical atomic data can also be duplicated when necessary.

In our implementation, positioning matrices are read from PDB file headers. Copies of atomic-level information are not needed for low-resolution surface display and zone calculations involving molecular chains, but residue-level and atomic-level display (ribbons, spheres, etc.), structure modifications, selecting atoms, and many other operations require creating separate copies. In real analysis, few copies of atomic data for the asymmetric unit of a 60-mer icosahedral virus capsid are typically used.

Low-Resolution Surfaces

Useful displays of large assemblies require low-resolution depictions for most of the component parts (Figure 2). Three disadvantages of high-resolution depictions are that the boundaries between the molecules of the assembly are hard to see, the sense of three-dimensionality is poor, and graphics systems are unable to rotate the models smoothly. Using different colors for neighboring molecules can make boundaries clear, but symmetric assemblies like icosahedral virus particles or chaperonins having 7-fold axial symmetry permit no simple regular coloring pattern. Also, this interferes with other uses of color like classifying structural components. The poor impression of three-dimensionality occurs because depth cues from lighting are less effective when reflective surfaces vary their orientations on a short spatial scale (Figure 2B). To quantify the graphics speed limitation, using a solvent-excluded molecular surface depiction created by the MSMS program (Sanner et al., 1996) to display the bluetongue virus capsid of Figure 1 produces a scene with 27 million triangles. Redrawing while rotating the model requires more than 1 s per frame on current high-end desktop graphics

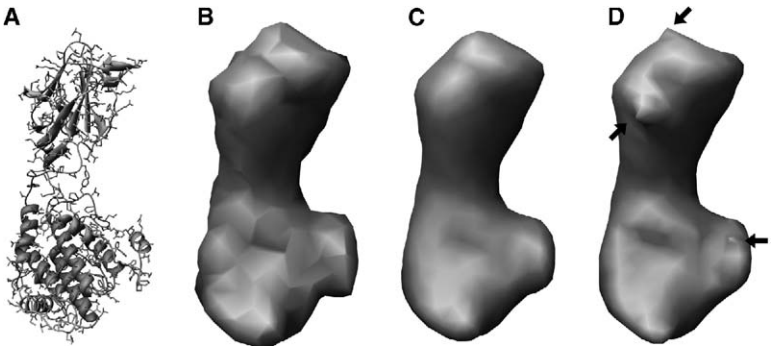

Figure 3. Low-Resolution Surface Example

(A) Ribbon and stick style drawing of a protein monomer from bluetongue virus capsid (PDB model 2btv).

(B) Low-resolution surface at 6 Å resolution, with no smoothing; facet artifacts are visible.

(C) Two iterations of smoothing algorithm, moving vertices half way toward average neighbor position, reduces facet artifacts.

(D) Surface cusp artifacts at 5.3 Å resolution caused by smoothing concentrations of small triangles.

hardware, a factor of 10 too slow for good interactive control.

Low-resolution depictions of assembly components mixed with high resolution in regions of interest avoid the above-mentioned drawbacks. Two types of low-resolution depictions are surfaces that match as closely as possible the shape of the molecule, or caricatures of the shape using simple regular solids such as cylinders, cones, slabs, and polyhedra. The regular solids approach has the potential for easier object recognition. It shows orientations well, but not complementarity of adjacent components, and only offers a single resolution. We developed an algorithm of the shape-matching type, described here, that creates a surface at any desired resolution.

Our surface construction algorithm was designed for practicality rather than highest-quality surface appearance. The practical factors are low-code complexity, robustness, and speed. The essence of the method is to make a spatial density map for a molecule by counting atoms in a three-dimensional grid of bins, then calculating an isosurface, and smoothing it to improve appearance. The most complex code for this algorithm is in calculating the isosurface, and that is often already implemented in molecular display programs for showing density maps from crystallography and electron microscopy. The remaining pieces of the surface construction algorithm are simple to implement. The input for our surface construction is a set of points (atom positions); a resolution, r (length); a density threshold (atoms per unit volume); and two smoothing parameters described below. The output is a set of triangles defining a surface. For each point, the nearest grid point (i*r, j*r, and k*r), where i, j, and k are integers, is found. The number of atoms near each grid point is stored in a three-dimensional array. This gives an atom density map for which we compute an isosurface by using the marching cubes algorithm (Lorensen and Cline, 1987). The isosurface threshold is the density threshold times the resolution cubed.

The isosurface usually appears faceted (Figure 3B). Also, the marching cubes algorithm produces triangle vertices that lie on lines parallel to the x, y, and z axes passing through the discrete grid positions, sometimes making the underlying grid apparent from the pattern of facets. These artifacts give a misleading appearance of structure and undesirable extra detail. To eliminate them, we postprocess the resulting surface by moving each vertex a specified fraction of the way toward the average position of adjoining vertices. All average positions are calculated, and then all vertices are moved so that the order of processing the list of vertices does not matter. This smoothing operation can be performed for a specified number of iterations. The two parameters, the fraction of the distance to move toward the average neighbor position and the number of iterations, determine the result. The smoothing eliminates the faceted appearance (Figure 3C). A drawback is that it sometimes introduces cusp-like features on the surface (Figure 3D). These occur because the smoothing moves vertices with close neighbors by shorter distances than vertices with distant neighbors. The marching cubes algorithm produces small clusters of close vertices where the surface lies close to grid points. The close vertices are relatively immobile, while the surrounding vertices move significantly during smoothing. The presence of cusps depends on the grid spacing (i.e., resolution) and is a minor artifact compared to the facets, but it is an area where the algorithm could be improved. Rotating the input set of atom positions also has a slight effect on the shape of the calculated surface. For assemblies containing identical copies of a molecular chain in different orientations, it is beneficial to calculate the surface only once and use rotated versions of it for the identical units.

A range of surface-calculation parameter values gives useful depictions. We use positions of all atoms, and each atom is given equal weight in the surface calculation. This gives a surface that follows the envelope of the molecule. The threshold value can be used to control whether the surface lies inside (high threshold) or outside (low threshold) of the envelope of the atoms. High-threshold values produce surfaces with space between contacting molecules, while low thresholds cause the surfaces of neighboring molecules to interpenetrate. For calculating surfaces of proteins, a density of 0.02 atoms per cubic angstrom creates a surface close to the envelope. For protein structures with only α carbon positions, a density threshold of 0.001 creates a surface close to the envelope. Resolutions in the 3–15 Å range provide varying levels of detail. Larger values speed up rendering and simplify the appearance for assemblies containing thousands of molecules. A wide range of smoothing parameter values are effective. We generally use two smoothing iterations, each time moving vertices 0.3 of the way toward the average neighbor

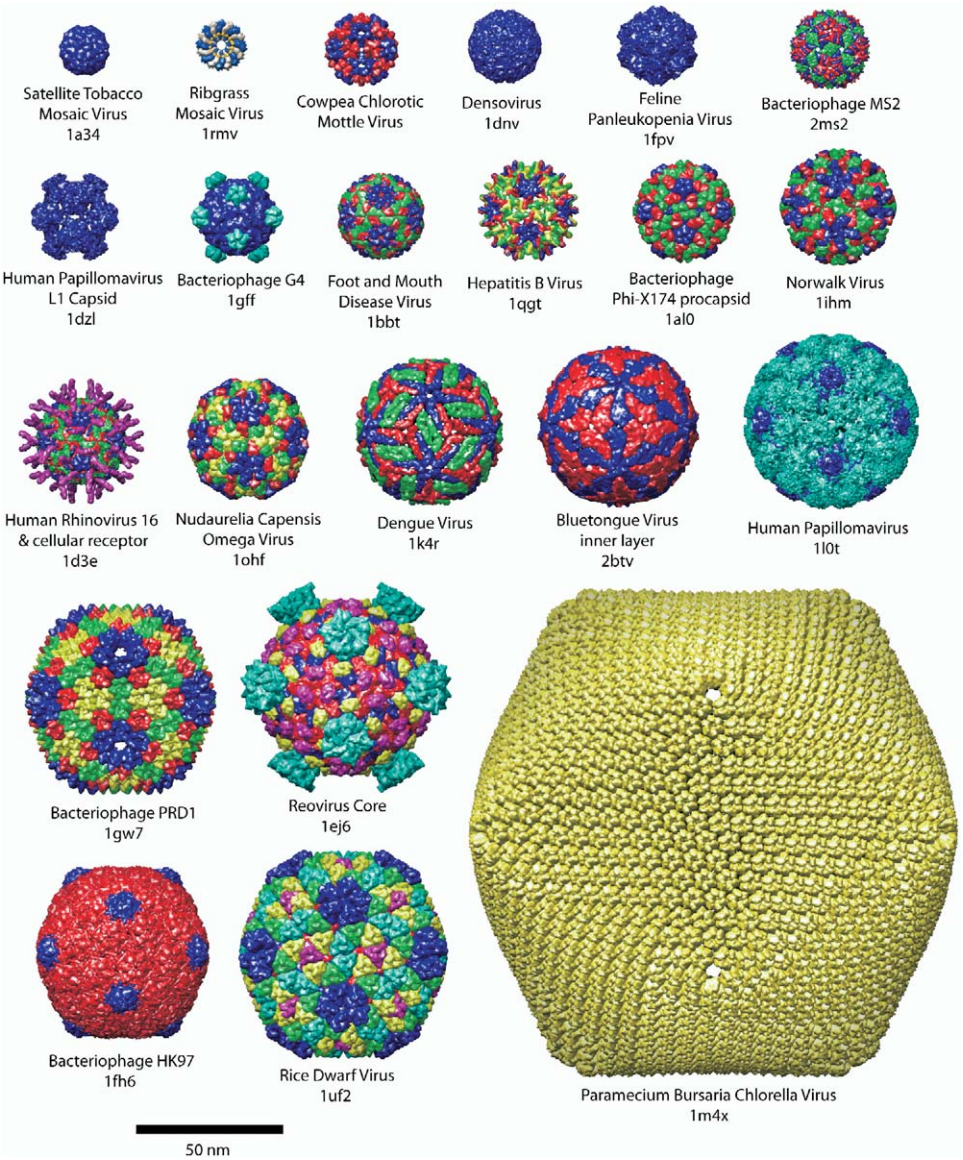

Figure 4. Virus Capsid Architectures

Representative virus capsid architectures from the Protein Data Bank shown to scale. Approximately 230 capsid structures are available, most having icosahedral symmetry, and a few having helical symmetry (second entry top row). PDB identifiers are given below the structures.

position. More smoothing changes the shape of the surface by small amounts, and less smoothing sometimes does not completely remove the faceted appearance.

In our Chimera implementation, each molecule of an assembly is represented by a low-resolution surface of a single color. We do not currently associate atoms and residues with nearby surface vertices, which would permit fine-grained coloring of surfaces to show properties such as residue hydrophobicity or residue conservation. Such colorings are useful (and provided by Chimera) on high-resolution solvent-excluded molecular surfaces and could be useful on the surfaces described here, provided the resolution is high enough to show sufficient spatial detail.

Calculating Atomic Contacts

To study how molecules in an assembly are held together, it is useful to look at intermolecular atomic contacts. The calculation amounts to finding atoms in one set that are close to atoms in another set. When the sets of atoms are large, the naïve method of calculating distances between all possible pairs of atoms from the two sets is not fast enough for interactive analysis. Figure 1 shows an example in which contacts between 17,000 RNA atoms and 3 million virus capsid atoms are desired. Calculating distance squared for the 51 billion atom pairs at a rate of 150 million per second, typical of current desktop computers, requires more than 5 min. A more complex algorithm that we now describe reduces this time for finding all contacts within 5 Å to 2 s.

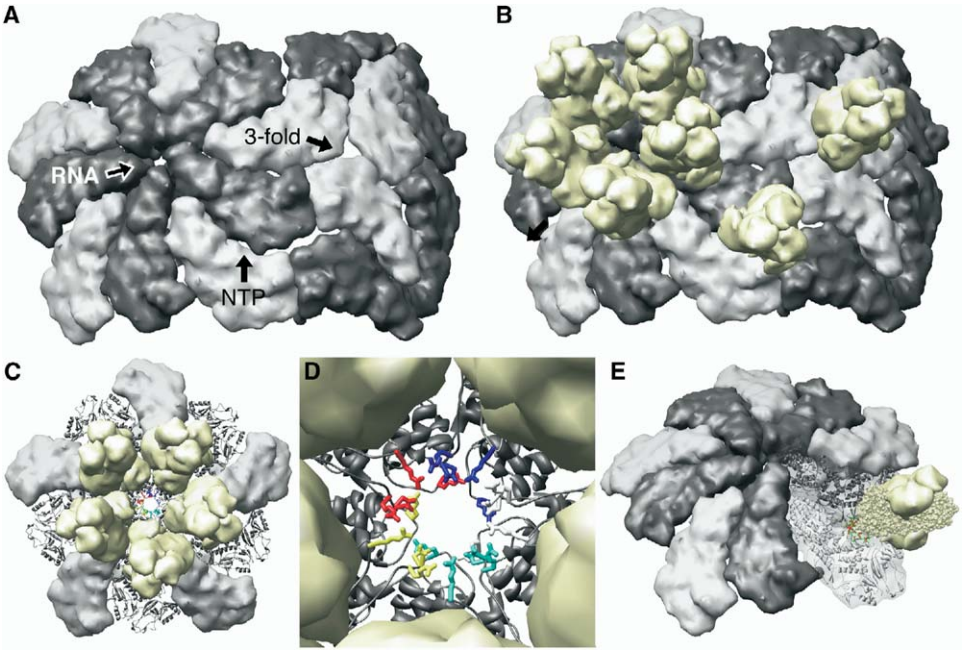

Figure 5. Bluetongue Virus Pores

(A) Portion of the bluetongue virus capsid inner layer showing pores used for intake of metabolites and export of single-stranded RNA replicated within the closed capsid. Arrows indicate the pore at the 5-fold symmetry axis used for exporting ssRNA, the NTP intake pore, and the blocked pore at the 3-fold symmetry axis.

(B) Locations of the capsid outer layer trimers over pores.

(C and D) ssRNA exit pore with the five polar residues colored.

(E) Nucleoside triphosphate intake pore with nearby polar residues colored.

The technique for speeding up the calculation is easily understood with reference to the example of Figure 1. Most of the 980 molecules making up the virus capsid are far away from the RNA compared to the 5 Å range. The minimum and maximum x, y, and z coordinates of the atoms in a capsid molecule can be calculated and compared with those for the RNA strand to quickly determine that the entire capsid molecule is more than 5 Å from the RNA. No distances for individual pairs of atoms need to be calculated. For a capsid molecule of 3,000 atoms and an RNA strand of 17,000 atoms, a total of 20,000 atom positions must be scanned to find the minimum and maximum x, y, and z values. This avoids having to compute distances for 51 million atom pairs. This optimization only eliminates from consideration pairs of atoms that are far apart. It never omits a pair of atoms that are close to one another.

Finding close pairs of points is a problem that occurs in many scientific disciplines; for example, in simulating gravitational interactions of large numbers of stars. The application to molecular interfaces has two special properties that are relevant to choosing an efficient algorithm. First, the sets of points are usually nonoverlapping, the contacts occurring at the surface of the sets. The second property is that a set of points can be composed of many copies of a reoriented and repositioned subset (for example, the 780 copies of the outer layer capsid protein in Figure 1).

The input to our algorithm consists of two sets of points and a distance threshold. Each set of points is specified as a list of subsets, where a subset is an array of {x,y,z} coordinates and a positioning transformation consisting of a rotation matrix and translation vector. The output consists of arrays of indices for each subset giving the contacting points. The output does not return pairs of contacting points. While some analysis would benefit from having the list of pairs rather than lists of individual contact points, the list of pairs can be very large, much larger than the input data. It is trivial to adapt our algorithm to produce that output, but extra precautions, perhaps a specified limit on the pair list size, would be important to avoid running out of memory if a large distance threshold is used. The form of the input avoids the need to calculate and save in memory many repositioned copies of the same subset of points. The same array of {x,y,z} coordinates can be used for multiple subsets with different rotations and translations specified. In many cases, the rotated and translated coordinates are never calculated, as explained below.

The algorithm starts by computing bounding boxes (minimum and maximum x, y, and z coordinates) for each unique subset coordinate array given in the input data. If a coordinate array is used in multiple subsets, its bounding box is only computed once. The algorithm then compares all subsets of set 1 against all subsets of set 2 and eliminates subset pairs in which the bounding boxes are separated by more than the distance threshold. To compute a bounding box for a subset, the

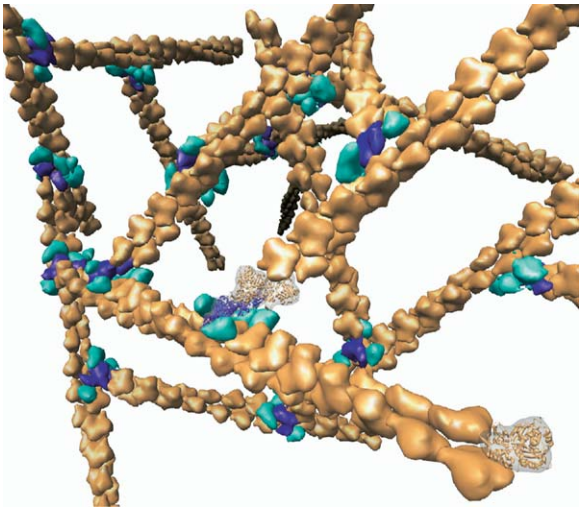

Figure 6. Branched Actin Fibers

Network of actin filaments (orange) with branches nucleated by the ARP2/3 complex (light and dark blue). PDB models for an actin monomer (1atn), a short actin filament (1alm), and the ARP2/3 branching complex (1k8k) were used to build this 2.5 million-atom model. Growing intracellular networks propel motile cells.

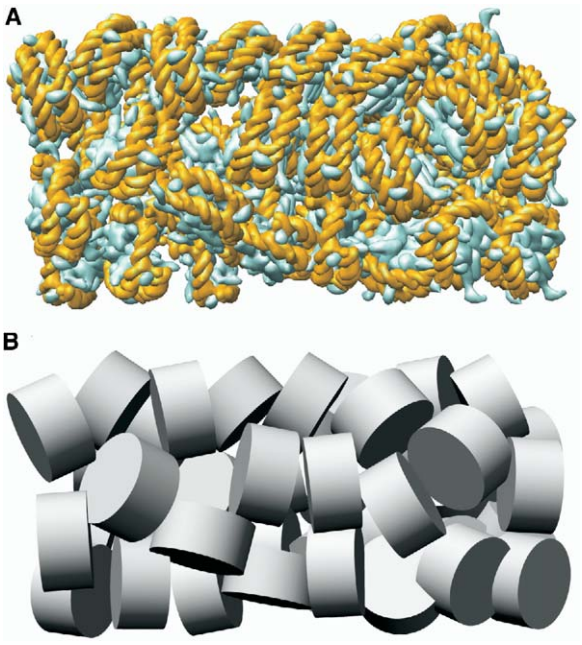

Figure 7. Chromatin Fiber Model

(A) 40 nucleosomes (PDB model 1eqz) arranged as a 30 nm diameter fiber. This is a conceptual model of a chromatin fiber, a DNA-packaging substructure in eukaryotic chromosomes. Nucleosomes are placed on a helix with partially randomized orientations. Each nucleosome consists of eight histone proteins wrapped by two turns of double-stranded DNA. All proteins and DNA strands are depicted with separate low-resolution surfaces.
(B) Same fiber model in which each nucleosome is represented as a cylindrical tablet provides a clearer view of nucleosome placements.

bounding box for the coordinate array of that subset is used. The corners of the bounding box are transformed by using the rotation and translation for the subset, and a bounding box for those corners is obtained. This bounding box bounds the subset but is typically bigger than the minimal box that would be obtained by taking the minimum and maximum x, y and z values of the rotated and translated coordinates. But we try it initially to avoid computing the transformed coordinates. If a pair of subsets is found to have these bounding boxes separated by less than the distance threshold, then new minimal bounding boxes are calculated by using the transformed coordinates for such subsets, and the minimum separation of the new subset bounding boxes is checked. These minimal bounding boxes are remembered for use in measuring the separation of other subset pairs.

Each pair of subsets in which the minimal bounding boxes are separated by less than the distance threshold is processed with a divide-and-conquer method. The minimal bounding box of each subset is divided in half along its longest axis (x, y, or z), and index lists for points lying in each half are determined. Then, the four possible pairings of these half subsets are examined for close points by recursively checking if the minimal bounding boxes are close enough and, if so, subdividing further. If one or both subsets contains fewer than 20 points, then a distance-squared calculation is made for all pairs of points, and those within the distance threshold are recorded.

In the above-mentioned description, the separation between bounding boxes means the minimum distance between the surface of one box and the surface of the other box. This is not simply the minimum separation along the x, y, and z axes, since the closest approach may be along a diagonal. It is simple to calculate by summing the squares of the minimum x, y, and z distances between boxes.

An unusual feature of this algorithm is that the two input sets of points are specified as collections of subsets. The algorithm consists of one stage of comparing input subsets and a subsequent stage using bisection of subsets. It would be simpler to implement if only the bisection method were needed. This would be a natural approach if the input were simply two sets of coordinates. Our approach of taking subsets as input allows for the handling of larger multimeric systems and is simple to use by treating each molecule as a separate subset.

This algorithm performs best relative to comparing all pairs of points when the distance threshold is small. In this case, most points will never be used in distance calculations because they are found to be too far from the other set of points using bounding box checks. Another application of this algorithm uses a large distance threshold relative to the size of molecular subunits. As described below, it is sometimes useful to show only molecules in the neighborhood of a selected component of a large assembly. To improve the speed of our close points algorithm in this scenario of large distance threshold, a simple optimization can be added. When two subset bounding boxes are found to be separated by less than the distance threshold, another check is

made to see if the maximum distance between the surfaces of the two boxes is less than the distance threshold. If this is the case, then all points of the two subsets are contact points and can be recorded as such without doing further subdivision or computing distances for any pairs. Faster algorithms can be devised for the large distance threshold regime. The algorithm described here is adequately fast (finishing in seconds) for finding nearby molecular chains in practical cases with large virus capsid structures.

In our implementation, contacts are found between the currently selected atoms and all unselected atoms. The distance range can be specified or a default of 5 Å is used. Two buttons in the user interface, "Near" and "Contacts," invoke the same contact calculation but differ in which atoms end up selected. The "Near" button extends the initially selected set of atoms to include the nearby atoms. The "Contacts" button replaces the initial selection with just the atoms involved in contacts. The first mode expands the initially selected set of atoms, while the second gives just the atoms on the contact interface. Selected atoms in Chimera are shown highlighted with a green outline. They can be used in subsequent Chimera actions, such as finding hydrogen bonds, or written to a file if desired.

Selecting Subassemblies

Basic visualization steps such as hiding molecular chains outside a region of interest, recoloring pieces of an assembly, or changing display styles require methods for selecting chains. This is often done in molecular graphics software by clicking on the desired pieces with the mouse or by typing a name (e.g., residue 145). For large multimeric assemblies, additional methods are useful. In Figure 1, the outer layer of the virus capsid has been hidden on the bottom half of the particle. Individually selecting the several hundred protein monomers to be hidden with a mouse is time consuming. Names for the 60 icosahedral symmetries are not easily used to choose the desired monomers. An efficient method would be to select the monomers by defining a box by using the mouse. This has the undesired effect of also selecting the inner layer of capsid proteins. Two selection methods that extend an initial selection can help. The first method selects all structurally identical copies of the currently selected items. A small part of the outer capsid layer can be selected and then extended to select the whole outer layer. Then, everything except for the selected outer layer can be undisplayed and a box can be specified with the mouse to select just the half of the outer layer we wish to hide. A small drawback here is that it is necessary to initially select 13 different monomers on the bluetongue virus capsid. Although the same protein studs the surface of the virus, there are 13 copies of the protein in the icosahedral asymmetric unit. Because these monomers have slightly different contacts with neighbors, their conformations differ. One solution is to permit extending selections to all equivalent protein sequences. This would make it possible to select just one surface monomer and extend the selection to all proteins of the outer layer. We have not implemented extending selections by using equivalent sequences, but we can make the outer capsid selection in a few steps by using the more general technique described next.

The second selection method can handle the same example from Figure 1 by using an approach that is not limited to multimeric assemblies. An assembly can be described as a hierarchy, specifically a computer data structure called a tree. The top level of the tree is the entire assembly. For the bluetongue virus capsid, the next level down would consist of the inner and outer protein layers. The outer protein layer would have child nodes for the trimers composing it. The leaves of the tree would be the individual molecules. With such a hierarchy, an initial selection can be extended to include all components at the next higher level. An individual outer layer capsid protein can initially be selected with the mouse. This selection can be promoted to the containing trimer, and can be promoted once again to select the entire outer capsid layer. In our Chimera implementation, the hierarchy of a molecular assembly is described with a script in the Python programming language. The script lists the PDB chains and transformation matrices for the lowest level of the tree, and then the groupings for each higher level in terms of the level below it. The script for the bluetongue virus example is about 40 lines long. The user interface for promoting a selection to the next higher level is a single button. Promotion is done by finding nodes of the tree with all chains selected, then extending the selection to all chains in the parents of these nodes. In addition to copy and promotion types of selection, many common selection methods such as mouse operations, inverting selections, extending a selection to nearby molecular chains, intersecting or joining two selections, and naming useful selections for future use are important to allow easy manipulation of assembly displays.

Applications

The techniques described above have been included in UCSF Chimera (Pettersen et al., 2004), a molecular visualization and analysis package containing an extensive set of tools for studying macromolecules. Our additions provide low-resolution overview depictions of large assemblies, and the ability to easily zoom in on small functional pieces at atomic resolution and back out to the large scale structure. This enables two types of use, presentation and analysis. The low-resolution depictions are useful in publications and talks to convey high-level structural information. The facility for quickly navigating between a high-level view and multiple small functional sites at atomic resolution is useful for analyzing how a large complex functions. We now give examples of these presentation and analysis capabilities applied to virus capsids and actin networks.

Low-resolution molecular surfaces provide a clear picture of virus capsid architectures (Figure 4). Renderings were created by using UCSF Chimera for the Virus Particle Explorer web site (mmtsb.scripps.edu/viper) to depict the organization of proteins in the capsid. They provide a better picture of the arrangement of proteins than alternative methods. Displaying a single depth-cued virus particle surface has the drawback that boundaries between proteins are indistinct. A ribbon

representation of capsid proteins provides a poor sense of depth because lighting cues are ineffective when so much detail is present.

The analysis of virus capsid pores (Figure 5) illustrates the utility of easy navigation between functional sites of a large assembly. Bluetongue virus replicates its 10 segment single-stranded RNA genome entirely within the closed capsid and in the process takes in nucleoside triphosphates (NTP) and expels copied single-stranded RNA through capsid pores. Crystal soaking experiments have helped identify the location of these pores (Diprose et al., 2001). The 5-fold symmetry axes are the exit location of RNA (Figures 5C and 5D), while NTP enters a separate pore (Figure 5E). The large pore in the inner capsid layer at the 3-fold symmetry axis is sealed off by trimers of the outer capsid layer (Figure 5B). While these different sites in the virus capsid could be analyzed with programs designed only for small complexes by constructing files containing the nearby molecular components, it is substantially simpler to use a low-resolution model of the whole capsid to navigate to the regions of interest.

Another example is provided by networks of branched actin filaments, which propel motile cells. A dendritic network in the cytoplasm is the functional form, where growth at the leading edge and disassembly at the rear propels the cell (Svitkina and Borisy, 1999). The network shown in Figure 6 contains about 2.5 million atoms (no hydrogens) and 920 molecules. For atomic analysis, only a single branch, composed of a short base filament, an ARP2/3 branch nucleation complex, and a short branch filament, is relevant; this complex is composed of about 20 molecules. Presentation images frequently show assemblies that are larger than are needed for atomic resolution analysis as an aid to explaining high-level function.

The actin filament branch model was built by using three PDB structures: an actin monomer (1atn), a short 5 molecule actin filament (1alm), and the 7 protein ARP2/3 branch nucleation complex (1k8k). Sequence-based structure alignment, manual docking, and large assembly visualization were combined to determine the positions of the component molecules. The rotation and translation for adding the next monomer to a growing filament were determined by aligning the actin monomer to two successive monomers in the short filament structure. The ARP2/3 branch nucleation complex contains two actin-related proteins (ARP2 and ARP3) that form the base of the branch filament. Aligning a model filament with each ARP protein produces branches that extend in directions differing by about 20°. A conformational change in the ARP2/3 complex is believed necessary to align ARP2 and ARP3 proteins to have the correct geometry for two successive actin monomers in a filament. This conformational change was modeled as a rigid motion of three proteins of ARP2/3 by using the structure alignments to filaments. A last step in the model building was to manually dock the "closed" form of ARP2/3 to a base filament. The interplay of large assembly visualization and standard small system analysis capabilities within UCSF Chimera allowed the single branch model to be constructed in a couple of hours. That analysis produced a set of positioning matrices that were added (using a text editor) to the actin monomer PDB file as biological unit matrices. The dendritic network model was produced the same day with a Python script to grow filaments and branch them at random locations by using the matrices determined from the single branch model.

Conclusions

The software components we developed add basic functionality for presentation and analysis of large molecular assemblies. Many improvements are possible. Our implementation provides low-resolution surfaces of single molecules. As seen in the chromatin model of Figure 7A, in which each nucleosome contains eight histone proteins, this can show too much detail, obscuring the arrangement of the nucleosomes. Being able to represent a group of proteins such as the eight histone molecules as a single low-resolution surface would be helpful. The chromatin illustration in Figure 7B uses cylindrical tablets to represent nucleosomes, providing a clear depiction of their arrangement. Use of regular solids (cylinders, spheres, cones, slabs, polyhedra) to represent molecules or subassemblies would be a generally useful capability. Low-resolution surface depictions could also be used at a finer scale than single molecules in some cases. For example, the 101 helices of the 23S ribosomal RNA could be individually depicted, as could domains of multidomain proteins.

One of the formidable obstacles in analyzing large assemblies is that many of the database structures are poorly annotated. For example, about half of the 230 virus capsid structures at the PDB do not contain machine-readable matrices for generating the 60-fold icosahedral symmetry, or the matrices are incorrect. As another example, the helix domains for the 23S rRNA, standardized in the literature, are not part of the large ribosomal subunit database entries. These problems reflect the lack of software capable of using this data and verifying its correctness. Coupled improvements in database annotations and software systems for large assemblies will be needed to make these structures as amenable to analysis as single macromolecule structures are today.

Acknowledgments

This work builds on the extensible UCSF Chimera molecular modeling system developed by Greg Couch, Tom Goddard, Dan Greenblatt, Conrad Huang, Elaine Meng, Eric Pettersen, and Thomas Ferrin. It is funded by the National Center for Research Resources grant P41 RR-01081.

Received: November 1, 2004
Revised: January 19, 2005
Accepted: January 25, 2005
Published: March 8, 2005

References

Bajaj, C., Djeu, P., Thane, A.G., and Siddavanahalli, V. (2004). Interactive visual exploration of large flexible multi-component molecular complexes. In Proceedings of IEEE Visualization, Oct. 10–15 (Austin, Texas), pp. 243–250.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N.,

Weissig, H., Shindyalov, I.N., and Bourne, P.E. (2000). The Protein Data Bank. Nucleic Acids Res. *28*, 235–242.

DeLano, W.L. (2002). The PyMOL Molecular Graphics System (San Carlos, CA: DeLano Scientific).

Diprose, J.M., Burroughs, J.N., Sutton, G.C., Goldsmith, A., Gouet, P., Malby, R., Overton, I., Zientara, S., Mertens, P.P., Stuart, D.I., and Grimes, J.M. (2001). Translocation portals for the substrates and products of a viral transcription complex: the bluetongue virus core. EMBO J. *20*, 7229–7239.

Diprose, J.M., Grimes, J.M., Sutton, G.C., Burroughs, J.N., Meyer, A., Maan, S., Mertens, P.P., and Stuart, D.I. (2002). The core of bluetongue virus binds double-stranded RNA. J. Virol. *76*, 9533–9536.

Duncan, B.S., and Olson, A.J. (1995). Approximation and visualization of large-scale motion of protein surfaces. J. Mol. Graph. *13*, 250–257.

Humphrey, W., Dalke, A., and Schulten, K. (1996). VMD—Visual Molecular Dynamics. J. Mol. Graphics *14*, 33–38.

Hussain, A.S., Kumar, Ch.K., Rajesh, C.K., Sheik, S.S., and Sekar, K. (2003). SEM (Symmetry Equivalent Molecules): a web-based GUI to generate and visualize the macromolecules. Nucleic Acids Res. *31*, 3356–3358.

Klein, D.J., Moore, P.B., and Steitz, T.A. (2004). The roles of ribosomal proteins in the structure assembly, and evolution of the large ribosomal subunit. J. Mol. Biol. *340*, 141–177.

Lorensen, W.E., and Cline, H.E. (1987). Marching cubes: a high resolution 3d surface construction algorithm. Comput. Graph. (ACM) *21*, 163–169.

Macke, T.J., Duncan, B.S., Goodsell, D.S., and Olson, A.J. (1998). Interactive modeling of supramolecular assemblies. J. Mol. Graph. Model. *16*, 115–120, 162–163.

Pettersen, E.F., Goddard, T.D., Huang, C.C., Couch, G.S., Greenblatt, D.M., Meng, E.C., and Ferrin, T.E. (2004). UCSF Chimera—a visualization system for exploratory research and analysis. J. Comp. Chem. *25*, 1605–1612.

Sanner, M.F., Olson, A.J., and Spehner, J.C. (1996). Reduced surface: an efficient way to compute molecular surfaces. Biopolymers *38*, 305–320.

Svitkina, T.M., and Borisy, G.G. (1999). Arp2/3 complex and actin depolymerizing factor/cofilin in dendritic organization and treadmilling of actin filament array in lamellipodia. J. Cell Biol. *145*, 1009–1026.