VisiCoor: A simple program for visualization of proteins

D.A. Kuznetsov* and H.A. Lim†

**Engelhardt Institute of Molecular Biology, USSR Academy of Sciences, Moscow, USSR; †Supercomputer Computations Research Institute, Florida State University, Tallahassee, FL, USA*

A well-drawn picture acts as an excellent metaphor for something real, and human vision provides instant, random access to any part of which the picture represents. It is in this sense that pictures can convey information more effectively than words alone. The power of the graphics workstations available today makes visual presentation of scientific results a reality. A molecular graphics program for investigating protein structures, as well as several sample plots that show the power of the program, are presented.

Keywords: molecular graphics, protein, secondary structure, supersecondary structure

INTRODUCTION

VisiCoor (visible coordinates) is a program for the visualization of protein structures. It has the capability to emphasize different regions of protein structures using a flexible logic composed of a large set of basic rules; it supports high-quality raster graphics; and when connected to a user-friendly interface, the program enables a user to build images in which all unnecessary details are suppressed so one can concentrate on substructures of interest. The latter relieves the user from the burden of unnecessary information while allowing one to capitalize on the power of human recognition of images. The program is small, easy to use, and nonproprietary, allowing the user more flexibility than if he or she is using a commercial package. It is useful in situations when the power of huge packages[1] is unnecessary, such as when an overview of a protein structure that requires simple analysis is desired.

GENERAL DESCRIPTION

The program is built around the assignment of a Boolean flag (TRUE) to atoms that satisfy a certain set of criteria, i.e.,

$$M(i;\alpha, \ldots ,\delta) = \begin{cases} .\text{TRUE.} & \text{if criteria satisfied for atom } i \\ .\text{FALSE.} & \text{otherwise} \end{cases}$$

The array $M(i;\alpha, \ldots ,\delta)$ is a mask, applied to the whole set of the atoms, which selects a subset of the atoms that satisfy the criteria $\alpha, \ldots ,\delta$. This assignment functionally enables the user to mark only pertinent atoms ($M(i;\alpha, \ldots , \delta) = .\text{TRUE.}$) as *active* and to manipulate these logically linked atoms as a single object. This subset of atoms is called a class. The definition of a subset of atoms, i.e., class building (also see below), is based on structural, physicochemical, and geometrical properties. After a class is defined, the user can *dress* these atoms by predefined graphics primitives and paint the atoms with different hues, with undo capability. The latter enables the user to build the molecular image from fragments representing different entities in a protein structure, e.g., α-helixes, hydrophobic cores, and active sites, etc.

GRAPHICS

The program uses the conventional atomic representations: stick (emulating a covalent bond), ball (emulating the van der Waals space of an atom), ball-and-stick (functionally the same as stick) and dots–shell (semitransparent ball). The first three representations are constructed from facets in the Tektronix version or "polished" facets in the Iris version that have been colored, lighted by three sources of light, Gouraud shaded, z-buffered, depth-cued, while the fourth representation consists of color points. Although the program exists in two versions (Iris and Tektronix versions, also see below), the same algorithms are used for the construction of these atomic representations in both versions.

A rather unconventional shape is used for the atomic representation for *stick*. The conventional shape of a long cylinder has been replaced by the shape of a long parallelepiped (*matchstick*). The matchstick consists of two joined halves colored according to the chemical types of the contiguous atoms. The calculation of the vertex coordinates of a matchstick is not obvious because the angle of rotation of a matchstick about an atom–atom axis is ambiguous and cannot be determined uniquely solely from the pair of atomic coordinate vectors. This problem is solved by an original

Color Plates for this article are on pages 21–22.

Address reprint requests to Dr. Lim at the Supercomputer Computations Research Institute, Florida State University, B-186, Tallahassee, FL 32306–4052, USA.

Received 10 December 1990; accepted 9 July 1991

algorithm that has been implemented in the program. The algorithm constructs a matchstick such that the two short edges of each of its sides lie in the x–y plane.

The matchstick constructed in this manner and used as an atomic representation has several advantages over a stick constructed as a long cylinder:

(1) Oriented sticks (the relative orientations of the sticks are maintained during the rotation of an image) reduce the possibility of confusion (Color Plate 1) and help the viewer to perceive the image as more structural. It helps, for instance, in estimating more accurately the direction of a bond or of dihedral angles between atoms.
(2) Two factors help to enhance the computational performance in the process of constructing a stick. First, the use of flat verges (all four normals to the vertices of a verge are the same) turns off the Gouraud interpolation of colors within a verge (such interpolation cannot be omitted in the case of a cylinder). Second, the use of a reduced number of long verges obviously decreases the computational time. These factors make possible an interactive access by the user to the dressed-by-sticks, colored-, lighted-, z-buffered image of a protein, which may be composed of up to a thousand atoms.

A ball primitive is constructed using the somewhat obvious approximation by two-polar polygons (like a globe). To enhance performance, only the front halves of the spheres are used. A ball-and-stick primitive consists of the described stick in combination with a scaled down ball. A dots–shell consists of points lying on atomic van der Waals spheres. Points that are obscured by other dots–shell spheres (i.e., in the background) are automatically removed by the program. The stick, ball, and ball-and-stick primitives are interactively *rotatable*, e.g., an image built from them can be rotated by clicking the mouse button. However, real-time operation of *stick wear* is available only for images consisting of up to a thousand atoms, and in the case of *ball wear* only for images consisting of up to fifty atoms. The dots–shell (if applied) disappears during rotation and is recalculated automatically in the new view angle.

The atoms might be dressed by any combinations of the above described representations and these primitives may appear on the screen simultaneously. The menu item *Dress* is the function option for this operation. The dressing operations may be undone either by deleting the graphics primitives in the reverse order of dressing, or by deleting the graphics primitives of a whole subset of atoms. These operations might be carried out using the menu item *Erase*.

By default all atomic representations are colored following the CPK-palette convention: H, white; C, grey; N, blue; O, red; S, green; and P, yellow. However, the user can change this default color scheme in selected subsets of atoms (see below). The colors violet, orange, and yellow are available for this operation, which is achieved by using the menu item *Paint*.

Another type of paint, *digitized color*, may be applied to the displayed protein structure. Such paint enables one to investigate the arrangement of a one-dimensional function associated with the primary structure of a protein (like hydrophobicity, and frequency of the occurrence of amino acids, etc.) on the protein's three-dimensional globule. The algorithm of color coding works in four successive steps:

(1) It reads in the table of function values assigned to the specified amino acids from the provided external file,
(2) it maps the function's magnitude table into the numerical range 1, . . . ,5,
(3) it takes the ith, (i = 1, . . . ,5) subrange and while scanning along the sequence, assigns the preset color of the ith subrange to an amino acid residue whose function value is within the subrange,
(4) it applies the formed color table to the protein structure image.

The amino acids thus become color coded based on their values of the function. This task may be carried out using the subitem in the menu item *Paint*.

CLASS BUILDING

Class building is the creation of a list of atoms that are pertinent for visualization by the user. The program is designed as nonproprietary at this point and this enables the user to specify a list that satisfies his or her criteria and that can be read in as an externally provided file. However, the program has 12 preset criteria that may be used for class building. The creation of a class might be accomplished either by building it from scratch or by modifying an existing list of atoms that may have been read in earlier from the external file.

The preset criteria are divided into three categories (structure, residues features, and geometry), and are listed in the submenus located under the menu item *Set class*. Having opened one of the submenus, the user can start accumulating criteria by selecting them from the preset list. To finish the accumulation the user must open the menu item *Exit* and specify how to use the list of accumulated criteria.

Each entry from the list of preset criteria may be used as itself or as its complement (e.g., α or NOT α). The list of accumulated criteria may be dynamically modified or updated. After the user has finished building the criteria list, the program sets up a list B of atoms that satisfy *all* the accumulated criteria and puts B into a temporary buffer. There are three ways of modifying the current class (defined here as list A) using the list B: overwriting A, adding to A and deleting from A (each of these operations is encoded by the item in the submenu *Exit*):

$$A \rightarrow \begin{cases} B & \text{(overwriting)} \\ A \cup B & \text{(addition)} \\ A \cap \bar{B} & \text{(deletion)} \end{cases}$$

Before modifying the list A (current class), the program always copies it into an archive buffer, so the user may recover previous lists. The current class may also be inverted, i.e., $A \rightarrow \bar{A}$. As described, it is possible to get the list A from an external file. The obvious possibility, to quit without modifying list A, is provided as well. Each of the four functions described above may be invoked by using the provided submenu items; the list B (if it exists) remains intact in these cases. Below is the list of 12 preset criteria:

In the *Structure* submenu the user can choose residues or atoms that:

(1) Belong to the main chain
(2) Belong to α-helices*
(3) Belong to β-sheets*
(4) Belong to turns*
(5) Have the selected range of torsion angles ϕ–ψ.†

Three sample plots highlighting the supersecondary structures in the proteins flavodoxin (4FXN), Bence-Jones protein (2RHE) and plastocyanin 1PCY (Brookhaven Protein Data Bank (PDB) file names are given in the parentheses) are shown in Color Plates 2–4. Only the main chains of the proteins were read in.

In the *Residue* submenu the user can choose residues:

(6) Which are Hydrophobic
(7) Which are Aromatic
(8) Which are Charged
(9) By name‡
(10) By number (from/to).‡

A sample plot revealing the hydrophobic core surrounded by the hydrophilic main chain and the polar side chains of the protein plastocyanin (1PCY) is shown in Color Plate 5.

In the *Distance* submenu the user can choose atoms that are:

(11) Near a particular atom‡
(12) Alone (atoms which have no bonds with any other atoms).

An example that exposes both the binding and catalytic sites of the protein ribonuclease 1RN3 is given in Color Plate 6.

DATA FILE USAGE

The program assumes that the input files match the PDB format.[2] Certain fields of the PDB format (HELIX, SHEET, TURN, and HETATM) are essential for logic features of the program. In the preface menu the following options are available:

(1) Read all atoms, the main chain, or C_α atoms.
(2) Read or skip heteroatoms.
(3) Set up the fragments to be read on the basis of residues numbers (up to 20 fragments).
(4) Set new covalent radii of C, H, N, O, P, and S atoms.

THE USER ENVIRONMENT

The user interface is written as a set of hierarchical pop-up menu frames. The nine menu items supported are: *Help, Rotate, Dress, Set Class, Paint, Calc, List Class, Erase,* and *Setup*. Below is a short description of each item:

(1) *Help* enables the user to obtain on-line help information (about 400 lines).
(2) *Rotate* enables the user to rotate the protein image around X–Z or Y–Z axes with variable increments from 0.1 to 30°. The *Rotate* item itself exists only in the Tektronix version, where the user might manage the rotation process by VT220 arrow keys while a pop-up pilot-display is showing the speed and direction of the rotation. In the Iris version, the user might perform a rotation using the mouse. The program defines both the directions and speed of rotation by the displacement of the mouse cursor from the center of the picture when the user is clicking the mouse button 1. Although the user might activate rotation asynchronously and it is therefore independent of the menu status, the program detects this and guides the rotation by the pop-up pilot display.
(3) *Dress* is described in the GRAPHICS section.
(4) *Set Class* is described in the CLASS BUILDING section.
(5) *Paint* is described in the GRAPHICS section.
(6) *Calc* enables the user to evaluate distances between two specified atoms or between a specified atom and atoms that are in the current class, with hard-copy capability.
(7) *List Class* enables the user to examine the current class contents.
(8) *Setup* enables the user to adjust miscellaneous features (background colors and depth cueing, etc.) that may vary within versions of the program.
(9) *Exit* enables the user to exit from the program.

In both versions of the program the mouse is used as a means of interactively polling the image (the user may get a report about both the atom and residue numbers of a selected atom). All time-consuming operations are accompanied both by an animated picture (an indicator filling up a bar dial with the length of the dial representing the fraction of the operation completed) and by a descriptive message.

TECHNICAL DETAILS

Program VisiCoor is available in two versions. A Tektronix version, which runs on a graphic terminal Tektronix 4129 coupled to a VAX, and an Iris version, which runs on an IRIS 4D/240GTX workstation.

Each version has separate displays for graphics and menu control. In the Tektronix version these run on physically separate displays, and on the Iris version they share one display using the X-window manager. The image building procedures were written using either the Tektronix 4129 escape sequence graphic language (Tektronix version)[3] or the IRIS 4D/240GTX graphics library (Iris version).[4] The menus in the Tektronix version were written using the VMS screen management utility SMG$. For supporting menus in the Iris version, an interface between the VMS SMG$ utility and the UNIX Curses utility was written. The timing data presented in the section on GRAPHICS pertains to the Iris version. The photos used as Color Plates in this article were taken using both the Iris version (Color Plate 1) and the Tektronix version (Color Plates 2–6).

CONCLUSIONS

Two versions of the molecular graphics program VisiCoor, one for the VAX-Tektronix 4129 and one for the IRIS 4D/240GTX Silicon Graphics workstation, are presented. The program, which has full graphics capabilities, is used

*Classification is performed based on the descriptions made in the PDB file[2]

†After the user has decided on this choice, the Ramachandran map of the protein will appear on the graphics screen. Only those residues having ϕ–ψ values within a rectangle marked by the mouse cursor will be chosen

‡The user will be prompted to specify residue name (Item 9), range of residues (Item 10) or both the central atom and the sphere radius (Item 11)

for displaying protein structures. Sample plots drawn using the program's ability to manipulate the display of protein structures can produce very instructive graphics for visualizing the supersecondary structures, functional sites, and other features of proteins. The program may be particularly useful in those investigations of protein structures where the description cannot be easily provided by mathematical approaches, but where human intuition and knowledge might provide a better guide in their interpretation.

ACKNOWLEDGMENTS

The original program was written for the VAX-Tektronix 4129 at the Engelhardt Institute of Molecular Biology during 1988–89. The program was ported to the IRIS 4D/240GTX workstation at the Supercomputer Computations Research Institute (SCRI) of the Florida State University in 1990.

Kuznetsov would like to thank SCRI for providing a most hospitable environment for the duration of this work. He would also like to thank Professor Vladimir Tumanyan for unfailing attention to this work and Professor Alex Wlodawer for helping revise the manuscript. The authors would also like to thank Eric Pepke, Douglas Lee, and the members of the Biophysics Group at SCRI for scientific advice. Lim is partially funded by a TRDA grant under contract number TRDA 116 and by SCRI, which is partially funded by the US DOE under contract number DE-FC05-85ER250000.

REFERENCES

1 For example, Quanta™, Polygen Corp.; and CHARMm™, Polygen Corp.

2 *Brookhaven Protein Data Bank*. version of January 1990; Bernstein, F.C., Koetzle, T.F., Williams, G.J.B., Meyer Jr., E.F., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T., and Tasumi, M. The Protein Data Bank: A Computer-based Archival File for Macromolecular Structures. *J. Mol. Biol.* 1977, **112**, 535–542; Abola, E.E., Bernstein, F.C., Bryant, S.H., Koetzle, T.F., and Weng, J. Protein Data Bank. In: *Crystallographic Databases—Information Content, Software Systems, Scientific Applications* (F.H. Allen, G. Bergerhoff, and R. Sievers, eds.) Data Commission of the International Union of Crystallography, Cambridge, 1987, pp. 107–132

3 *TEK Command Reference Manual*. Part no. 070-5141-02, Product Group 16, 1987

4 Silicon Graphics Computer Systems. Document no. 007-1210-020, version 2.0, 1990