A very fast program for visualizing protein surfaces, channels and cavities

Richard Voorintholt, M.T. Kosters and G. Vegter

Department of Computer Science, University of Groningen, Groningen, The Netherlands

Gerrit Vriend and W.G.J. Hol

Laboratory of Chemical Physics, University of Groningen, Groningen, The Netherlands

A method for visualizing molecular surfaces is described that uses a grid to store the distance to the nearest atom. Using on-the-fly three-dimensional (3D) contouring of a molecular graphics program such as FRODO,[1] one can obtain a good impression of van der Waals surfaces and solvent-accessible surfaces. The main advantages of the method described here are its high speed and the fact that no recalculations need to be done to obtain the solvent-accessible surface visualized for a probe with another radius.

Keywords: solvent-accessible surface visualization

INTRODUCTION

Cavities and channels play an important role in the structure and functioning of proteins. Knowledge of the size and shape of cavities can, for instance, assist in designing closer-packed proteins. Information about surface depressions and channels running into the protein can also be useful when designing drugs or, more generally, ligands interacting with proteins in a highly specific manner.

To facilitate the analysis of cavities in a protein, as well as in protein surfaces, a program was written that calculates a map whose value is in each point a function of the distance to the nearest atom. This map is inspected interactively on a PS300 using the on-the-fly contouring facilities of the program FRODO.[1] This is done in the same way that a normal electron density map is treated. In this manner, the accessible surface can be visualized, and the size of the probe for which the atoms are accessible can be varied interactively. When cavities are being investigated, the program shows contours around cavities large enough to hold a probe with radius R, without having overlap with protein atoms. The radius of this probe can be varied simply by contouring at a different level.

PROGRAM DESCRIPTION

A grid with a spacing of 0.3 to 1.0 Å (the user can choose any value) is established, and the value of each gridpoint is determined in the following way: If the gridpoint falls within the van der Waals radius of any atom, it is assigned a value of 100. When a gridpoint is further than the van der Waals radius, R_v, away from the closest atom, but closer than the van der Waals radius plus the radius R_p of the maximal probe desired, a value between 0 and 100 is assigned. Formula (1) summarizes the values given to gridpoints:

$$\left.\begin{aligned} F_{sq}(x,y,z) &= 0 \qquad \text{for } d > R_p + R_v \\ F_{sq}(x,y,z) &= C \cdot \frac{(R_v + R_p)^2 - d^2}{(R_v + R_p)^2 - R_v^2} \\ &\qquad\qquad \text{for } R_v < d < R_p + R_v \\ F_{sq}(x,y,z) &= C \qquad \text{for } d < R_v \end{aligned}\right\} \quad (1)$$

where d is the distance of gridpoint (x, y, z) to the nearest atom; R_v is the van der Waals radius of the nearest atom; R_p is the maximal radius of the probe for which the solvent exclusion surface is calculated (C is the constant maximal grid value = 100 in our program). Whenever a gridpoint would receive multiple values, the maximum value is selected.

Because there will always be many more gridpoints than

Dr. Vriend's present address is EMBL, Postfach 10.2209, 6900 Heidelberg, FRG.

Address reprint requests to Dr. Voorintholt at the Laboratory of Chemical Physics, University of Groningen, Nijenborgh 16, 9747 AG Groningen, The Netherlands.

Received 24 May 1989; accepted 21 June 1989

atoms, we decided to let the major loop in the program go over the atoms rather than over the gridpoints. Additional CPU time is saved by using the squares of distances rather than the real distances themselves throughout the program.

The program output consists of a direct access file of "bricks" of 512 bytes, where the grid coordinate is stored as the position of the brick in the file, combined with the position of the byte in the brick, and the grid value is stored as the value of the byte. These files can be used directly by the program FRODO as map files (also called DSN6 files). Source code is available that can be modified to generate a more general formatted or unformatted map file.

NOTES ON THE IMPLEMENTATION

Since the above-mentioned algorithm spends virtually all its CPU time calculating the distances from atoms to gridpoints, we decided not to use these distances themselves, but rather the squares of these distances. The following notes on re-writing Pythagoras' law show why this is beneficial:

For every atom and the square around it, the gridpoints falling within it are filled starting at the gridpoint (X_g, Y_g, Z_g) nearest to the center (X_m, Y_m, Z_m) of the atom. The square of the distance d_1 between (X_g, Y_g, Z_g) and (X_m, Y_m, Z_m) is:

$$d_1^2 = (X_m - X_g)^2 + (Y_m - Y_g)^2 + (Z_m - Z_g)^2 \qquad (2)$$

If we now, for example, want the squared distance d_2^2 between (X_m, Y_m, Z_m) and $(X_g + n\Delta X, Y_g, Z_g)$, we will have to calculate:

$$d_2^2 = (X_m - X_g - n\Delta X)^2 + (Y_m - Y_g)^2 + (Z_m - Z_g)^2 \qquad (3)$$

in which ΔX is the gridpoint-to-gridpoint distance along the X-direction. It can easily been shown from equations (2) and (3) that:

$$d_2^2 = d_1^2 - 2.n.X_m.\Delta X + 2.n.X_g.\Delta X + (n\Delta X)^2 \qquad (4)$$

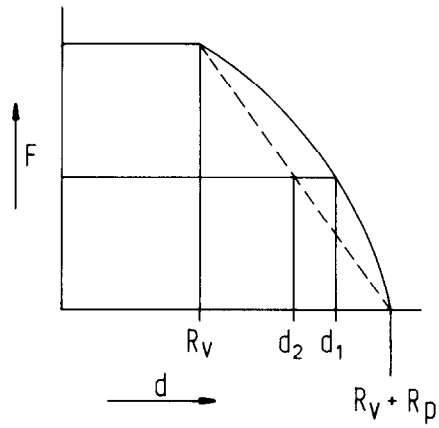

Figure 1. The solid line shows the quadratic density function from formula 1. The dashed line indicates the linear function (formula 2) needed for "perfect" linear interpolation results

If we now walk over the grid in steps along the X-direction, then all terms to be added to d_1^2 are incremental constants, depending only on the grid definition and the position of the atom.

This algorithm depends on the grid being orthogonal. The introduction of nonorthogonality in the grid definition would lead to a more complex algorithm and therefore more CPU time per run. However, for most applications an orthogonal grid is sufficient. A problem arises from the fact that we use a linear interpolation in the 3D-contouring program, whereas a quadratic interpolation is actually needed. Figure 1 shows the example with the largest error due to this linear interpolation.

To illustrate the severity of the error, the case that $R_v \approx R_p \approx \Delta X$ is worked out for this situation: the two points between which is interpolated lie exactly at gridpoints and are, respectively, R_v and $R_v + R_p$ away from the center of the nearest atom. If we now want to contour at level $F_{sq}(d_1)$, then instead of finding a point at distance d_1 away from the nearest atom, this distance becomes d_2 (see Figure 1). The misplacement of the point is $\in = d_1 - d_2$.

Since, under the above assumptions

$$F_{sq}(d_1) = C.\frac{4\Delta X^2 - d_1^2}{3\Delta X^2} \qquad (5)$$

the error $\in$ as function of d_1 becomes

$$\in(d_1) = d_1 - 2\Delta X + \Delta X.\frac{4\Delta X^2 - d_1^2}{3\Delta X^2} \qquad (6)$$

Solving $\frac{\partial \in(d_1)}{\partial d_1} = 0$ gives $d_1 = 1\,1/2\ \Delta X$ (i.e., half way R_v and $R_v + R_p$) which leads to a maximal error of $1/12\ \Delta X$. In practice, this error will always be much smaller, of course. Another pathological case is, for example, $R_v + R_p = n\Delta X + \delta$ and $d_{true} = n\Delta X + \in (\in < \delta)$. In this case, a maximal error of $\sim 1/2\ \Delta X$ can arise.

RESULTS

Running the program on a MicroVAX II with 2 megabytes of real memory and 10 megabytes of virtual memory available, typical CPU times of 2 to 10 minutes are observed for medium-sized proteins (around 2 500 atoms) and a normal probe radius (1.5–2.5 Å) on an intermediate fine grid (0.5–1.0 Å spacing). This program is at least one order of magnitude faster than other programs reported earlier (e.g., Ref. 2). Another great advantage over other programs (like the bit lattice method described in Ref. 3) is that no recalculation of the surface is needed if another probe size is required, provided the maximum probe size is not exceeded. In practice, this also saves considerable CPU time.

Even though this surface visualization method is not identical to the well-known Connolly method,[4] it is much faster and is useful particularly in instances where different probe radii have to be tested rapidly.

If memory size limitations exist, the program can easily be adapted to using much less memory. To do so, one needs to sort the atoms on increasing Z-coordinate and write out

sections of density as soon as they are ready, instead of all at once at the end of the grid value calculations.

This program is the result of a collaboration between the departments of Computer Science and Chemical Physics. The source code for the program is available to nonprofit organizations. Please use bitnet mail to VRIEND@EMBL for requests.

This program is also present as a subroutine in the Groningen program for molecular modeling and drug design WHAT IF.[5]

REFERENCES

1 Jones, T.A. *J. Appl. Cryst.* 1978, **11,** 272–288
2 Barford, D. and Johnson, L.N. *Inf. Quart. Prot. Cryst.* 1987, **20,** 41–43
3 Pearl, L.H. and Honegger, A. *J. Mol. Graphics* 1983, **1,** 9–12
4 Connolly, M.L. Ph.D. thesis, University of California, 1981
5 Vriend, G. *J. Mol. Graphics,* in press