Implicit surface visualization of reconstructed biological molecules☆

Edgar Garduñoa,*, Gabor T. Hermanb

aNational Center for Microscopy and Imaging Research at the University of California, San Diego, CA, USA
bDepartment of Computer Science, The Graduate Center, City University of New York, New York, USA

Dedicated to the memory of Alberto Del Lungo, friend and collaborator. We miss his smile

Abstract

An implicit surface of a density function is the set of points at which the value of the function is equal to a fixed threshold. An object that is defined as the collection of points at which the density function value is above the threshold can be visualized by displaying the implicit surface. Some methods for the reconstruction of biological macromolecules from their electron microscopic projections produce density functions that are specified by a linear combination of smoothly-varying radially-symmetric basis functions of finite support, also known as blobs. When density functions are determined by such a blob representation, the implicit surfaces are smoothly varying and the normal at any point on such a surface can be analytically calculated. This property can be utilized to produce high-quality visualizations by raycasting. While raycasting tends to be computationally expensive, we present a methodology that uses techniques of computer graphics and image processing to significantly reduce the cost of visualization.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Implicit surfaces; Body-centered cubic grid; Spherically symmetric basis functions; Generalized Kaiser–Bessel functions; Blobs; Algebraic reconstruction techniques; Raycasting

☆ This work has been supported by NIH Grant HL70472.
* Corresponding author.
E-mail addresses: edgargar@ieee.org (E. Garduño), gaborherman@yahoo.com (G.T. Herman).

0304-3975/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.08.027
1. Introduction

An essential tool toward the understanding of how living systems function is the visualization of the molecules from which they are constructed. Such molecules cannot be seen directly. A standard device for collecting information about them is the electron microscope. This instrument can be used to obtain projected images of a particular kind of molecule from a number of directions; such images are similar in nature to X-ray images of the human body (but are much noisier and, relative to the size of the molecules, are much less detailed). Multiple images of a molecule are then processed by a reconstruction algorithm, the output of which is a function of three variables that approximates the electron-density distribution in space associated with the molecule. (The molecule is typically distinguishable from its background by having a higher electron density.) Such a function is typically represented as a linear combination of some fixed basis functions; in order to obtain the necessary details, the number of such basis functions tends to be well over a million.

A successful choice for the basis functions are the so-called blobs (to be defined below); these are spatially limited and continuously differentiable functions for which closed-form formulas exist to calculate their gradients at any point in space. A consequence of this is that, once the coefficients of the blob decomposition have been obtained by the reconstruction algorithm, the gradient of the reconstructed electron-density distribution can be analytically calculated at any point of the implicit surface that separates the values associated with the molecule from the values associated with its background. This provides us with the ability to visualize the shape of the molecule without having to make approximations in addition to those that were unavoidably introduced by the electron microscopic reconstruction process. (Compare this, for example, with a polygonal approximation of the implicit surface prior to visualization: the display of such a polygonal approximation contains additional inaccuracies both in the location of the implicit surface and in the normal to it.)

This paper is devoted to the discussion of how computer graphics techniques can be applied to achieve accurate visualization of biological molecules based on reconstructions that are presented to us as linear combinations of well over a million blobs. This application is essentially different from the constructive solid geometry applications that are the main motivations for many implicit surface visualization papers in the literature (e.g., [16]): we are not trying to design a surface, rather we are trying to accurately visualize a particular surface that exists in nature, based on a representation that is imposed on us by the electron microscopic reconstruction process.

What we present below in detail will be one approach to visualizing density functions by two-dimensional (2D) images (generated on a computer screen) that use depth cues to deliver the three-dimensional (3D) information contained in the density functions. Volume and surface rendering (also referred to as direct and indirect volume rendering, respectively) are commonly used in biomedical imaging to create such representations of density functions.

Volume rendering is a technique that assumes that the density function to be imaged is made up of one or several translucent objects. The resulting 2D image is a projection of the translucent density function onto the screen. To create this image, the reconstructed density function is discretized into small volume elements (voxels, which can be simple abutting cubes or overlapping ellipsoids [51]), and then the opacities and colors of every object in the density function are computed locally for every voxel.
Surface rendering assumes that the objects embodied in the density function can be represented by the surfaces enclosing them. The estimation of the surfaces defining the objects generally requires some preprocessing of the density function. One of the common approaches to surface rendering is the polygon-projection method that explicitly approximates an object’s surface by a collection of polygons or other patches [2,5,14,19,20,36,48].

The advantages of one of these two approaches over the other are open to question; both have their strengths and weaknesses [44,46]. A preliminary process of object delineation and identification is assumed in surface rendering, this task is typically challenging and in many occasions does not produce the desired results. Moreover, there are natural objects whose surface representation does not accurately reflect its real structure (e.g., a cloud of particles). Volume rendering, on the other hand, leaves the task of structure recognition to the observer by projecting all the voxels onto the screen; in this case, the identification of structures may be difficult due to the presence of occluding objects. (However, some approaches allow automatic assignment of opacities to voxels so that there is no obstruction to “important” regions of the volume and also provide clues for better interpretation of the information contained within the visible regions [24,25]). In computational terms, identifying the surface of an object can be costly, but once the surface representation is available it is possible to rapidly generate images from any viewpoint (this is particularly true for surfaces approximated by polygons as there exists specialized hardware for polygon projection operations). As opposed to surface rendering, volume rendering is in general a computationally demanding task as it requires the projection of all the voxels in the data set every time the observer’s point of view changes and the computation of absorption, emission and color for every object in the density function can be challenging [43]. Techniques exist, nevertheless, for the combination of object rendering and surface rendering to produce a single image [17,26].

In this paper we use a modality of an algorithm known as raycasting for surface rendering which uses implicit surfaces. This approach is known to produce high quality images (a reason why we selected this modality), but also for its slow performance. (Other approaches using raycasting, but with polygons modeling the surfaces, have been shown to produce high quality images in interactive times [47].) We present a methodology that uses techniques of computer graphics and image processing to significantly reduce the cost of visualizing macromolecular complexes by raycasting. The methodology slightly increases the use of memory but, as we show later, the cost is minimal as compared to the memory use by polygonal methods.

In the next section we introduce some basic concepts such as implicit surfaces, basis functions, linear approximation by basis functions, grids, and projections. In Section 3 we present our particular choice for the basis functions. In Sections 4 and 5 we introduce a basic method to visualize implicit surfaces by raycasting with smooth basis functions and apply it to electron microscopy. Section 6 presents our approach to improving the raycasting method and some results. We summarize the paper in Section 7.

2. Background and related work

Several techniques have been used to model surfaces, e.g., fractals, hypertextures, particle systems, parametric surfaces. Here we adopt the use of implicit surfaces (also
called isosurfaces or isointensity surfaces). These are appropriate for objects with complex topologies and geometries, such as organic objects or man-made shapes (see [10] and its references), and therefore have been used to visualize objects of interest in many areas of science.

An *implicit surface* \( S \) of a density function \( \rho \) is defined as

\[
S = \{ \overline{x} \mid \rho(\overline{x}) = t \},
\]

where \( \overline{x} \) denotes a point in space (mathematically \( \overline{x} \in \mathbb{R}^3 \)). The assumption is that there is a threshold \( t \) such that the object of interest consists of exactly those points at which the value of \( \rho \) is greater than the threshold. If the total volume of the object of interest is known (as is the case in some applications, such as electron microscopy), then \( t \) is uniquely determined by the criterion that \( S \) should enclose exactly the known volume. Consequently, it is sufficient to display the surface \( S \), as defined by (1), of the object of interest for its visualization.

A standard way of specifying a density function \( \rho \) is by a linear combination

\[
\rho(\overline{x}) = \sum_{j=1}^{J} c_j b_j(\overline{x}),
\]

where \( \{b_j\} \) is a set of basis functions, each of which is weighted by a coefficient \( c_j \). While the basis functions are selected a priori, the coefficients vary with the density function \( \rho \).

In the computer graphics field Blinn introduced the so-called *blobby* model [9] that uses Gaussians for the basis function. Other basis functions have been suggested since; some examples include multiscale wavelets [39], piecewise quadratics [40], splines [45] and polynomials [49]. In practice, the basis functions have to drop to a negligible value beyond a moderate radius. The idea of using basis functions that are smooth (in the sense of being multiply differentiable everywhere) has been suggested by several authors, motivated by the belief that implicit surfaces of the resulting density functions would reflect more accurately the smoothness of natural objects. This idea has been also adopted in the field of reconstructing surfaces from point sets, typically generated by high resolution scanners, where (2) is used to smoothly approximate the surface represented by the data [3,11,19,30,50].

In the field of reconstruction from projections Lewitt [28,29] and Matej [35] proposed the generalized Kaiser–Bessel window functions, also referred to as *blobs*, as basis functions. These are functions with compact support, spherical symmetry, and a smooth transition from one to zero. They have proved to be efficacious for a number of reconstruction tasks [18,23,34] and, in particular, the authors of [31,33] obtained results suggesting that the combined use of the so-called algebraic reconstruction techniques (ART) and blobs produce better reconstructions than (or at least as good reconstructions as) those produced by other algorithms. (In both papers, the authors compare ART with other reconstruction algorithms, based on transform methods, using several figures of merit—FOMs—that permit the quantification of the quality of reconstructions of mathematical phantoms. Such FOMs can measure the mean error between a phantom and its reconstruction and, as is more relevant in practice, the success in recovering interesting features of the phantom. For each FOM, a level of statistical significance is assigned to the claim of superiority of one reconstruction method over another.)
Fig. 1. In the series expansion methods a density function $v$ is approximated by the linear combination of basis functions. Smooth basis functions, such as blobs, may reflect more accurately the smoothness of natural objects. The support of a blob is a sphere that we indicate transparently. To approximate a density function, each basis functions $b_j$ is weighted by a coefficient $c_j$. The coefficients $c_j$ are associated with points $\vec{p}_j$, which are the centers of the basis functions $b_j$; we represent these points by small solid spheres. Under this model, a line along which the integral $[Pv]_i$ is calculated will intersect only a few basis functions.

While the basis functions of (2) are preselected and fixed, the coefficients have to be determined by the reconstruction algorithm. For reconstruction purposes, we consider a projection to be a collection of line integrals. For a function $v$ over $\mathbb{R}^3$ we define the ray transform as

$$[Pv](\vec{o}, \vec{x}) = \int_{\mathbb{R}} v(\vec{x} + \tau \vec{o}) \, d\tau.$$ (3)

The operator $P$ provides all the integrated densities along all the lines $\ell$ defined by $\vec{x} + \tau \vec{o}$, where $\vec{o}$ is a unit vector and $\tau \in \mathbb{R}$ (see Fig. 1). It follows from (3) and (2) that, for any $(\vec{o}, \vec{x})$,

$$[Pv](\vec{o}, \vec{x}) = \sum_{j=1}^{J} c_j [Pb_j](\vec{o}, \vec{x}).$$ (4)

Suppose that our measurements are made for $I$ lines, characterized by $(\vec{o}_i, \vec{x}_i)$, for $1 \leq i \leq I$. Then we get

$$y_i \approx \sum_{j=1}^{J} \ell_{i,j} \, c_j,$$ (5)

where $y_i$ is the $i$th measurement and $\ell_{i,j} = [Pb_j](\vec{o}_i, \vec{x}_i)$, a value that can usually be determined analytically. The reason for the $\approx$ in (5) is that $y_i$ is not exactly $[Pv](\vec{o}_i, \vec{x}_i)$. 
due to noise in the measurements and the fact that the physical object can only be approximated by an expansion of the form given in (2). Thus (5) is approximately a system of linear equations $\mathbf{y} = \mathbf{L}\mathbf{c}$ that can be solved either by some sort of direct inversion method or by an iterative method. In practice, the system $\mathbf{y} = \mathbf{L}\mathbf{c}$ is often overdetermined (to compensate for the noise in the measurements, typically many more measurements are taken than there are unknowns $c_j$).

A particular iterative method to solve the system $\mathbf{y} = \mathbf{L}\mathbf{c}$ is a generalization of Kaczmarz’s method [21] for solving both over- and underdetermined linear systems of equations. For this generalization, we consider that the matrix $\mathbf{L}$ can be divided in $N$ blocks ($n$ denotes the number of a block, $1 \leq n \leq N$) with $M$ rows in each block, and so $\mathbf{I} = M \times N$. The $k$th iterative step of the method is

$$
\mathbf{c}(k+1) = \mathbf{c}(k) + \lambda(k) \sum_{i=(n-1)M+1}^{nM} \frac{y_i - \langle \mathbf{e}_i, \mathbf{c}(k) \rangle}{\sum_j \mathbf{e}_{i,j}^2} \mathbf{e}_i, \text{ for } n = [k \text{ (mod } N)] + 1,
$$

where $\mathbf{e}_i$ is a $J$-dimensional vector whose $j$th component is $\ell_{i,j}$ and $\lambda(k)$ is a relaxation parameter that determines how much the solution is updated in the $k$th iteration. In practice, the blocks are chosen to correspond to single projections; thus $\mathbf{e}_i$ has the same value for all $i$’s in a single block. ($N$ is the number of projections and $M$ is the number of sample points in a projection.) It turns out that, in our application, solving the linear system by (6) is faster than inverting the matrix $\mathbf{L}$ because a line intersects only a very few basis functions (see Fig. 1), resulting in a matrix that is sparse (most of its entries are zero valued) whereas a generalized inverse $\mathbf{L}^+$ [7] (typically $\mathbf{L}$ is not invertible) has mostly non-zero entries. The sparsity of $\mathbf{L}$ implies that (6) can be efficiently implemented [35].

When this approach is taken, the result is a representation of a biological molecule that uses blobs to specify density functions. To visualize the molecule we need to render an implicit surface (1), the threshold $t$ for which is determined by the total volume of the molecule (known to us from other sources). To avoid introducing approximations in addition to those that had to be made in the electron microscopic reconstruction process, we desire to visualize the implicit surface directly, without making additional approximations to it. This can be achieved by raycasting, similar to that suggested in [9], with adjustments suitable for data sets produced by reconstruction methods such as ART.

3. Blobs and grids

The general form of a single blob is [28]:

$$
b(m, x, a; r) = \begin{cases}
I_m \left( 2 \sqrt{1 - \left( \frac{a}{r} \right)^2} \right) \left( \sqrt{1 - \left( \frac{r}{a} \right)^2} \right)^m, & \text{if } 0 \leq r \leq a,
\end{cases}
$$

where $r$ is the radial distance from the blob center, $I_m$ denotes the modified Bessel function of order $m$, $a$ is the radius of the blob (the value of $b$ is zero for $r > a$), and $x$ is a parameter controlling the blob shape. The three parameters $m$ (a non-negative integer), $a$ and $x$ (non-negative real numbers) control the smoothness and shape of a blob and influence the results.
yielded by reconstruction and visualization algorithms; therefore, the appropriate selection
of them is highly important. Hereafter we set $m$ equal to 2, which makes the blobs to have
continuous first derivatives everywhere.

The individual blobs $b_j$ of (2) are shifted versions of the blob $b$ defined in (7). We refer to
the set of points $\{\overline{p}_j\}$ to which the centers of the blobs are shifted in such a representation
as a grid. As an example for (2) using blobs as basis functions we refer to Fig. 1, in which
a density function describing the donut-shaped object is to be approximated by a linear
combination of blobs. The grid is represented by the small spheres in Fig. 1.

The choice of the grid $\{\overline{p}_j\}$ is important. It is shown in [38,42] that the body-centered
cubic (bcc) grids provide the most “efficient” sampling of $\mathbb{R}^3$. The bcc grids are defined by

$$B_A = \{ \Delta \overline{k} | \overline{k} \in \mathbb{Z}^3 \text{ and } k_1 \equiv k_2 \equiv k_3 \mod 2 \}, \quad (8)$$

where $\overline{k}$ is the transpose of the 3-tuple $(k_1, k_2, k_3)$ (i.e., a three-dimensional vector), whose
components belong to the set of integers denoted by $\mathbb{Z}$ and $A$ is a positive real number
(the sampling distance). In order to visualize this grid, we can use a small portion of it
and take advantage of its periodic repetition; see Fig. 2. For reconstruction purposes, Matej
and Lewitt [35] demonstrated that whenever a linear combination of blobs is employed to
obtain a reconstruction, the bcc grids provide desirable sets of locations for the centers
of the blobs. In what follows we will use for $\{\overline{p}_j\}$ the set obtained by the intersection of some
finite convex region of space with a $B_A$ of (8).

Having decided that we use $m = 2$ and the bcc grid, there are three parameters to be
chosen: $A$, $a$, and $\alpha$. Clearly, to be able to approximate arbitrary distributions using (2),
the value of $A$ should be small. However, in a fixed volume of space, the number of grid
points (and consequently the cost of a reconstruction algorithm) is proportional to $1/A^2$ and
so practical considerations do not allow us to choose $A$ to be very small. The cost of
reconstruction (in our implementation using footprints [29,35]) is also proportional to $a^2$.
The computational cost does not depend on $\alpha$, and so this parameter may be chosen purely
based on the quality of the resulting reconstructions.

In [35] a method was proposed for the selection of the parameters $A$, $a$, and $\alpha$ with
the aim of ensuring that ART will produce “good” reconstructions. This method selects the
blob parameters by assuming that a linear combination of blobs with $c_j = 1$, for $1 \leq j \leq J$,
should approximate a constant valued function. For this case, the right hand side of (2) is a
convolution of the blob $b$ in (7) with a truncated version of the train of pulses $\text{III}_{B_A}$ (pulses
arranged on the bcc grid). The Fourier transform of such convolution is approximately
$$\mathcal{F}\{b \ast \text{III}_{B_A}\} = \mathcal{F}\{b\} \times \mathcal{F}\{\text{III}_{B_A}\}.$$ Since $\mathcal{F}\{\text{III}_{B_A}\}$ is also a train of pulses (on a so-called
face-centered cubic grid), for this to best approximate the Fourier transform of a constant-valued function (an impulse at the origin) it is useful to select $b$ in such a way that $\mathcal{F}\{b\}$ is zero-valued at the locations of the pulses in $\mathcal{F}\{\text{III}_{B_A}\}$ that have the smallest positive
distance from the origin. It follows from this discussion and from the analytic formula for
$\mathcal{F}\{b\}$, available from [29], that we should select

$$\alpha = \sqrt{2\pi^2 \left(\frac{a}{A}\right)^2 - 6.987932^2}. \quad (9)$$
In [15] it was reported that the parameters yielded by the methodology suggested in [35] produced in some cases nonconvex reconstructions from data obtained from convex density functions; causing a significant inaccuracy in the visualization of the resulting implicit surfaces, see Fig. 3. To correct this problem [15] proposes an additional criterion for the selection of the parameters $A$, $a$, and $z$: $a$ should be chosen as small as possible consistent with both satisfying (9) and achieving that if two blobs at nearest grid points in the grid $B_A$ (those separated by $\sqrt{3}A$) are given coefficients 1 with all other blobs given coefficients 0, then the implicit surface thresholded at $t = 0.5$ should enclose a convex set, see Fig. 4. The selection of the latter criterion is not arbitrary but is based on the principle that the grid spacing should limit the resolution: nearest grid points should not be resolvable from each other. Using such a second criterion determines a single pair $(a, z)$ among all those that satisfy (9). This is the methodology used in the rest of this paper.

4. A basic raycasting-blobs technique for visualization

A suitable method for visualizing the surface in (1) is raycasting. In general, raycasting is slower than the polygon-projection methods. However, an accurate visualization of an implicit surface requires a careful selection of polygons, something that is avoided by raycasting whose accuracy is automatically determined by the pixel locations on the computer screen. Furthermore, implicit surfaces are particularly well suited for ray-intersection processing, since the density function defining the implicit surface enables us to compute the intersection between a ray and the surface by standard numerical zero-finding methods.

Raycasting produces a projection of $S$ onto the screen by casting a finite number of rays toward $S$. In one of its forms, the rays are perpendicular to the plane representing the plane of view, typically the computer screen. In order to produce a foreshortening effect in the...
Fig. 3. Visualization of the implicit surfaces of a reconstructed sphere-shaped “molecule” using parameters (a) $A = \frac{1}{\sqrt{2}}$, $a = 1.25$ and $\alpha = 3.60$ and (b) $A = \frac{1}{\sqrt{2}}$, $a = 2.40$ and $\alpha = 13.36$. (Both these choices satisfy (9).) The reconstructed values for the two choices are practically identical, see (c) and (d) for gray-value displays of the central slices of the reconstructions. The artifact in (a) is due not so much to the location of the implicit surface (which is nearly identical for the two cases), but to the directions of the normals at the implicit surface.

The final image (the farther the object, the smaller it appears) it is possible to use a perspective projection in which all the rays cast from the screen intersect in a point called the center of projection [48]. Because for the visualization of biological molecules foreshortening does not appear to be important, we present only orthogonal projections.

For every ray $R$ that intersects $S$, we need to find the point $q$ in $R \cap S$ nearest to the screen, and compute its distance from the screen and the normal to $S$ at $q$ (these are used to assign an intensity value on the computer screen [48]). In practice, the finding of the points $q$ is computationally expensive. In general there is no prior estimate of how far $q$ is from the screen.

Clever procedures have been published in the literature that guarantee finding the intersection (if there is one). In most cases these require calculating derivatives associated with the density function $\nu$; this, in principle, is not a problem for us since $\nu$ is defined by (2) and the basis functions $b_j$ can be analytically differentiated based on (7). Such procedures include ones based on interval analysis [37] and ones that compute a (local) Lipschitz constant [16,22]. A practical difficulty that we perceived with such approaches when applied to our application is that we may end up with such small step sizes for the search as to make the procedure computationally unacceptable. For example, the sphere tracing procedure as described in [16] produces a step size that is inversely proportional to the Lipschitz constant. For our choices of the parameters $A$, $a$, and $\alpha$, there maybe more than 50 basis functions
Fig. 4. Representations of the implicit surface at level $t = 0.5$ for the combination of two blobs whose centers are immediate neighbors in the bcc grid $B_{1/\sqrt{2}}$ and whose coefficients are 1. We show three surfaces with different values of $a$ and $z$ satisfying (9) that produce a concave, a “just convex” and a “too convex” objects, respectively. The parameters $A, a,$ and $z$ used for (a) and (b) are the same as used for (a) and (b) in Fig. 3. The parameters used for (c) would also result in a smooth-looking visualization of the reconstructed sphere-shaped “molecule,” but it would appear larger than it should be.

making a nonzero contribution to $v(\mathbf{x})$ as defined by (2). Using the estimate provided by the fact that the Lipshitz constant of the sum is bounded by the sum of the Lipshitz constants (as suggested in Appendix D.1 of [16]), we get that the step size to be used in the search is inversely proportional to the sum of the 50 or so largest absolute values of the coefficients $c_j$ in (2). For this reason we looked for an alternative methodology that makes specific use of the nature of the representation (2) that is the output of the electron microscopic reconstruction process.

We first do a preprocessing of the set of grid points $\{\mathbf{p}_j\}$ at the end of which, for every pixel on the screen from which we cast a ray, we have the list of those grid points (arranged in order of increasing distance from the screen) whose associated coefficients can possibly influence the value of the distribution $v$ anywhere along the ray. (These grid points all lie within a cylinder of radius $a$ whose central axis is the ray in question.) This preprocessing is easily done by identifying on the screen the shadows of the blobs centered at the grid points, one-by-one in an appropriate order. In locating $\mathbf{q}$ for a particular ray, we make use of the associated list of grid points. For all grid points in the list (recall that these are arranged in order of increasing distance from the screen), we evaluate $v$ at the projection of the grid point onto the ray (for this we need the blob coefficients for only a few grid points, all of which are at similar positions in the list), until we find (if ever) two consecutive projections onto the ray, $\mathbf{q}_a$ and $\mathbf{q}_b$, such that the value of $v$ is below the threshold at $\mathbf{q}_a$ and is above it at $\mathbf{q}_b$. In fact, with the previously discussed methodology for the selection of the parameters $A, a,$ and $z$, there are up to only 51 values $c_j$ that contribute to the value of $v$ at any given point along a ray. Thanks to the radial symmetry and compact support of our basis functions, it is possible to create (just once) a 1D look-up table to represent the blob (7) at a very fine sampling of the radial distance $r$ and so rapidly compute the $c_j$ contributions. (Note that it is theoretically possible that there are two consecutive projections of grid points onto the ray such that the value of $v$ is below the threshold at both and yet it is above the threshold at some point between them. In this situation, our algorithm will miss this point of the surface and this will cause an inaccuracy in the displayed image of the surface. However the nature
of our blobs, as defined by (7), makes such an event unlikely: the values of the blobs peak at the grid points and they are circularly symmetric, consequently the contribution of any blob to the value of \( \tilde{v} \) on the ray is strongest at the projection onto the ray of the grid point that is at the center of the circular support of that blob.) This initial search for \( \overline{q_a} \) and \( \overline{q_b} \) is limited by a not very large constant (that is the same for all rays).

Once two such points \( \overline{q_a} \) and \( \overline{q_b} \) are found, then \( \overline{q} \) is located by a combination of the Newton–Raphson and the bisection methods [41] within the interval \([\overline{q_a}, \overline{q_b}]\) (for this we need the coefficients of only those blobs which were used for calculating \( \tilde{v} \) at \( \overline{q_a} \) and at \( \overline{q_b} \)). Under the right circumstances, the Newton–Raphson method has a quadratic rate of convergence, but it can also happen that it moves outside the interval \([\overline{q_a}, \overline{q_b}]\). In such a case, we continue our search for \( \overline{q} \) using the bisection method (which is slower but guarantees convergence) within the refined interval produced by the Newton–Raphson method.

With our choice of the basis functions \( b_j \), a function \( \tilde{v} \) defined by (2) is continuously differentiable and its gradient, at any point \( \overline{x} \), is given by

\[
\nabla \tilde{v}(\overline{x}) = \sum_{j=1}^{J} c_j \nabla b_j(\overline{x}).
\]

In our application, the set \( \{c_j\} \) is produced by the reconstruction algorithm and we have a closed formula to compute the \( \nabla b_j \) [28]. The visualization obtained by raycasting is therefore an accurate representation of the object of interest, limited only by the reconstruction process and the accuracy of the \( t \) in (1).

5. Visualization of reconstructed molecules

We obtained from the authors of [6] electron microscopic projections of the complex DnaB-DnaC of *Escherichia coli*; see Fig. 5. (DnaB and DnaC are two proteins that dock during DNA replication.) We applied the reconstruction method ART with blobs to these data using \( \lambda = \frac{1}{\sqrt{2}} \), \( a = 2.40 \) and \( \alpha = 13.36 \) (the selection of these values is justified by the discussion in Section 3; see, especially, Figs. 3 and 4). The reconstruction process took 65 h 22 min 48 s and produced a set \( \{c_j\} \) with 1,600,065 values. After reconstruction we applied the raycasting-blobs method just described. The display of the surface, at 512 × 512 pixels resolution, took 1 h 28 min 12 s (all times are for a single-Pentium4®-processor based computer with 2 GHz and 1 Gbytes of RAM operated under Linux®); see Fig. 6. (We consider that displaying the implicit surface at a resolution higher than 512 × 512 pixels to be impractical.)

![Fig. 5. Four of the images obtained by transmission electron microscopy (i.e., micrographs) of specimens of the DnaB-DnaC complex of *Escherichia coli* [6].](image-url)
is not justified by the quality of the data from which the reconstructions were produced; see Fig. 5.)

Because the exact shape of the DnaB-DnaC complex is unknown (i.e., there is no alternative gold-standard methodology which identifies the exact positions of its atoms), we decided to obtain also data from the molecule bacteriorhodopsin [27], for which there is a model of its atoms available in the protein data bank (PDB) [8]. This allows us to compare the rendering of the reconstructed surface with that based on the atomic model.

We used the PDB description file to simulate the acquisition process in transmission electron microscopy [12] with programs that permit us to calculate projections with noise from PDB models [13,32]. We utilized the conical tilt scheme (a common geometry in electron microscopy [12]) to create the projections. For the uncertainty of angles we used ±5 degrees for rotation and ±1 degree for tilt, these values are suggested in [31]. For the level of noise in the pixels, we selected the value of 0.5 SNR (see Fig. 7), a value within the range presented in [12] for uncorrelated additive noise with zero mean.

We applied ART with blobs to the bacteriorhodopsin data (using the same parameters as used for the DnaB-DnaC reconstruction). The reconstruction process took 36 h 13 min 36 s and produced a set \( \{c_j\} \) with 1,482,624 values. Once again, we applied the raycasting-blobs method described above. In this case, the display of the surface took 1 h 37 min 55 s; see Fig. 8(b).

The raycasting method of Section 4 for visualizing implicit surfaces is computationally demanding because of the search, for every ray, for the intersection point \( q \). The times just reported above (over 90 min in both cases) are too long for routine generation of multiple views. In the next section we consider a method to accelerate the raycasting-blobs technique by taking advantage of the properties of the representation in (2) when the \( b_j \)s are blobs based on (7).
Fig. 7. Projections (a) without noise and (b) with noise of the molecule bacteriorhodopsin created using an atomic description [8] and simulation programs [13,32].

Fig. 8. (a) Rendering of an implicit surface of the bacteriorhodopsin reconstructed from noiseless projections, some of which are shown in Fig. 7(a). The surface has the same orientation and encloses the same volume as that shown in (b), and serves as the representation of truth. (b) Rendering of an implicit surface of the bacteriorhodopsin reconstructed from noisy projections, some of which are shown in Fig. 7(b). We can compare (b) with the “real” shape of the molecule in (a).

6. Method to speed up the raycasting-blobs technique

Our original implementation of the raycasting-blobs method searches, for every pixel of the image displaying the surface, for two points $q_a$ and $q_b$ that are end points of a line segment containing the point $q$ such that $v(q) = t$. This search is computationally expensive as there are no prior estimates of how far the points $q_a$ and $q_b$ are from the screen. In fact, the most computationally intensive part of the whole raycasting process is the identification of the interval $[q_a, q_b]$ that contains an intersection (if there is such an interval); the actual search for the intersection within the interval can be done in a stable and efficient manner.
We now introduce a preprocessing method that finds an estimate of the point \( \overline{q} \), for every casted ray in the image. It first calculates, for every grid point \( \overline{p}_j, v_j = v(\overline{p}_j) \) and then proceeds to utilize both the set \( \{v_j\} \) and a list of all the \( j \), such that \( v_j \geq t \) in a z-buffer algorithm [48] that operates as follows. A value is assigned to every ray of the original raycasting-blobs algorithm of Section 4. Initially this value is “infinity” (in practice, a very large number). Then we loop through all \( j \) such that \( v_j \geq t \). For each corresponding blob, we calculate the distance \( d_j \) of \( \overline{p}_j \) from the screen and, for all rays which intersect the blob, we replace the currently assigned value by \( d_j \) if, and only if, \( d_j \) is smaller than the currently assigned value. Upon completion of this process, the value assigned to any ray will be an approximation to the distance \( \overline{q}_b \) for that ray in the raycasting algorithm.

The new raycasting-blob method utilizes only those rays whose corresponding value in the z-buffer is different from infinity. For every ray of this set, we assume that the value stored in the z-buffer provides a point \( \overline{q}_c \) that in most of the cases is near to point \( \overline{q}_b \). We check whether the condition \( v(\overline{q}_c) \geq t \) is true. When the condition is not true, we search for \( \overline{q}_a \) and \( \overline{q}_b \) as we did in the original implementation but starting from the point \( \overline{q}_c \) (which is a projection of a grid point in the list associated with the casted ray). In the alternate case, we search for the points \( \overline{q}_a \) and \( \overline{q}_b \) in the direction toward the plane, again starting from \( \overline{q}_c \).

Once the points \( \overline{q}_a \) and \( \overline{q}_b \) are found, the point \( \overline{q} \) is located as in the original implementation in Section 4.

The preprocessing (the calculation of the \( v_j \)) took 5 s for the set \( \{c_j\} \) corresponding to the DnaB-DnaC and 6 s for the bacteriorhodopsin. After creating the set \( \{v_j\} \), we used our modified raycasting method to obtain the computer representation of both the macromolecular complexes DnaB-DnaC and bacteriorhodopsin. When compared point-by-point with those created with the raycasting-blobs method introduced in Section 4 (see Figs. 6 and 8(b)), we found that the images are exactly the same. However, the times to generate the images in Figs. 6 and 8(b) by the original method are around 1.5 h, while the times to generate them by the new method are 94 s and 86 s, respectively; thus reducing the computing time by a factor between 57 and 69, see Table 1.

In order to analyze the speed of the new ray casting algorithm, we assume that there are \( J \) blobs produced by the reconstruction algorithm and \( N \times N \) pixels in the image to be displayed. The z-buffer algorithm runs linearly in \( J \). After that, for each of the \( N^2 \) rays, we do calculations that is proportionately bounded above by the number of blobs intersected by the ray, and this is of the order \( J^{3 \frac{1}{3}} \). Assuming, as is reasonable, that \( N \) is also of the order \( J^{\frac{1}{3}} \), the whole algorithm is seen to be linear in \( J \).

We also compared the results of our approach to those produced by two software packages: OpenDX [1] (a freely-available and popular package, formerly IBM Data-Explorer) and Amira® [4] (an integrated software package for 3D visualization and volume modeling that is frequently used in engineering, biological and medical laboratories). Both of these can produce renderings of polygonal approximations of surfaces. The input to the programs requires evaluating \( v \) at points of a simple cubic grid; such can easily be achieved using (2). We found that in order to obtain image quality similar to what was obtained using our raycasting-blobs algorithms, we needed to create the input to both programs at a minimum of \( 400 \times 400 \times 400 \) points. The resulting images (to be compared
Table 1

<table>
<thead>
<tr>
<th>Method</th>
<th>Process</th>
<th>DnaB-DnaC</th>
<th>Bacteriorhodopsin</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenDX</td>
<td>Loading</td>
<td>12 s</td>
<td>10 s</td>
</tr>
<tr>
<td></td>
<td>Polygonization</td>
<td>43 s</td>
<td>45 s</td>
</tr>
<tr>
<td></td>
<td>Rendering</td>
<td>1 s</td>
<td>1 s</td>
</tr>
<tr>
<td>Amira®</td>
<td>Loading</td>
<td>6 s</td>
<td>6 s</td>
</tr>
<tr>
<td></td>
<td>Polygonization</td>
<td>34 s</td>
<td>33 s</td>
</tr>
<tr>
<td></td>
<td>Rendering</td>
<td>1 s</td>
<td>1 s</td>
</tr>
<tr>
<td>Raycasting-blobs original</td>
<td>Loading</td>
<td>1 s</td>
<td>1 s</td>
</tr>
<tr>
<td></td>
<td>Raycasting</td>
<td>1 h 28 min 12 s</td>
<td>1 h 37 min 55 s</td>
</tr>
<tr>
<td></td>
<td>Rendering</td>
<td>1 s</td>
<td>1 s</td>
</tr>
<tr>
<td>Raycasting-blobs speeded</td>
<td>Loading</td>
<td>1 s</td>
<td>1 s</td>
</tr>
<tr>
<td></td>
<td>PreProcessing</td>
<td>5 s</td>
<td>6 s</td>
</tr>
<tr>
<td></td>
<td>Raycasting</td>
<td>1 min 27 s</td>
<td>1 min 18 s</td>
</tr>
<tr>
<td></td>
<td>Rendering</td>
<td>1 s</td>
<td>1 s</td>
</tr>
</tbody>
</table>

with Figs. 6 and 8(b)) are shown in Fig. 9. The timings for this software are also reported in Table 1. While the total time for generating a single view is only about twice as fast when using OpenDX or Amira® than when using the speeded raycasting-blobs method, if multiple views are desired then the polygon-based techniques are much faster, since (once the polygonization is completed) the projection of the polygons for a new point of view requires only approximately a second.

We also compared the memory requirements of the various approaches; see Table 2. We found that our new (speeded) implementation requires actually less memory than the original raycasting-blobs approach and that both raycasting-blobs methods require significantly less memory than the programs OpenDX and Amira®. It is easy to see that, under the same assumptions that were made for the time-complexity analysis, the space-complexity of the raycasting-blobs algorithms is also linear in $J$.

7. Summary

At the end of the electron microscopic reconstruction process, biological macromolecules are often represented as linear combinations of well over a million special basis functions called blobs. The implicit surface separating the reconstructed molecule from its background can be accurately visualized by raycasting. Our initial implementation of this was too slow to allow user interaction. However, one can take advantage of the nature of the blob representation to reduce the time required to visualize molecules by more than a factor of fifty. Compared to popular visualization programs that produce renderings of polygonal approximations of the surfaces, our new implementation is slower but requires less memory to display the implicit surface (rather than a polygonal approximation of it).
Fig. 9. Visualizations of the explicit surfaces obtained from the reconstructions of the macromolecular complexes DnaB-DnaC (upper row) and bacteriorhodopsin (bottom row). Images in (a) and (c) were produced using the software OpenDX [1] while those in (b) and (d) were produced with Amira\textsuperscript{{\textregistered}} [4], based in both cases on a $400 \times 400 \times 400$ voxelized distribution.

An important advantage of using direct methods to visualize implicit surfaces is that surfaces in nature are frequently highly complex and in some circumstances the programs in charge of polygonization of the implicit surface might generate the wrong representation of the underlying surface or violate the underlying topology of the natural surface. Since the method presented here relies in the model used to reconstruct the biological object, the final representation will depend only on the sampling used between pixels; thus producing, in principle, an accurate representation of the biological surface.
Table 2
Memory used to generate images of the complex DnaB-DnaC and the bacteriorhodopsin by both raycasting-blobs implementations and by two programs for scientific visualization: OpenDX and Amira

<table>
<thead>
<tr>
<th>Method</th>
<th>Process</th>
<th>DnaB-DnaC</th>
<th>Bacteriorhodopsin</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenDX</td>
<td>Digitization</td>
<td>$400 \times 400 \times 400$</td>
<td>$400 \times 400 \times 400$</td>
</tr>
<tr>
<td></td>
<td>Loading</td>
<td>254 Mbytes</td>
<td>254 Mbytes</td>
</tr>
<tr>
<td></td>
<td>Polygonization</td>
<td>514 Mbytes</td>
<td>512 Mbytes</td>
</tr>
<tr>
<td></td>
<td>Rendering</td>
<td>518 Mbytes</td>
<td>516 Mbytes</td>
</tr>
<tr>
<td>Amira\textsuperscript{a}</td>
<td>Digitization</td>
<td>$400 \times 400 \times 400$</td>
<td>$400 \times 400 \times 400$</td>
</tr>
<tr>
<td></td>
<td>Loading</td>
<td>245 Mbytes</td>
<td>245 Mbytes</td>
</tr>
<tr>
<td></td>
<td>Polygonization</td>
<td>292 Mbytes</td>
<td>293 Mbytes</td>
</tr>
<tr>
<td></td>
<td>Rendering</td>
<td>292 Mbytes</td>
<td>293 Mbytes</td>
</tr>
<tr>
<td>Raycasting-blobs original</td>
<td>Set of coefficients</td>
<td>1,600,065</td>
<td>1,482,624</td>
</tr>
<tr>
<td></td>
<td>Loading</td>
<td>9028 bytes</td>
<td>8536 bytes</td>
</tr>
<tr>
<td></td>
<td>Raycasting</td>
<td>47,152 bytes</td>
<td>43,596 bytes</td>
</tr>
<tr>
<td></td>
<td>Rendering</td>
<td>11,956 bytes</td>
<td>11,548 bytes</td>
</tr>
<tr>
<td>Raycasting-blobs speeded</td>
<td>Set of coefficients</td>
<td>1,600,065</td>
<td>1,482,624</td>
</tr>
<tr>
<td></td>
<td>Loading</td>
<td>15,312 bytes</td>
<td>14,484 bytes</td>
</tr>
<tr>
<td></td>
<td>PreProcessing</td>
<td>16,164 bytes</td>
<td>15,284 bytes</td>
</tr>
<tr>
<td></td>
<td>Raycasting</td>
<td>45,292 bytes</td>
<td>41,688 bytes</td>
</tr>
<tr>
<td></td>
<td>Rendering</td>
<td>18,248 bytes</td>
<td>17,364 bytes</td>
</tr>
</tbody>
</table>

Acknowledgements

We thank Carlos Oscar Sánchez Sorzano and José Jesús Fernández for helpful advice regarding this manuscript. We appreciate the assistance of Montserrat Bárcena in providing us with data and information regarding the macromolecular complex DnaB-DnaC.

References


