1. Tosaki, A., Balint, S. and Szekeres, L., *J. Cardiovasc. Pharmacol.*, 1988, **12**, 621-628.
2. Clarkson, C. W., Follmer, C. H., TenEick, R. E., Hondeghem, L. M. V. and Yeh, J. Z., *Circ. Res.*, 1988, **63**, 869-878.
3. Zivanovic, L., Zivanov-Stakic, D. and Radulovic, D., *J. Pharm Biomed. Anal.*, 1988, **6**, 809-812.
4. Blyden, G. T., Greenblatt, D. J., LeDuc, B. W. and Scavone, J. M., *Eur. J. Clin. Pharmacol.*, 1988, **35**, 413-417.
5. DeLangen, C. D. J., Balke, C. W., Spear, J. F., Levine, J. H. and Moore, E. N., *J. Cardiovasc. Pharmacol.*, 1988, **12**, 683-688.
6. Onuaguluchi, G. and Igbo, I. N. A., *Arch. Int. Pharmacodyn. Ther.*, 1985, **274**, 253-266.
7. Martins, J. B. and Kelly, K. J., *Am. Heart J.*, 1985, **109**, 533-539.
8. Hinton, R. J., Dechow, P. C. and Carlson, D. S., *Oral Surg., Oral Med. Oral Pathol.*, 1985, **59**, 247-251.
9. Merrifield, A. J. and Carter, S. J., *Anaesthesia*, 1965, **20**, 287-289.
10. Hargrove, R. L., Hoyle, J. R., Parker, J. B. R., Beckett, A. H. and Boyes, R. N., *Anaesthesia*, 1966, **21**, 37-40.
11. Foldes, F. F., Molly, R., Mcnall, P. G. and Koukal, L. R., *J. Am. Med. Assoc.*, 1960, **172**, 1493-1498.
12. Bromage, P. R. and Robson, J. G., *Anaesthesia*, 1961, **16**, 461-463.
13. Ritchie, J. M. and Ritchie, B. R., *Science*, 1968, **162**, 1394-1395.
14. Buchie, J. and Perlia, X., *Arzneim-Forsch.*, 1962, **10**, 1-8, 117-124, 174-177, 297-301, 456-467, 554-559.
15. Ritchie, J. M. and Greengard, P., *Annu. Rev. Pharmacol.*, 1966, **6**, 405-409.
16. Thorsteinn, L., Masaaki, M., Caldwell, R. W., Gildersleeve, N. and Bodor, N., *Int. J. Pharm.*, 1984, **22**, 345-355.
17. Prippenow, G., Fruhstorfer, H., Seidlitz, P. and Nolte, H., *Reg. Anaesth. (Berlin)*, 1985, **8**, 15-20.
18. Pajunen, A., *Acta Crystallogr.*, 1982, **B38**, 928-929.
19. Gutkoska, R., Lyford, P. and Zubieta, J. A., *Cryst. Struct Commun.*, 1982, **11**, 1311-1316.
20. Abello, L., Ensuque, A., Demaret, A. and Lapluye, G., *Transition Met. Chem.*, 1980, **5**, 120-121.
21. Pavelcik, F. and Majer, J., *Acta Crystallogr.*, 1980, **B36**, 1645-1646.
22. Anan'eva, N. N., Polyakova, I. N., Polynova, T. N. and Porai-Koshits, M. A., *Koord. Khim.*, 1981, **7**, 1578-1584.
23. Petrovic, D., Leovac, V. M. and Lazar, D., *Cryst. Struct. Commun.*, 1981, **10**, 823-826.
24. Bertrand, J. A., Fujita, E. and Van Derveer, D. G., *Inorg. Chem.*, 1980, **19**, 2022-2028.
25. Cameron, A. F., Taylor, D. W. and Nuttall, R. H., *J. Chem. Soc. Dalton Trans.*, 1972, **15**, 1603-1608.
26. Coetzer, J., *Acta Crystallogr.*, 1970, **B26**, 1414-1417.
27. Pabst, I. and Bats, J. W., *Acta. Crystallogr.*, 1985, **C41**, 1297-1299.
28. Saha, C. R., Sen, D. and Guha, S., *J. Chem. Soc. Dalton Trans.*, 1975, **16**, 1701-1706.
29. Stephens, F. S. and Tucker, P. A., *J. Chem. Soc., Dalton Trans.*, 1973, **21**, 2293-2297.
30. Matsumoto, K., Ooi, S., Nakasuka, K., Mori, W., Suzuki, S., Nakahara, A. and Nako, Y., *J. Chem. Soc. Dalton Trans.*, 1985, **10**, 2095-2100.
31. Bailey, N. A., Mckenzi, E. D. and Worthington, J. M., *J. Chem. Soc., Dalton Trans.*, 1973, **11**, 1227-1231.
32. Bertini, I., Dapporto, P., Gatteschi, D. and Scozzafava, A., *J. Chem. Soc., Dalton Trans.*, 1979, **9**, 1409-1414.
33. Xinmin, G., Ning, T., Xin, W., Yin, Z. and Minyu, T., *Polyhedron*, 1989, **8**, 933-935.
34. Sridhar, M. A., Babu, A. M., Indira, A., Bellad, S. B., Shashidhara Prasad, J., Ramappa, P. G. and Nagendrappa, G., *Z. Kristallogr.*, 1992, **202**, 292-295.
35. Germain, G., Main, P. and Woolfson, M. M., *Acta. Crystallogr.*, 1971, **A27**, 368-378.
36. Gabe, E. J., Le Page, Y., Charland, J. P., Lee, F. L. and White, P. S., *J. Appl. Crystallogr.*, 1989, **22**, 384-387.
37. Bellad, S. B., Indira, A., Sridhar, M. A., Babu, A. M., Shashidhara Prasad, J. and Prout, C. K., communicated.
38. Barclay, G. A., Sabine, T. M. and Taylor, J. C., *Acta. Crystallogr.*, 1965, **19**, 205-209.
39. Drew, M. G. B., Nelson, J. and Nelson, S. M., *J. Chem Soc., Dalton Trans.*, 1981, **8**, 1685-1690.
40. Palenik, G. J., Koziol, A. E., Gawron, M., Palenik, R. C. and Wester, D. W., *Acta. Crystallogr.*, 1988, **C44**, 85-88.

ACKNOWLEDGEMENTS. We (A.I., A.M.B., S.B.B. and M.A.S.) thank UGC, New Delhi for financial assistance. We also thank Professor P. G. Ramappa, Department of Studies in Chemistry, University of Mysore, Mysore for kindly providing the sample. J.S.P. thanks DST, New Delhi for the financial assistance under the project No. SP/S2/M-34/86.

Received 14 August 1992; revised accepted 23 November 1992

Visualization of three-dimensional data by 'volume rendering'

N. Ramesh and G. Athithan
Advanced Numerical Research and Analysis Group,
P. O. Kanchanbagh, Hyderabad 500 258, India

We report an implementation of a ray-tracing method for visualizing volumetric data. For evaluating the performance of the implementation, we consider the visualization of two classes of volumetric data. The first consists of the probability density functions of a subset of states of an electron in hydrogen atom and represents the case of visualizing data derived from analytical expressions. The second class of volumetric data is a set of three-dimensional fractals which are generated over regular three-dimensional cartesian grids of various sizes. The motivation for generating and visualizing the fractal data sets is two-fold. Firstly a fractal data set is a good test input to verify the correctness of the implementation. Secondly, we want to model stellar clouds and see how they can be visualized. Clouds are known to be akin to fractal data sets and we find that the volume rendered images of these data sets bear a close resemblance to stellar clouds. Ray-tracing is computationally expensive and we therefore report the CPU timings for a representative set of images. General comments on the utility of this method for visualizing in various disciplines conclude the paper.

REPRESENTATION and visualization of data in a graphical form is a practice as old as experimental science. Whether it is an array of numbers or a mathematical function, a graphical plot of the same gives better insight. However, before the advent of computer graphics, one was limited to visualizing the relationship between just two variables in a graphical form. Vizualization of any relationship among three variables usually called for highly skilled draftsmen and that among four variables was out of the question. With the application of computer graphics, the problem of visualizing trivariate relationships has been solved in a number of ways. The key task in all the solutions is the hidden element (line, surface) removal. Currently, one of the main preoccupations in computer graphics is to represent and visualize relationships between four variables or what is more commonly known as volumetric data visualization. The data set to be visualized comes in the form of scalar values at each point of a regular or irregular, three-dimensional grid. Our discussion here is, however, confined to regular cartesian grids. The values at the grid points may come from an experiment such as magnetic resonance imaging of human bodies or may be calculated from a trivariate analytic function.

One of the standard methods of visualizing such a grid of data is to construct surfaces inside the volumetric grid, over which the scalar value remains invariant. Such surfaces, commonly known as 'isovalue surfaces' can then be visualized graphically by means of various surface display techniques. A single isovalue surface does not suffice to represent the entire volumetric data, and so one may construct many such surfaces simultaneously, for different isovalues so as to visualize them all together. Usually this approach works when the data vary somewhat smoothly over the grid. However, if the data are poor, noisy or vary rapidly from point to point, then one finds spurious additions or deletions from the isovalue surfaces. This happens due to the binary nature of the classification of the cells of the grid into those that are intercepted by the isovalue surface and those that are not. False negatives contribute to holes while false positives may cause spurious additions[1]. To circumvent this problem, researchers recommend the concept of 'volume rendering' wherein no intermediate geometric construction is carried out[2]. In this approach screen images are computed directly from the volumetric data by means of ray-tracing. The main advantage of this approach is that it generates a mechanism for extracting and displaying weak or fuzzy features out of the volumetric data. Here one can emulate the technique of isovalue surface visualization without the problems of spurious additions or deletions, as shown in ref. 1. Alternatively one can render the entire volume of data as we demonstrate below using two classes of volume data.

After a survey of the literature for a suitable algorithm for volume rendering we chose to implement the one suggested by Marc Levoy[1]. While he concentrates more on the isovalue surface visualization by means of volume rendering, our interest here is to carry out entire volume visualization. We give a brief outline of his method, modified to suit our need, in what follows. The interested reader may refer to his paper for further details.

The basic input to the algorithm is a regular three-dimensional grid of scalar values. In case the grid is irregular, it has to be transformed to a regular grid in a separate preprocessing step. In the subsequent discussion, the grid points are referred to as voxels (volume elements) and the scalar value is referred to as intensity. They are denoted by g_{ijk} and $f(g_{ijk})$ respectively. The algorithm has two functional stages. In the first stage, all the grid points are assigned a colour value C_β (β = red, green, blue) (shading) and an opacity value α (classification). Both the shading and classification calculations depend heavily on the normal at that point. In the absence of an explicit geometry, the normal is computed from the intensity gradient using

$$N(g_{ijk}) = \frac{\delta f(g_{ijk})}{|\delta f(g_{ijk})|},$$

where the intensity gradient $\delta f(g_{ijk})$ at g_{ijk} is computed as

$$\begin{aligned}\delta f(g_{ijk}) = [&1/2\,(f(g_{i+1,j,k}) - f(g_{i-1,j,k})),\\ &1/2\,(f(g_{i+1,j,k}) - f(g_{i-1,j,k})),\\ &1/2\,(f(g_{i+1,j,k}) - f(g_{i-1,j,k}))].\end{aligned}$$

The colour value C_β at g_{ijk} is computed using the following colour model.

$$\begin{aligned}C_\beta(g_{ijk}) = C_{s,\beta}\,k_{a,\beta} + C_{s,\beta}\,[&k_{d,\beta}\,(N(g_{ijk})\cdot L)\\ &+ k_{s,\beta}(N(g_{ijk})\cdot H)^n],\end{aligned}$$

where, $C_\beta(g_{ijk})$ is the βth colour component at voxel location g_{ijk}; $C_{s,\beta}$ the βth colour component of the light source at infinity; $k_{a,\beta}$ the ambient reflection coefficient for βth colour component; $k_{d,\beta}$ the diffuse reflection coefficient for βth colour component; $k_{s,\beta}$ the specular reflection coefficient for βth colour component; $N(g_{ijk})$ the normal vector at g_{ijk}, L the normalized vector in the direction of light source; H the normalized vector in the direction of maximum highlight and n the exponent used in highlight approximation.

The highlight vector H is computed as

$$H = \frac{V + L}{|V + L|}$$

where V is the normalized vector in the direction of the observer.

The opacity value of a voxel is a measure of its transparency. A value of zero means the voxel is fully transparent (and hence invisible) and one means it is fully opaque (thus hiding everything behind it). Values between 0 and 1 mean the voxel is translucent. For entire volume rendering the opacity value of a voxel is assigned in proportion to its intensity value. This makes the high intensity voxels more opaque (and hence seen more) and low intensity voxels more transparent. That is,

$$\alpha(g_{ijk}) = (f(g_{ijk}) - f_{\min})/(f_{\max} - f_{\min})$$

Here $f_{\min}$ and $f_{\max}$ are the minimum and maximum intensity values in the grid.

To avoid voxels having intensity value equal to $f_{\min}$ from becoming fully transparent (and hence not seen at all), instead of using the complete unit interval [0–1] for opacity value assignment, a sub interval within [0–1] may be used. That is,

$$\alpha(g_{ijk}) = p + \frac{(f(g_{ijk}) - f_{\min})}{f_{\max} - f_{\min}}(q - p),$$

where $0 < p, q \leq 1$, $p < q$.

Once the colour and opacity values are assigned to all voxels in the volume, a beam of rays is sent parallel to the eye vector through the viewing plane into the volumetric grid. In this stage, a colour for each ray is computed and the pixel in the viewing plane through which the ray passes is assigned that colour. The viewing plane is analogous to the screen.

Along each ray, the volume is resampled at K equally spaced intervals. This interval is taken as the minimum distance inside a cell bounded by eight voxels. Colour and opacity value at a sample point are calculated by trilinearly interpolating from the eight neighbouring voxels that surround the point. Finally, a fully opaque background ($C_{bg,\beta}$) is draped behind the volume as the last sampled point of all the rays. These colour and opacity values along ray are denoted as $C_\beta(R^i_{xy})$ and $\alpha(R^i_{xy})$ and colour for the entire ray is denoted as $C_\beta(R_{xy})$. R_{xy} denotes the ray passing through (x, y) position of the viewing plane and R^i_{xy} denotes the ith sample point on the ray R_{xy}. From this array of colour and opacity values, a single colour is computed per ray by a process of compositing. More specifically, the colour $C_{\text{out},\beta}(R^i_{xy})$ of the ray as it leaves sample point i is related to the colour $C_{\text{in},\beta}(R^i_{xy})$ of the ray as it enters that point and the colour and opacity values $C_\beta(R^i_{xy})$, $\alpha(R^i_{xy})$ at that sample point, by the transparency formula

$$C_{\text{out},\beta}(R^i_{xy}) = C_{\text{in},\beta}(R^i_{xy})(1 - \alpha(R^i_{xy})) + C_\beta(R^i_{xy})\alpha(R^i_{xy}).$$

For an array of sample values, compositing is done by repeated use of the above formula. That is

$$C_\beta(R_{xy}) = \sum_{j=K}^{0} C_\beta(R^j_{xy})\alpha(R^j_{xy}) \prod_{i=j-1} (1 - \alpha(R^i_{xy})).$$

Note that the compositing is done in back-to-front order (i.e. from the background screen towards the eye).

After the implementation of the above algorithm, we tried its utility in visualizing two classes of data. The first one was the class of probability density functions of an electron in hydrogen atom. The second class, a set of fractals, was generated artificially over a regular grid using the mid-point displacement method[3]. While the former represents a multivariate relationship whose form is analytically known, the latter is a case of discrete samples of data akin to what one obtains from numerical computations or experimental measurements.

It is well-known that the probability density function of an electron in hydrogen atom is given by the square of the modulus of the corresponding wave function $\psi_{nlm}(r, \theta, \varphi)$. Here nlm denote the quantum numbers of the state of the electron. In analytical form (ignoring scale factors)

$$\psi_{nlm}(r, \theta, \varphi) = C\,e^{-r/2}\,r^l L^{2l+1}_{n+l}(r) P^m_l(\cos\theta).e^{im\varphi},$$

where L^{2l+1}_{n+l} and P^m_l are the associated Laguerre polynomials and the associated Legendre polynomials respectively. C is the normalization constant which is in general a function of the quantum numbers. The factors $P^m_l(\cos\theta)\,e^{im\varphi}$ together constitute what are known as the spherical harmonics $Y_{lm}(\theta, \varphi)$ which express the angular dependance of the corresponding wavefunction. For visualizing the electron probability density function, the input data are obtained by computing $|\psi|^2$ over a regular cartesian grid of points for a given triplet of quantum numbers. Using the series expansions given in Morse and Feshbach[4] we could evaluate both L^{2l+1}_{n+l} and P^m_ls without the need for computing large factorials to compose the summands. The associated Laguerre polynomials are given by

$$L^q_p(r) = C_1 F(-p \mid q + 1 \mid r),$$

where $F(a \mid b \mid r)$, the confluent hypergeometric function is evaluated using

$$F(a|b|r) = 1 + \frac{a}{b}r + \frac{a(a+1)}{2!\,b(b+1)}r^2 + \frac{a(a+1)(a+2)}{3!\,b(b+1)(b+2)}r^3 + \cdots$$

For the associated Legendre polynomials we used

$$P^m_l(z) = (1 - z^2)^{m/2}\; T^m_{l-m}(z),$$

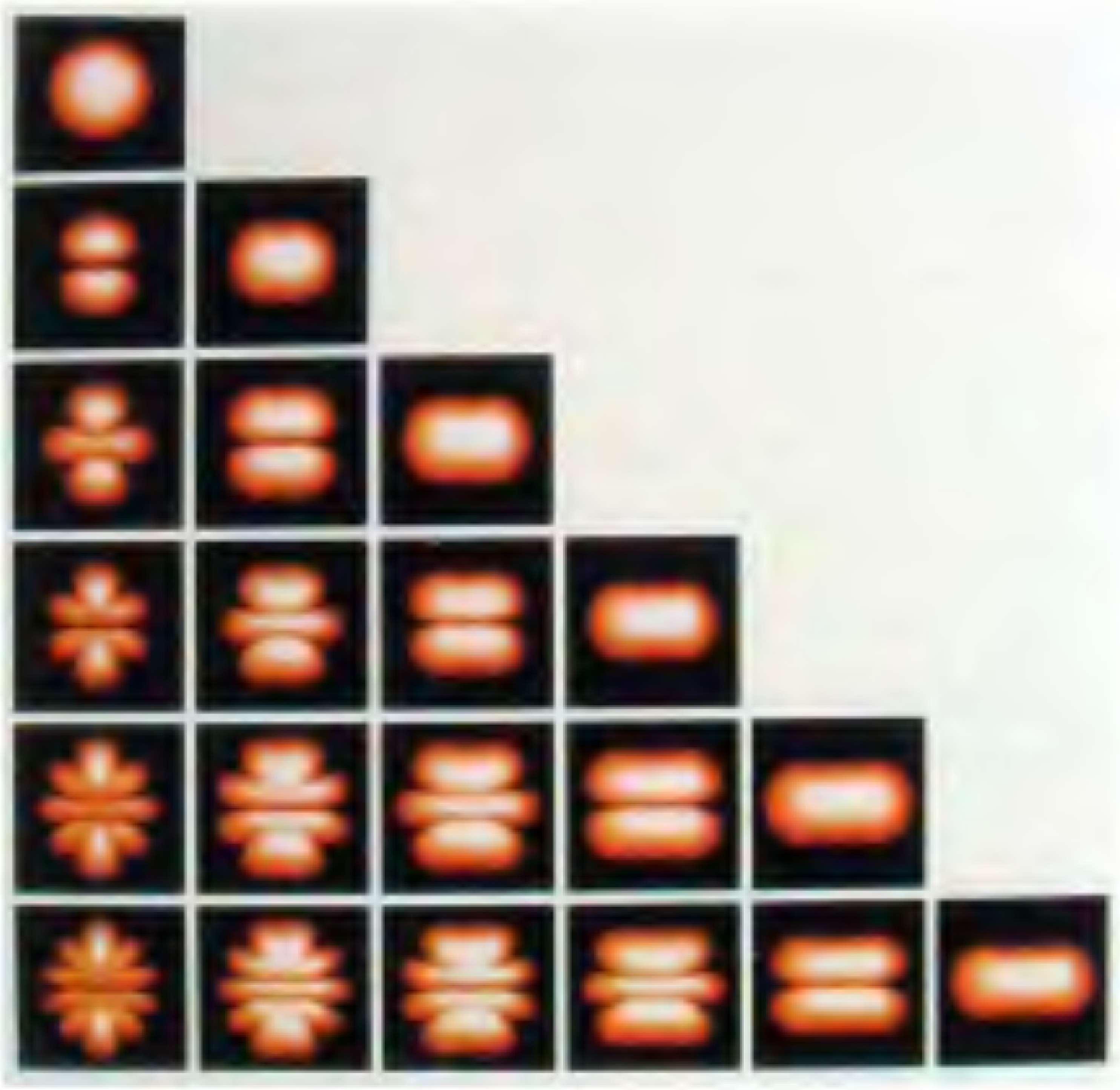
where

$$T_n^m(z) = C_2\left[z^n - \frac{n(n-1)}{2(2n+2m-1)} z^{n-2} + \frac{n(n-1)(n-2)(n-3)}{2.4(2n+2m-1)(2n+2m-3)} z^{n-4} \ldots \right]$$

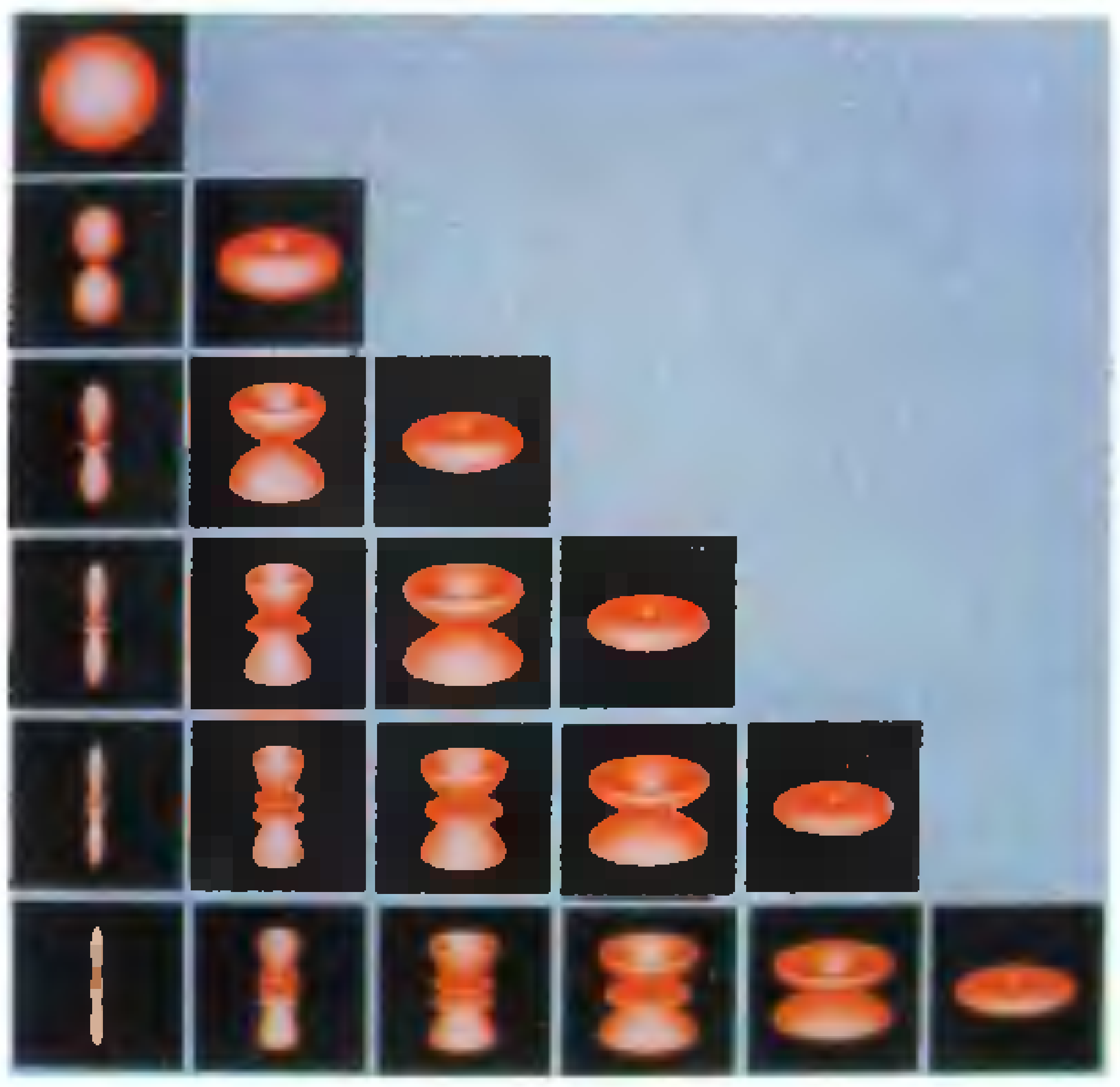
C_1 and C_2 are constants of proportionality which can be safely ignored. With the help of the above expressions, the probability density was computed over a grid of size 50 × 50 × 50. The exact computation of the overall normalization constant was omitted. Instead, a suitable scale factor was found out by trial and error, so that a substantial part of the density function would be enclosed within the chosen volumetric grid. The resulting volumetric data were rendered by the ray tracing algorithm over a screen window of size 120 × 120 picture elements, or pixels for short. Figure 1 *a* shows the set of probability density functions for all possible quantum states when $n=6$ and m is positive. The view point is on the x-axis and the origin is the view reference. In Figure 1 *b*, the same set of functions have been volume rendered from a different viewpoint (150, 150, 500). For the purpose of a structural comparison we also created images of the angular part of the probability density functions, i.e. $|Y_{lm}(\theta,\varphi)|^2$ with the aid of a surface rendering program. These are shown in Figure 1 *c*.

The class of volumetric data that we tried to visualize next, was generated by means of the mid-point displacement technique. Employed for creating fractal data sets, this technique is described by Barnsley *et al*

a



c



b

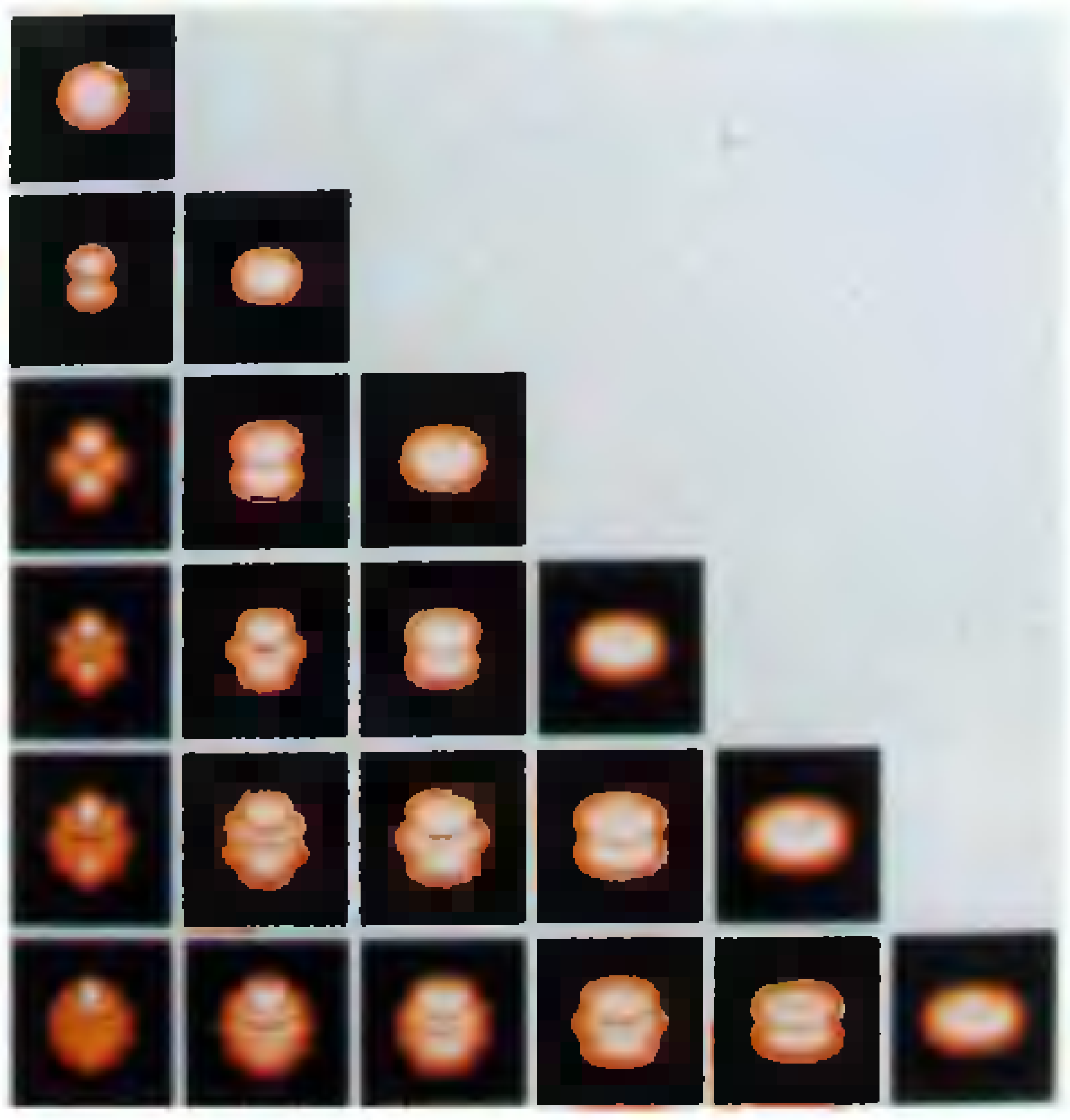


Figure 1. ***a***, Entire volume rendering of all possible probability density functions $|\psi_{nlm}|^2$ for $n=6$ and positive m. While l varies from 0 to $n-1$ along the column, m varies from 0 to l along each row. The axis of symmetry is z and the viewpoint is on the x-axis. ***b***, Entire volume rendering of the same functions as in ***a***, seen from a different viewpoint (150, 150, 500). ***c***, Surface rendering of the angular part ($|Y_{lm}(\theta,\varphi)|^2$) of the $|\psi_{nlm}|^2$ functions shown in ***a*** and ***b*** seen from (150, 150, 500)

for the case of one and two dimensions. We extended their algorithm in a natural manner for the case of three dimensions. One of our motivations to generate fractal data sets over a three-dimensional grid was to subject our implementation of Marc Levoy's algorithm to a good test. Besides, we were also hoping to create cloud-like images as a result of volume rendering a fractal volumetric data set. After experimenting with various values for the fractal dimensions we settled for a value of 3.7. For the standard deviation of the Gaussian random additions to the mid-points we chose a value of 1.0, while the mean was set to be 0. Use of different seeds for the random number generator was employed to generate different fractal data sets. With a view to creating some empty space in the grid, a suitable threshold value was subtracted uniformly from the values at every point and among the resulting values the negative ones were replaced by zeros. To ward off some undesirable boundary effects, the data were modulated by a decreasing ramp before thresholding. Figure 2*a* was created using one set of fractal data while Figure 2*b* using another. One cannot fail to notice the resemblance between these images and stellar clouds.

All the images were generated using the IRIS workstation 4D/20. The demands on the computing resources for rendering volumetric data are enormous. The summary of our observations with regard to the computing times required for various cases is given in Table 1. It may be relevant to mention here that the LINPACK rating of the IRIS 4D/20 workstation is about 0.9 MFLOPS and the computations to be performed in our implementation of ray tracing are LINPACK-like. It may also be noted that it is the floating-point computation speed of the system which is crucial as against the graphic performance. One sure way of speeding up the rendering is to use a parallel computer. Since the ray tracing algorithm is easily parallelizable, we are in the process of exploiting PACE[5] type of parallel computers for speeding up our implementation; results will be reported later.

Table 1. CPU timings for volume rendering the various test input data described in the test. The timing for the probability density functions refers to the case of $|\psi_{612}|^2$

Type of data set	Grid size	Window size	Time (sec)
Probability density functions	$50 \times 50 \times 50$	120×120	3.35
Fractal set-I	$65 \times 65 \times 65$	240×240	18.7
Fractal set-II	$65 \times 65 \times 65$	240×240	16.9
Fractal set-II	$129 \times 129 \times 129$	480×480	149.5

The need to visualize volumetric data, is met with in a number of disciplines, crystallography and radiology being two examples. In the former, one obtains the electron density distribution within the unit cell of a crystal to be studied. The volume rendering method implemented by us will be useful in this case. Likewise, the electron density distributions in crystals as obtained from band structure calculations can also be visualized. Indeed, such a visualization would be particularly useful when one repeats the calculation for various values of the unit cell parameters and tries to see

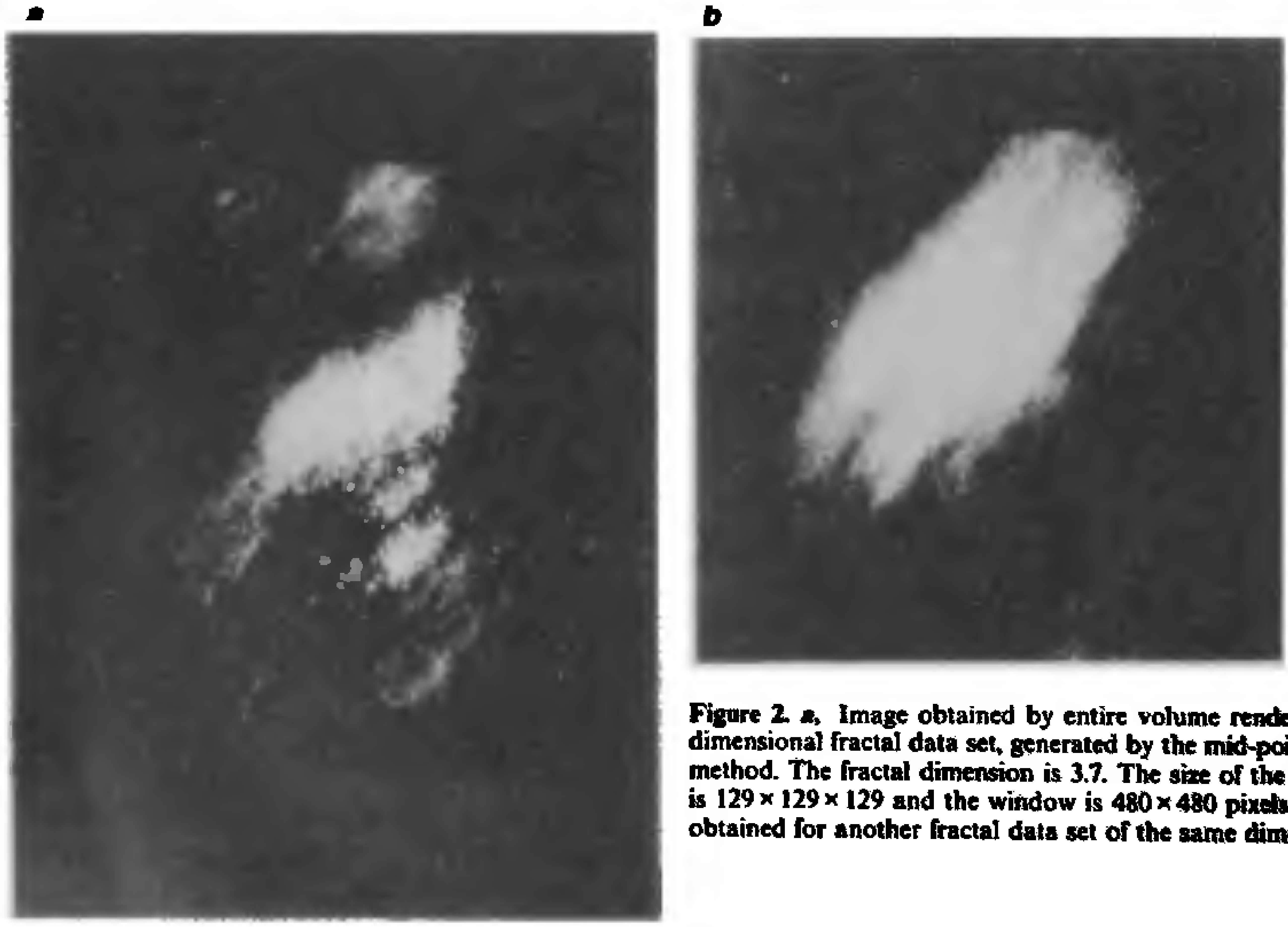


Figure 2. **a,** Image obtained by entire volume rendering of a three-dimensional fractal data set, generated by the mid-point displacement method. The fractal dimension is 3.7. The size of the volumetric grid is $129 \times 129 \times 129$ and the window is 480×480 pixels wide. **b,** Image obtained for another fractal data set of the same dimension and size.