Chapter 2

Visualization techniques for trivariate data

2.1. Existing methods and terminology

Several methodologies for visualizing trivariate data have been outlined in the first chapter. Contemporary computer graphics equipment is capable of performing millions of arithmetic operations per second, providing an excellent tool for rendering three-dimensional data sets in real time. All techniques presented here are designed to allow interaction with the user.

Imaging three-dimensional data is known as **volume visualization** or **volume rendering**. Algorithms based on ray-tracing have been widely used for volumetric data, e.g., in the medical field ([Fuchs et al. '89], [Kajiya & Herzen '84], [Levoy '88,'90], [Ney et al. '90], [Sabella '88], [Tiede et al. '90]). Considering the huge amount of data, ray-tracing is undoubtedly computing-intensive and hardly interactive. In [Foley et al. '90] nearly real time computing of ray-traced images is discussed. All these algorithms simulate the behavior of light rays (X-rays, for instance) passing through a finite volume containing the data to be visualized. "Objects" inside this volume usually appear more or less translucent. In Figure 2.1., CAT scan data (68 axial slices, each containing $64 \cdot 64$ integer density values in $\{0, 1, ..., 255\}$) of a human skull are rendered using the algorithm from [Levoy '88].

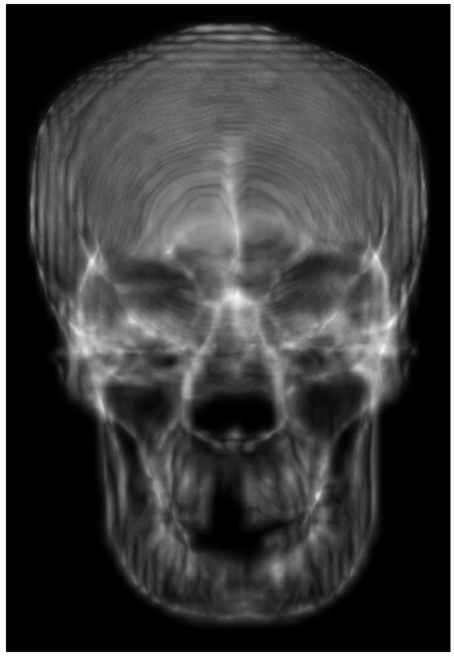

Fig.2.1. Skull rendered using Levoy's algorithm for CAT scan data.

Several rendering techniques are extensions of the bivariate to the trivariate case. In order to make use of lower-dimensional methods, a three-dimensional volume is intersected with a hyperplane, the trivariate function is restricted to this plane and rendered only for the single slice obtained (see examples in [Banchoff '90]). Numerous volume visualization techniques based on drawing graphs of bivariate functions and general overviews can be found in [Drebin et al. '88], [Nielson et al. '91], [Hamann '90a], and [Nielson & Hamann '90]. Algorithms based on approximating contours of trivariate functions are not discussed in this chapter. Generating linear contour approximations and modeling them is the content of the following chapters. A definition is given to understand the common nomenclature used in combination with visualizing and modeling multivariate data.

Definition 2.1. A **scattered trivariate data set** is the set

$$\{\ (\mathbf{x}_i{}^T, f_i) = (x_i, y_i, z_i, f_i) \mid \mathbf{x}_i \in I\!R^3,\ \ f_i \in I\!R,\ \ i = 0...n\ \}, \qquad (2.1.)$$

a **rectilinear trivariate data set** is the set

$$\{\ (\mathbf{x_i}{}^T, f_\mathbf{i}) = (x_i, y_j, z_k, f_{i,j,k}) \mid \mathbf{x_i} \in I\!R^3,\ \ f_\mathbf{i} \in I\!R,\ \ i = 0...n_x, j = 0...n_y, k = 0...n_z\ \}. \qquad (2.2.)$$

Often, these two data sets are simply referred to as scattered or rectilinear data. It is this kind of data that is visualized and modelled. Physical measurements are usually given as scattered data, whereas rectilinear data arise from evaluating some known trivariate function in a structured fashion. The geometry of an equidistantly-spaced rectilinear data set is directly reflected by the data's indices:

$$\begin{aligned} x_i &= x_0 + i\ \frac{x_{n_x} - x_0}{n_x}, && i = 0...n_x, \\ y_j &= y_0 + j\ \frac{y_{n_y} - y_0}{n_y}, && j = 0...n_y, \\ z_k &= z_0 + k\ \frac{z_{n_z} - z_0}{n_z}, && k = 0...n_z. \end{aligned}$$

The convex hull C of a rectilinear data set is therefore given by

$$C = [x_0, x_{n_x}] \times [y_0, y_{n_y}] \times [z_0, z_{n_z}] = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}].$$

In the case of scattered data, there is no underlying structure hidden in the geometry of the data points. For modeling purposes, the convex hull of scattered data points is split into a set of tetrahedra, usually, to obtain the Delaunay triangulation implied by the points (see [Preparata & Shamos '90]).

The different approaches used in computer-aided geometric design to model scattered or rectilinear data are not reviewed in detail in this dissertation. Generally, the modeling process can be divided into derivative estimation (typically, first and second order derivatives), construction of some trivariate function approximating the given function values and evaluation and visualization of that function. Normally, the function constructed to approximate either scattered or rectilinear data is evaluated on a rectilinear grid. This is the motive to primarily develop rendering techniques for rectilinear data sets.

Methods addressing the problem of estimating derivatives and approximating scattered and rectilinear data can be found in [Alfeld '89], [Barnhill & Little '84], [Boehm et al. '84], [Dahmen '89], [Davis '75], [Farin '83], [Farin '90], [Franke & Nielson '91], [Hoschek & Lasser '89], [Sederberg '85], [Stead '84], and [Worsey & Farin '87].

2.2. Domain subdivision and transparency techniques

The objective is to develop simple algorithms which can easily be implemented and can be used in a real-time, interactive fashion. All algorithms described here visualize rectilinear data sets. The domain-subdivision technique is based on extracting subvolumes $V_{r,s,t}$ from the convex hull C of the data points and coloring the surfaces of these subvolumes according to the function values on the surfaces of each such subvolume ($V_{r,s,t}$ is a box defined by its width, depth, and height).

Here, the idea is to allow for space between all subvolumes so that it is possible to "look inside" the data set when all subvolumes are rendered simultaneously. It is sufficient for this technique to specify resolution parameters q_x, q_y, and q_z for the three spatial directions and ratios α_x, α_y, and α_z determining the relative length of space between two consecutive subvolumes and an edge of a subvolume. Denoting the three edge lengths of a subvolume by $\triangle x$, $\triangle y$, and $\triangle z$, a subvolume $V_{r,s,t}$ is given by its left-front-lower corner point $(x_r, y_s, z_t)^T$ and its three edge lengths:

$$x_r = x_{min} + r\ (1+\alpha_x)\ \triangle x, \qquad \triangle x = \frac{x_{max} - x_{min}}{q_x + \alpha_x(q_x - 1)}, \qquad r = 0...(q_x - 1),$$

$$y_s = y_{min} + s\ (1+\alpha_y)\ \triangle y, \qquad \triangle y = \frac{y_{max} - y_{min}}{q_y + \alpha_y(q_y - 1)}, \qquad s = 0...(q_y - 1),$$

$$z_t = z_{min} + t\ (1+\alpha_z)\ \triangle z, \qquad \triangle z = \frac{z_{max} - z_{min}}{q_z + \alpha_z(q_z - 1)}, \qquad t = 0...(q_z - 1).$$

If the function to be rendered is known, it is evaluated at the eight corner points of each subvolume $V_{r,s,t}$; if it is not known, i.e., one is solely given a rectilinear data set, function values for $V_{r,s,t}$'s corner points are obtained by trilinear interpolation of those eight given function values in the rectilinear data set associated with a certain corner point. For example, if $x_r \in [x_i, x_{i+1}]$, $y_s \in [y_j, y_{j+1}]$, and $z_t \in [z_k, z_{k+1}]$, the function value at $(x_r, y_s, z_t)^T$ is the value of the trilinear interpolant to the set of known values $\{f_{i,j,k}, f_{i+1,j,k}, ..., f_{i+1,j+1,k+1}\}$ at $(x_r, y_s, z_t)^T$.

Now, minimal and maximal function values are determined among the corner function values of all subvolumes. They are denoted by f_{min} and f_{max}. A linear map is used to assign (integer) color values to each corner point of each subvolume. If c_{min} and c_{max} are the extreme (integer) color indices referring to a predefined color map, a (real) corner function value f is mapped to the color (-index) c, where

$$c = \left\lceil \frac{f_{max} - f}{f_{max} - f_{min}} \, c_{min} + \frac{f - f_{min}}{f_{max} - f_{min}} \, c_{max} \right\rceil .$$

Each face of the subvolumes is then Gouraud-shaded, i.e., a face's color is obtained by bilinear interpolation of the four colors (color-indices) associated with that face. If the function to be rendered is known and the extension of a subvolume $V_{r,s,t}$ (given by $\triangle x$, $\triangle y$, $\triangle z$) is relatively large compared to the extension of the convex hull C of the entire data set, it is appropriate to evaluate the function at more points than at the eight corner points of each subvolume. In principle, $(k+1)(l+1)$ function (and color) values are determined for points $\mathbf{x}_{i,j}$, $i = 0...k$, $j = 0...l$, arranged in a rectilinear fashion on a subvolume's face. Each two-dimensional grid cell on a single face, given by the four points $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, and $\mathbf{x}_{i+1,j+1}$, is then Gouraud-shaded itself.

Rendering all $q_x q_y q_z$ subvolumes and rotating them in real-time is possible. The impression of the three-dimensional structure can further be improved by drawing lines between the centroid of $V_{r,s,t}$ and the centroids of the six "neighbor" subvolumes $V_{r-1,s,t}$, $V_{r+1,s,t}$, ..., and $V_{r,s,t+1}$. Two examples for the domain-subdivision method using different resolution parameters are shown in Figures 2.2. and 2.3. The trivariate function visualized in both cases is

$$f\,(x,y,z) = 15\,\Big(e^{-0.005((x-10)^2+(y-10)^2+(z-10)^2)} + e^{-0.0025((x-15)^2+(y-20)^2+(z-20)^2)} + e^{-0.005((x-25)^2+(y-25)^2+(z-25)^2)}\Big),$$

$x, y, z \in [0, 39]$. Low values are mapped to green, high ones to white.

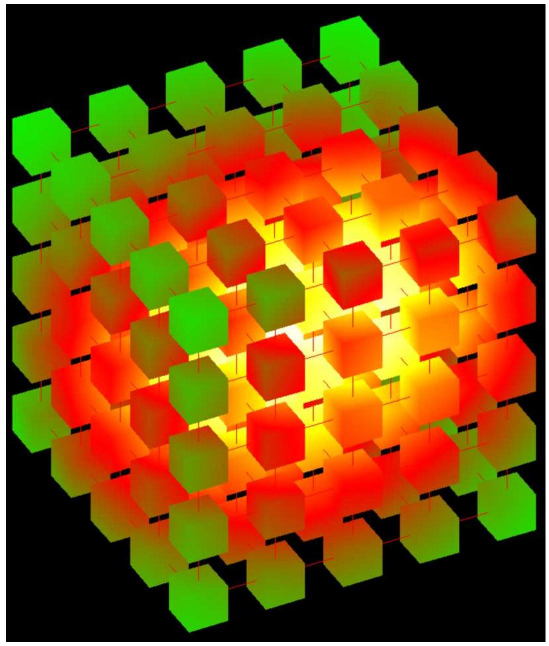

Fig. 2.2. Domain subdivision method for exponential function,
$q_x = q_y = q_z = 5, \quad \alpha_x = \alpha_y = \alpha_z = 1.$

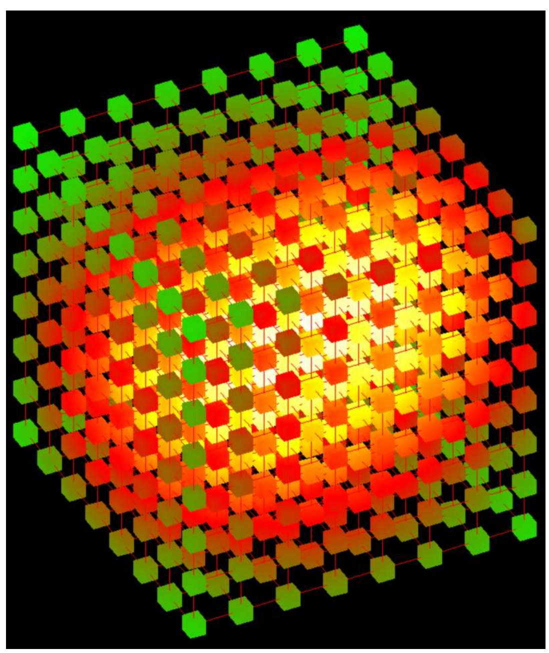

Fig. 2.3. Domain subdivision method for exponential function,
$q_x = q_y = q_z = 8, \quad \alpha_x = \alpha_y = \alpha_z = 2.$

The domain-subdivision technique can also be applied to volumes obtained by a map from three-dimensional $uvw-$space to three-dimensional $xyz-$space,

$$(x, y, z) = (\ x(u, v, w),\ y(u, v, w),\ z(u, v, w)\ ),$$
$$u \in [u_{min}, u_{max}],\ v \in [v_{min}, v_{max}],\ w \in [w_{min}, w_{max}].$$

Here, x, y, and z are continuous functions in all three variables. The function to be rendered is defined in $xyz-$space and the subdivision parameters actually refer to $uvw-$space. Using this approach, it is possible to visualize functions defined over more general volumes, such as volumes bounded by spheres, ellipsoids or paraboloids. Figure 2.4. shows the function

$$f\ (x, y, z) = \cos\ (\ 2\ \sqrt{x^2 + y^2 + z^2}\ ),$$

defined over a part of the unit ball.

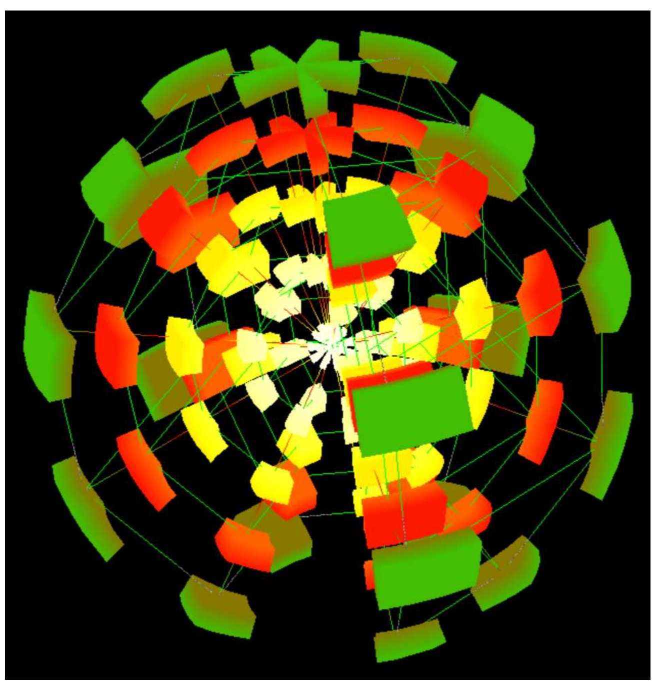

Fig. 2.4. Domain subdivision method for trigonometric function, unit ball, $q_u = q_v = q_w = 5, \quad \alpha_u = \alpha_v = \alpha_w = 2.$

The volume considered is the set of points $(x,y,z)^T$, where

$$(x,y,z) = (\ u\cos v,\ u\sin v\cos w,\ u\sin v\sin w\ ),\ u\in[0,1],\ \ v\in[0,\pi],\ \ w\in[0,\tfrac{3}{2}\pi].$$

Some graphics-workstations support a technique called *alpha blending.* It is a tool used for rendering transparent objects. Objects to be visualized in three-dimensional space are approximated by a set of planar polygons, where the term *polygon* is understood as the area bounded by a planar, closed, and non-self - intersecting piecewise linear curve. Alpha-blending requires an ordered set of polygons sorted with respect to their distance to the screen. Among a polygon's corner points one can use the one closest to the screen as sorting criterion.

A parameter, denoted by t, specifies the degree of transparency of polygons. If c_{old} is the color currently displayed at a screen position (i,j), and another polygon is to be rendered covering this particular area of the screen, the new color c_{new} for (i,j) is obtained by linear interpolation of the current color and c_{poly}, the color of the polygon at $(i,j) : c_{new} = (1-t)c_{poly} + tc_{old}$, $t\in[0,1]$. This implies that objects appear non-transparent, if t is 0.

In order to utilize alpha-blending, a trivariate function defined over $[x_0, x_{n_x}]\times[y_0, y_{n_y}]\times[z_0, z_{n_z}]$ is evaluated on a rectilinear grid yielding $(n_x+1)(n_y+1)(n_z+1)$ points and function values. Three sets of polygons are constructed: the first set consists of rectangles parallel to the $xy-$plane, the second of rectangles parallel to the $xz-$plane, and the third of rectangles parallel to the $yz-$plane. Each rectangle in the rectilinear grid in the first set is defined by the points $\mathbf{x}_{i,j,k}$, $\mathbf{x}_{i+1,j,k}$, $\mathbf{x}_{i,j+1,k}$, and $\mathbf{x}_{i+1,j+1,k}$, $i=0...(n_x-1), j=0...(n_y-1), k=0...n_z$. Similarly, one generates the rectangles of the other two sets. Again, the function values at these points determine the colors used for rendering.

The transparency parameter t and an orientation for the domain must be specified. Sorting all $3\, n_x n_y n_z + n_x n_y + n_x n_z + n_y n_z$ rectangles becomes an obstacle for using alpha-blending as an interactive visualization technique with high resolution parameters n_x, n_y, and n_z. Real-time performance can be achieved for varying t and fixed domain-orientation, but not conversely. The re-sorting of all rectangles takes too long in this case. In Figures 2.5. and 2.6., a gas-concentration is shown using different values for t (see [Long et al. '89]). Transparency definitely increases the visual understanding of a trivariate function by providing the possibility to perceive contours inside the function's domain.

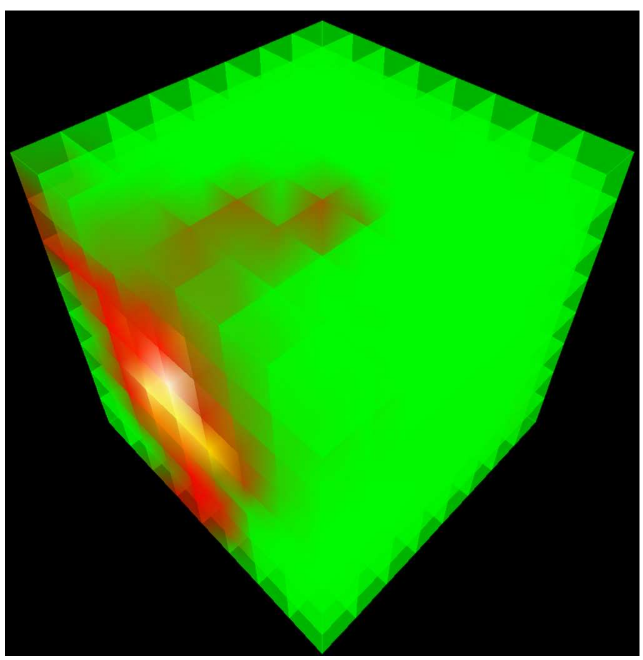

Fig. 2.5. Domain subdivision and transparency,
$n_x = n_y = n_z = 8, \quad t = 0.5.$

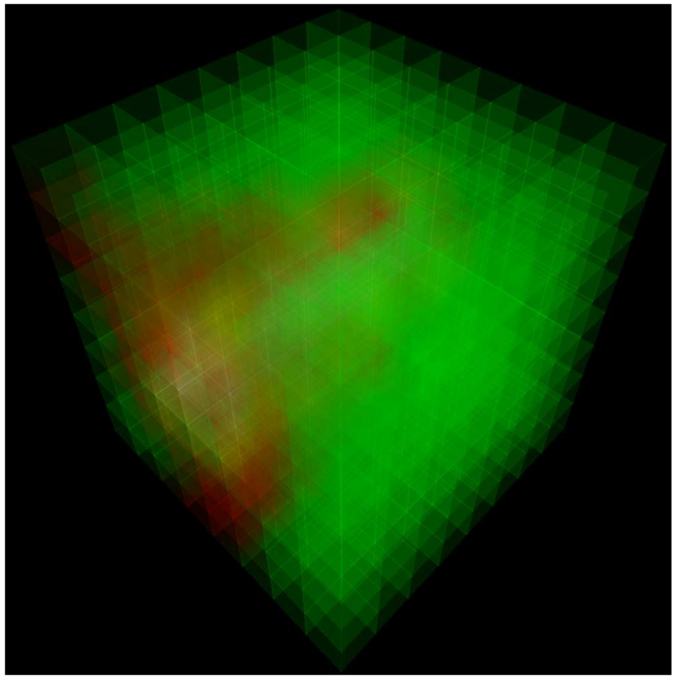

Fig. 2.6. Domain subdivision and transparency,
$n_x = n_y = n_z = 8, \quad t = 0.95.$

2.3. Slicing methods

Slicing methods are based on intersecting the domain D of a trivariate function with a hyperplane

$$P = \{ \ \mathbf{x} \mid (\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{n} = 0, \ \mathbf{x}, \mathbf{x}_0 \in I\!R^3, \ \mathbf{n} \ \text{ normal to } \ P \ \}$$

usually determining an area A bounded by a closed polygon. The trivariate function to be rendered is then restricted to A, evaluated and rendered for A only.

It is supposed that D is $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$, and the hyperplanes used have normal vectors $\mathbf{n}_1 = (1,0,0)^T$, $\mathbf{n}_2 = (0,1,0)^T$, and $\mathbf{n}_3 = (0,0,1)^T$. Three mutually perpendicular planes $P_i, i = 1,2,3$, are defined such that

$A_i = P_i \cap D \neq \emptyset$. The trivariate function is then evaluated on three $(k_i + 1)(l_i + 1)$ rectilinear grids, one grid per area A_i. These three areas can be visualized in two different ways.

The first possibility to visualize the function on all three areas A_i can be characterized as follows:

- assign colors (color indices referring to a predefined color map) to each point in each rectilinear grid,
- use Gouraud shading to color each rectangle given by the points $\mathbf{x}^i_{r,s}$, $\mathbf{x}^i_{r+1,s}$, $\mathbf{x}^i_{r,s+1}$, $\mathbf{x}^i_{r+1,s+1}$, $r = 0...(k_i - 1)$, $s = 0...(l_i - 1)$, $i = 1, 2, 3$, in each rectilinear grid and
- allow the user to interactively move any of the hyperplanes P_i in $x-$, $y-$, and $z-$direction, respectively.

Rotating the three hyperplanes in real-time and modifying the resolution parameters k_i and l_i can be accomplished interactively as well. In Figures 2.7. and 2.8., the same gas-concentration is visualized as in Figures 2.5. and 2.6. Depending on the data to be rendered, one might prefer a color map using multiple colors or a single color.

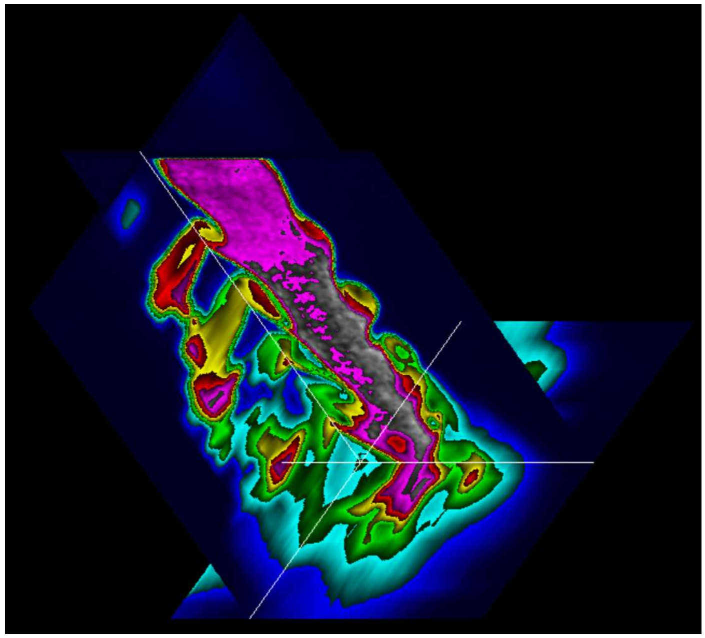

Fig. 2.7. Slicing method, coloring hyperplanes, multiple colors,
$k_1 = l_1 = 80,\ k_2 = l_2 = 130,\ k_3 = l_3 = 20.$

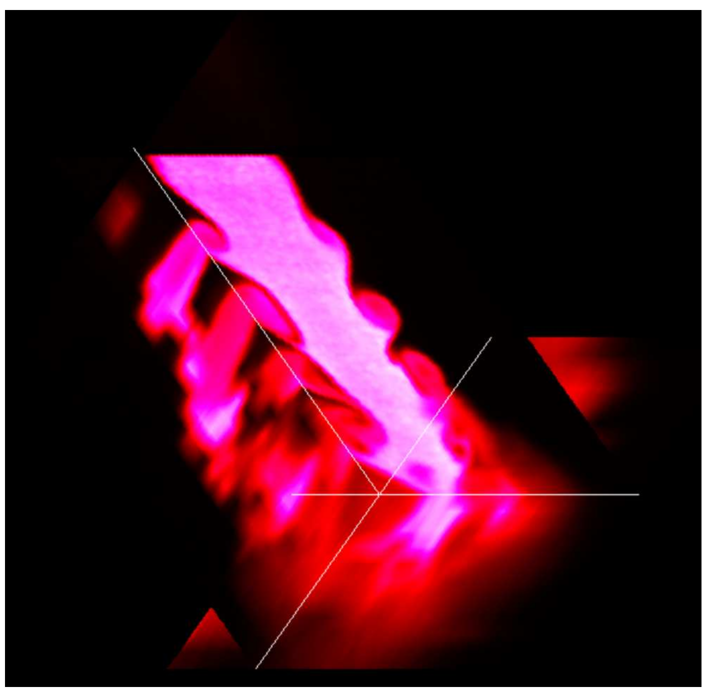

Fig. 2.8. Slicing method, coloring hyperplanes, single color,
$k_1 = l_1 = 80,\ k_2 = l_2 = 130,\ k_3 = l_3 = 20.$

The second possibility to visualize the function on all three areas A_i is similar to the standard procedure rendering a bivariate function. One generally evaluates a bivariate function over a specified (rectangular) domain. The result is a set of two-dimensional (domain-) points with associated function values. Points and function values combined are then interpreted as a set X of three-dimensional points yielding the graph of the bivariate function:

$$X \;=\; \left\{\; (x_i, y_j, f_{i,j})^T \;\mid i = 0...k,\; j = 0...l \;\right\}.$$

The points in X are usually mapped into $[0,1]^3$. The graph can either be a curve network of piecewise linear curves or a shaded surface. In the first case, curves are defined by the line segments

$$\overline{(x_i, y_j, f_{i,j})^T (x_{i+1}, y_j, f_{i+1,j})^T}, \quad i = 0...(k-1),\; j = 0...l \qquad \text{and}$$
$$\overline{(x_i, y_j, f_{i,j})^T (x_i, y_{j+1}, f_{i,j+1})^T}, \quad i = 0...k,\; j = 0...(l-1).$$

In the second case, a surface is defined by two sets, I_1 and I_2, of index triples, each triple referring to three points in X,

$$I_1 \;=\; \Big\{\; \big((i,j),(i+1,j),(i+1,j+1)\big) \mid i = 0...(k-1),\; j = 0...(l-1) \;\Big\} \qquad \text{and}$$
$$I_2 \;=\; \Big\{\; \big((i,j),(i+1,j+1),(i,j+1)\big) \mid i = 0...(k-1),\; j = 0...(l-1) \;\Big\}.$$

Each triangle determined by an index triple is shaded on the screen.

To utilize this rendering technique for a trivariate function, the function is restricted to the three areas A_i, $i = 1,2,3$, again, evaluated on a rectilinear grid for each A_i and finally visualized as a set of three graphs, each graph either a curve network or a shaded surface. It is convenient to choose $P_1 = \big\{\mathbf{x} | x = c_1 \in [x_{min}, x_{max}]\big\}$, $P_2 = \big\{\mathbf{x} | y = c_2 \in [y_{min}, y_{max}]\big\}$ and $P_3 = \big\{\mathbf{x} | z = c_3 \in [z_{min}, z_{max}]\big\}$.

The curve networks or shaded surfaces are then plotted over planes parallel to the $xy-$, $xz-$, and $yz-$plane, respectively. Special care must be taken to avoid intersections among the three graphs. To gain a better visual understanding of the position and orientation of the three areas A_i relative to each other and to the trivariate function's domain, the areas A_i are represented as single-colored rectangles in a box indicating the function's domain. Figure 2.9. is an example for this surface-based rendering technique using the same exponential function as in Figures 2.2. and 2.3.

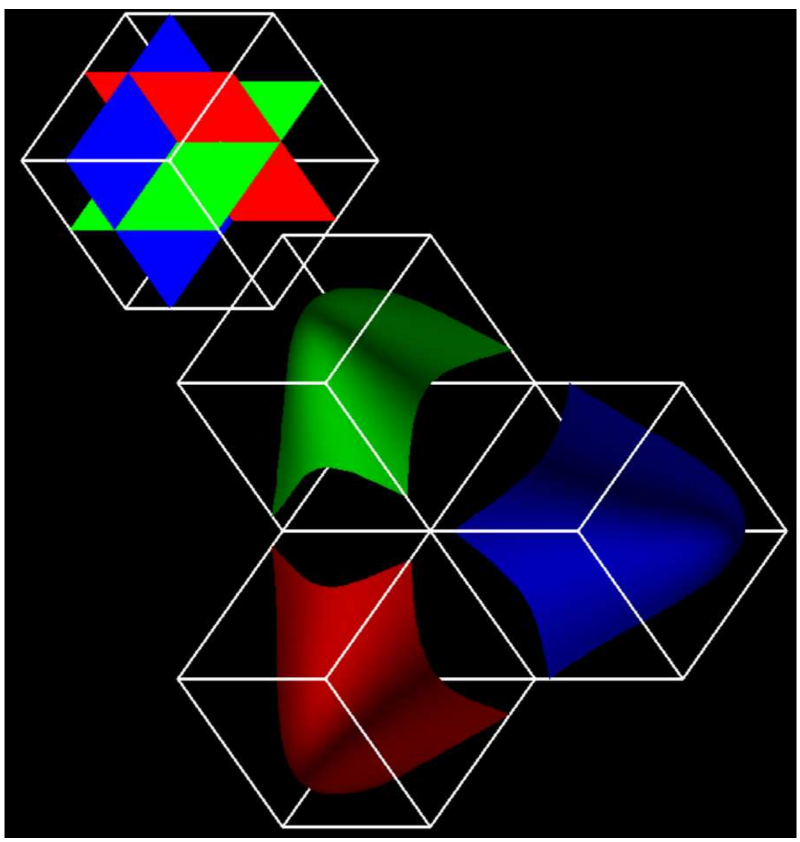

Fig. 2.9. Slicing method, bivariate surfaces, Gouraud-shaded, $k_1 = k_2 = k_3 \; = \; l_1 = l_2 = l_3 \; = \; 30.$