CWI Quarterly

Volume 7 (2) 1994, pp. 147 – 158

Visualization of Multi-Dimensional Scalar Functions Using HyperSlice

Robert van Liere

Centrum voor Wiskunde en Informatica

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

e-mail: robertl@cwi.nl

Jarke J. van Wijk

Netherlands Energy Research Foundation ECN

P.O. Box 1, 1755 ZG Petten, The Netherlands

e-mail: vanwijk@ecn.nl

HyperSlice is a new method for the visualization of scalar functions of many variables. With this method the multi-dimensional function is presented in a simple and easy-to-understand way in which all dimensions are treated identically. The central concept is the representation of a multi-dimensional function as a matrix of orthogonal two-dimensional slices. These two-dimensional slices lend themselves very well to interaction via direct manipulation, due to a one-to-one relation between screen space and data space. Several interaction techniques, for navigation, the location of maxima, and the use of user-defined paths, are provided.
In this article we present two extensions to the HyperSlice and show how the method can be used in practice.

1. Introduction

1.1. Problem

Scalar functions of several variables are often used in science and engineering. These functions can be denoted as $f(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ is a point in N-dimensional space. Scalar functions can be analytically defined, or can be the result of a simulation or measurements.

Visualization is an important tool for their analysis. Two types of use can be discerned. First, the function can be precomputed at a set of discrete points. The visualization then boils down to a visual inspection of a data set in which calculation of new function values is limited to interpolation of the values in the data set. Second, the function can be computed during the visualization. Here the user specifies what he is interested in, and a separate computation module generates the data. This approach is an example of *computational steering*: the simulation runs continuously, while the user simultaneously views the results and changes input parameters. This is highly efficient for multi-dimensional functions, because when the number of dimensions is large, the precomputation of data on a fine grid is prohibitively expensive in terms of processing power and memory requirements. However, computational steering obviously assumes that the function can be evaluated fast enough for interactive use.

The complexity of the visual presentation of multi-dimensional functions depends heavily on N. For $N = 1$ a simple graph suffices, for $N = 2$ two-dimensional color images or three-dimensional mountain plots are routinely used. The visualization of scalar functions of three variables is known as volume rendering, and is an important and active area of research. Many techniques have been proposed for their visualization [6].

The direct visualization of scalar functions of more than three variables is more complex, because the human mind is not able to imagine high-dimensional objects. With some effort, four-dimensional functions can be imagined as time-varying three-dimensional functions, but if $N > 4$ hardly anybody can produce mental images of such functions.

One solution to the presentation of functions with $N > 4$ is therefore to fix the value of a number of variables so that the number of free variables is lower than four, and then to use a standard visualization technique. In other words, a slice of the data is selected and visualized.

1.2. Previous work

Several researchers have proposed methods for the visual representation of multi-dimensional data and interaction on these representations. Although much progress has been made recently, most of the proposed solutions still do not seem to be satisfactory. All solutions compromise on the dimensionality, granularity and legibility of the representation. We make a crude classification of existing multi-dimensional data representations, by distinguishing between techniques using hierarchy, using iconic representations, and using scatterplot matrices.

The central idea of representation techniques involving hierarchy is to select a small number of dimensions and display these within a space of higher dimension. YOUNG, KENT and KUFELD [10] have developed the HyperSpace method for visualizing and interacting with multivariate data sets. First, this method uses interpolation to dynamically calculate and display a smoothly changing sequence of interpolations between two three-dimensional spaces. In effect, this is moving a three-dimensional object through a six-dimensional space. Second,

this method uses residualization to redefine two three dimensional spaces as a linear combination of six or more variables. Residualization allows the user to move the three dimensional space into any N-dimensional space, with $N > 6$. Other authors have suggested variants and enhancements to this hierarchical representation technique [5, 7].

In the Exvis project [8] icons with settable attributes are used to represent data. The original Exvis icon is a five limbed stick figure with controllable limb-angle, size, thickness and color. The authors show how this representation can be used to represent over twenty different data parameters. Presenting multi-dimensional data as a very large collection of icons produces a texture. Many other icons can be conceived to represent similar mappings. Other authors have also used icons and/or textures for representation [3, 1].

Scatterplot matrices [4] have been used extensively by the statistics community. Assuming an N-dimensional data set, a scatterplot matrix is an arrangement of $(N^2 - N)/2$ pairs of two-dimensional plots in which rows and columns of the matrix share common scales. Dependencies between variables can be obtained by scanning a row (or column) and visualizing how one variable is plotted against all others. Various interaction techniques have been proposed on the scatterplot matrix representation. For example, brushing is a simple but effective techniques that enables users to select groups of data points which are then highlighted in other projections. CLEVELAND [4] argues that scatterplot matrix representations augmented with highly interactive techniques provide more information than a simple sequence of scatterplot matrices themselves.

Both the hierarchical methods and the icon based methods provide sophisticated representations of continuous data. However, most of these representations are primary intended for a single static display, or a sequence of displays with limited interaction. The *Worlds within Worlds* concept of FEINER and BESHERS [5] is an important exception.

Scatterplot matrices provide simple representations of discrete data. An advantage is that all dimensions are treated identically, no more or less arbitrary decision is expected from the user how the data must be structured for presentation purposes. Furthermore, interaction techniques on this representation can be added relatively easily.

1.3. Overview

Our basic conjecture is that in scientific visualization, representation and interaction are equally important and that they are closely related. The visual representation should be such that the user can understand the behavior of the function, as well as easily interact on this representation.

The first choice to be made is on the dimensionality of the basic visual representation. The use of sophisticated three-dimensional techniques, possibly enhanced with animation and color, seems natural, because as many as possible dimensions are shown simultaneously. This solution is optimal if the function or data is three-dimensional. However, if more dimensions have to be visualized, only a selection can be shown, and hence navigation (e.g. modification of the

values of variables that are fixed for a single representation) becomes essential. Here we run into problems. First, although significant progress has been made, current techniques for volume rendering are too slow for direct manipulation. Second, such volume renderings are more difficult to interpret than simpler representation forms, and often tuning of the settings of thresholds, opacity mappings, etc. is required. Finally, three-dimensional interaction is not trivial.

We therefore use two-dimensional slices as the basic visual representation. The geometric coordinates denote two variables, a gray or color value denotes the value of the function. Rendering is fast, visual interpretation is easy, and interaction is direct, because of the one to one relation between the screen space and data space.

However, a single slice only shows a very limited subset of the multi-dimensional space. We therefore developed *HyperSlice*, a new method for the visualization of multi-dimensional functions. With this method the function is presented in a simple and easy to understand way, all dimensions are treated identically, and interaction via direct manipulation of the representation is easy and effective.

The format of this paper is: first, we review the underlying concepts of the HyperSlice representation and basic interaction methods with this representation. Second, we provide various examples of interactive visualization techniques that augment the direct interaction with higher level information. Finally, we present two applications.

A more elaborate description of HyperSlice can be found in [9]. This paper introduces two extensions. In particular, we present interaction techniques based on rotation and contouring.

2. HyperSlice

The central concept is the representation of a multi-dimensional function as a matrix of orthogonal two-dimensional slices. These two-dimensional slices lend themselves very well to interaction via direct manipulation, due to a one to one relation between screen space and data space. For example: users can translate and rotate through the data space by simply pointing the mouse and dragging the two-dimensional slices. Furthermore, higher level interaction tools are integrated into HyperSlice, resulting in a powerful environment for the analysis of multi-dimensional scalar functions.

In this section we introduce the definition of the current point and of a slice, then we use these definitions to discuss the HyperSlice representation. Finally, we show how interaction is expressed.

2.1. Definitions

We assume that the focus of the user is on a single N-dimensional point of interest, $\mathbf{c} = (c_1, c_2, ..., c_N)$, which is called the current point. The width of the focus is a set of scalar values w_i, with $i = 1, \ldots, N$. The range of values of interest for dimension i is the interval $R_i = [c_i - w_i/2, c_i + w_i/2]$.

A *two-dimensional slice* $S_{k,l}$, with $k \neq l$, is a visual representation of $f(\mathbf{x})$,

where $x_k \in R_k$ and $x_l \in R_l$ vary and the other x_i are equal to c_i. The horizontal axis of the slice is aligned with x_k, and the vertical axis with x_l.

A *one-dimensional graph* G_k is a graph of $f(\mathbf{x})$ where $x_k \in R_k$ and all other x_i are equal to c_i. In this case the horizontal axis is aligned with x_k, and the vertical axis is aligned with $f(\mathbf{x})$.

2.2. HyperSlice Representation

A HyperSlice representation is an NxN matrix of panels $< i, j >$, with $1 \leq i, j \leq N$. Panels on the diagonal of the matrix contain graphs G_i, panels at off-diagonal positions contain slices $S_{i,j}$. The ranges R_i are enumerated along the horizontal and vertical axes of each of the panels. As a result, the current point will always be centered in the middle of the panel.

For slices $S_{i,j}$, the function values are shown as a rectangular grid of cells. A linear transfer function is used to map the function values to grey values. For graphs G_i, the area under the graph is similarly colored according to the local value of the graph.

As an example consider Figure 1, which shows the concept for $N = 3$. Displayed on the left is the current point as a small sphere, whereas the matrix on the right displays the corresponding HyperSlice. The three principle axes are labeled with (X, Y, Z). Pseudo coloring is used to annotate the three principle axes (red, yellow, blue) and slice (cyan, orange, green). The generalization to higher N is straightforward.

2.3. Translation and scaling

The HyperSlice representation displays the multi-dimensional function at a user-defined focus. Probably the most important aspect of user interaction is therefore the manipulation of the current point $\mathbf{c}$, allowing the user to navigate the multi-dimensional space in search for interesting features of the function. The user can point at a panel, press a mouse-button, and drag the visual representation. Mouse buttons are bound to the basic navigation operations; the left mouse button will translate $\mathbf{c}$, the middle button will rotate $\mathbf{c}$, and the right button will scale.

When the user drags a slice $S_{k,l}$ over a displacement $[d_k, d_l]$, then the current point $\mathbf{c}$ is changed as follows:

$$\begin{aligned} c_k &\leftarrow c_k - d_k \ , \\ c_l &\leftarrow c_l - d_l \ . \end{aligned}$$

The visual effect is shown in Figure 2. Here the slice $S_{4,2}$ is dragged. As a result, slices in the same column move horizontally over a displacement d_k, whereas the slices in the same row move vertically over a displacement d_l.

If the graph G_i is dragged, the single variable x_i is changed. The effect is similar to that as described for slices. Thus, each panel serves not only as a visual representation, but also as one- or two-dimensional sliders for the current value of variables x_i.

Scaling is realized by changing the widths w_i of each range R_i. The user can change an individual w_i in the corresponding graph G_i, or can simultaneously change w_i and w_j in a slice $S_{i,j}$. In the current implementation all w_i can also be scaled simultaneously with the same factor with zoom-in and zoom-out buttons in a control panel.

2.4. Rotation

Rotation is a new feature added to HyperSlice. As with translation and scaling the user has direct control over the rotation by simply using the mouse.

A rotation is defined as a rotation around the current point, $\mathbf{c}$. Rotation of an angle α in the slice $S_{i,j}$ is denoted by the rotation matrix $\mathbf{R}_{i,j}(\alpha)$ whose elements are:

$$r_{k,k} = 1, \text{ except } r_{i,i} = r_{j,j} = \cos \alpha,$$
$$r_{k,l} = 0\ ,\ k \neq l, \text{ except } r_{i,j} = -r_{j,i} = -\sin \alpha.$$

For example: the rotation matrix for a rotation over an angle α in the slice $S_{2,4}$ in five-dimensional space is:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix}$$

Every multi-dimensional rotation can be written as the product of N ($N-1$) / 2 two-dimensional plane rotation matrices. Hence, by rotating the slices, any orientation may be obtained. For example, in four-dimensional space

$$\mathbf{R} = \mathbf{R}_{1,2}(\alpha_1)\mathbf{R}_{1,3}(\alpha_2)\mathbf{R}_{1,4}(\alpha_3)\mathbf{R}_{2,3}(\alpha_4)\mathbf{R}_{2,4}(\alpha_5)\mathbf{R}_{3,4}(\alpha_6)$$

The current rotation is stored in a global rotation matrix. Rotations are accumulated: individual rotation matrices are multiplied with the global rotation matrix, resulting in the new, updated, matrix. The multi-dimensional point $\mathbf{x}$ is transformed into screen space by multiplying it with the rotation matrix.

When rotation is applied the simple one-to-one relation between the axes in data space and the axes in screen space is lost. We, therefore, provide additional annotation by projecting all data space axes in every slice. During rotation these axes provide orientation feedback to the user. Since translations are defined along the screen axes, the user still can drag the visual representation after a rotation has occurred.

Rotation is typically used to sweep through the region around the current point. This can be realized by placing the mouse in one panel and rotating over an angle of 360 degrees.

3. Navigation in HyperSlice

In the previous section the emphasis was on direct, but simple, user interaction on the current point. In addition to this very direct type of interaction,

the value of $\mathbf{c}$ can also be adjusted automatically, according to some specific criterion. In this section we provide various examples of interactive visualization techniques that augment the direct interaction with some higher level information.

We show how edit paths, gradient paths and contouring can be used to guide the user during traversal through data space.

3.1. Paths

It is easy to get lost during the exploration of hyperspace. To prevent this, the user can define paths by setting out a number of interesting points and using these paths to quickly traverse through space. A path P is a sequence of marks $\mathbf{m}_i$, where a mark is a point in N-dimensional space. The projections of the marks are shown as crosses, the path is shown as a sequence of projected line segments $\mathbf{m}_i\mathbf{m}_{i+1}$. A set of standard editing operations is available to the user. A new path can be created, marks can deleted, moved or inserted into the path, etc. Using edit paths in this way allows HyperSlice to be used as a multi-dimensional drawing tool.

In addition to the path specification, the user has a control panel available to animate $\mathbf{c}$ using this path. The user can request to move $\mathbf{c}$ along the path, stepwise or continuously, in forward or backward direction. Whenever $\mathbf{c}$ changes, all panels are recomputed and redisplayed.

Path specification and animation has proven to be very powerful in practice. For example, the possibility to mark interesting points and to jump back and forth allows fast comparisons. The use of the path to animate $\mathbf{c}$ enables the viewer to observe trends in directions oblique to the principal axes. Another interpretation of the path is as a constraint on $\mathbf{c}$ to a user defined, one-dimensional subspace of the multi-dimensional space.

3.2. Gradient paths

When navigating through the multi-dimensional space, a visualization system can guide the user towards interesting regions, which may or may not be easy to find with only direct interaction. We discuss one such option: a tool to simplify the location of maxima. When enabled, a gradient path from $\mathbf{c}$ to the nearest maximum is computed. Each step in this gradient path is in the direction of the steepest ascent. Projections of this gradient path are shown on all slices. The gradient path is recomputed each time the current point $\mathbf{c}$ is changed, allowing the user to detect for instance saddle areas: regions in which the gradient path will jump wildly from one local maximum to another. In addition to the display of the gradient path, the user can request to animate $\mathbf{c}$ along the gradient path to the maximum.

3.3. Contouring

An iso-surface is the locus of all points $\mathbf{x}$ for which $f(\mathbf{x}) = C$ with C a chosen value. Cross sections of these surfaces with the slices are displayed as closed curves, i.e. contours. The user can select and change the value C in a

control panel. Contours will be recomputed each time the current point $\mathbf{c}$ is changed. This allows the user to determine interesting areas: regions which are segmented by the contour value.

In addition to a fixed contour value, we provide an option called dynamic contouring. Dynamic contouring sets C to $f(\mathbf{c})$. The effect is that all points in the image that have the same function value as the current point are highlighted.

4. Applications

4.1. Potential function

As a first example, we consider a synthetic application: a potential function that results from a set of multi-dimensional point objects. Each object has a position $\mathbf{p}_i$. The potential function $f_i(\mathbf{x})$ of a single object is defined by:

$$f_i(\mathbf{x}) = 1/(1 + |\mathbf{x} - \mathbf{p}_i|^2) \ .$$

The total potential $f(\mathbf{x})$ is their sum:

$$f(\mathbf{x}) = \sum_i f_i(\mathbf{x}) \ .$$

This function can be used for any number of dimensions.

Figure 3 shows an image of the four-dimensional potential function as it appears on the screen. For all data related information (graphs and slices) a Gouraud shaded, grey-scale coloring scheme is used. The current point is depicted as a small red box in the center of a panels. A gradient path (drawn in green) and a user defined path (yellow) are shown, as well as some simple annotation of the axes. Three point objects were defined in this data set. Their positions were located with the gradient path and marked. The user defined path connects the three objects.

4.2. Eccentricity

As a second example, we consider the shape of the orbit of a point mass in a gravity field of a stationary body. For example: the orbit of a satellite around the earth. The orbit of the point mass can be described by the following expression [2] :

$$\epsilon^2 \ - \ 1 \ = \ \frac{m^2 k^2}{c^2} \left(v^2 \ - \ \frac{2c}{mr} \right)$$

in which ϵ is the eccentricity, $v(t)$ and $r(t)$ is the tangential velocity and distance (in polar coordinates) of the point mass to the stationary body; m is mass of the stationary body and k and c are two constants, k is an integration constant, and c is related to the gravitation constant of the gravity field. The eccentricity is a mathematical constant that determines the shape of the orbit of the point mass. When $\epsilon < 1$ the orbit will be an ellipse (for $\epsilon = 0$ it will be a circle);

for $\epsilon = 1$ the orbit is a parabola; for $\epsilon > 1$ the orbit is a hyperbola. The orbit of the point mass is completely determined by the initial conditions; i.e. given the velocity and position at time $t = 0$ the orbit will be either an ellipse, a parabola or a hyperbola.

The previous expression was reformulated as $\epsilon\ (x,\ y,\ v_x,\ \gamma)$, in which x, y is the initial position of the point mass (in cartesian coordinates), v_x is the x component of its velocity ($v_y = 0$) and γ is a constant which relates to the gravitation constant. Figure 4 shows the HyperSlice representation. The panels are labeled from left to right as x, y, v and γ respectively. The contour $\epsilon = 1$ (drawn in cyan) shows all those points in which the orbit of the point mass is a parabola. A grey scale transfer function associates values to colors, with white being associated to $\epsilon = 0$ and black to $\epsilon = 1.3$.

Gradient paths can be switched on to determine those regions in which ϵ is at its local minimum. In these regions the orbit of the point mass will be a circle. Contouring can be used to segment the various images: the lighter regions in the images the orbit will be an ellipse, and in the darker regions the point mass will be a hyperbola.

All these operations can be done in interactive time on a modern workstation. Frame rates of over ten frames a second can easily be realized. The resolution of the image in Figure 4 is 32 cells for each axis, resulting in 6272 function calculations per frame.

5. Conclusion and future work

We have presented two applications of HyperSlice. This visualization method represents an multi-dimensional space as a matrix of orthogonal two-dimensional slices. This representation is simple and easy to understand, symmetric for all variables, and allows for easy interaction via direct manipulation. Other interaction techniques, such as edit paths, gradient paths and contours, are provided for a more automated traversal of the data space.

The HyperSlice technique has been developed within the CWI Scientific Visualization Project "Computational Steering". The goal of this project is to study methods and develop tools that aid end users in steering of large ongoing numerical simulations. Central themes of the steering project are navigation, monitoring, and tracking. From the development of HyperSlice we have learned that a close relation between visual presentation and interaction is very effective in computational steering. With this insight we are currently developing a more general environment which offers end users a very flexible set of tools to steer simulations and navigate through parameter spaces.

References

1. J. Beddow (1990). Shape coding of multidimensional data on a microcomputer display. In *Proceedings Visualization '90*, pp. 238–246. IEEE Computer Society Press, Los Alamitos, CA.
2. A.N. Borghouts (1978). *Inleiding in de Mechanica*. Delftsche Uitgevers Maatschappij, B.V., 3e druk edition.

3. H. CHERNOFF (1973). The use of faces to represent points in k-dimensional space graphically. *Jour. Amer. Stat. Assoc.*, pp. 361–368.
4. W.S. CLEVELAND (1985). *The Elements of Graphing Data.* Wadsworth Inc.
5. S. FEINER and C. BESHERS (1992). World within worlds. In *Proceedings Visualization '92*, pp. 283–290. IEEE Computer Society Press, Los Alamitos, CA.
6. P. HANRAHAN, J. KAJIYA, W. KREUGER, P. SCHROEDER and J. WILHELMS (1991). State of the art in volume visualization. In *SIGGRAPH '91 Tutorial Course Notes, volume 8.*
7. T. MIHALISIN, J. SCHWEGLER and J. TIMLIN (1992). Hierarchical multivariate visualization. In *Proceedings Interface '92.*
8. S. SMITH, G. GRINSTEIN and R.D. BERGERON (1991). Interactive data exploration with a supercomputer. In *Proceedings Visualization '91*, pp. 248–254. IEEE Computer Society Press, Los Alamitos, CA.
9. J.J. VAN WIJK and R. VAN LIERE (1993). Hyperslice : Visualization of scalar functions of many variables. In G.M. NIELSON and D. BERGERON, editors, *Proceedings Visualization '93*, pp. 119–125, San Jose.
10. F.W. YOUNG, D.P. KENT and W.F. KUHFELD (1988). *Dynamic Graphics for Exploring Multvariate Data*, pp. 391–424. Wadsworth Inc.

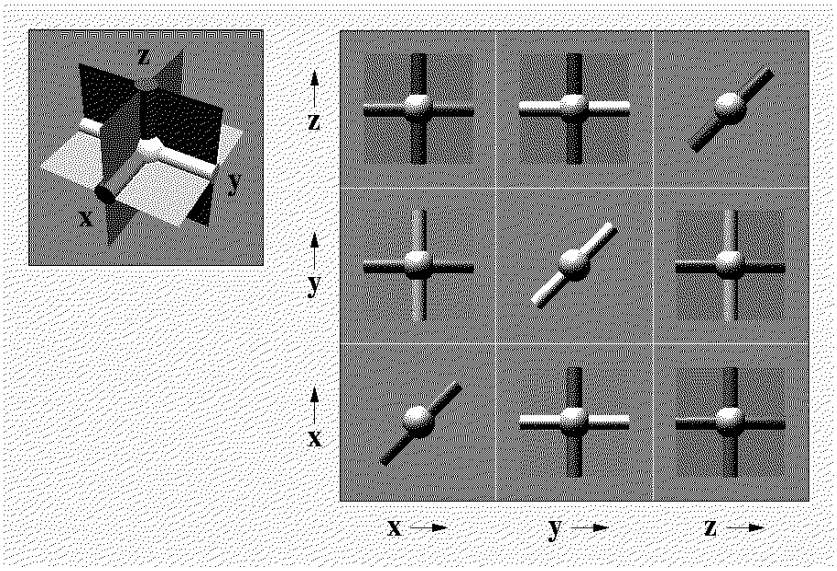

FIGURE 1. The concept of HyperSlice for $N = 3$

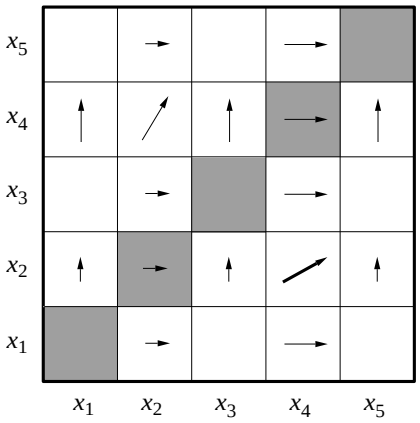

FIGURE 2. Effect of dragging a slice

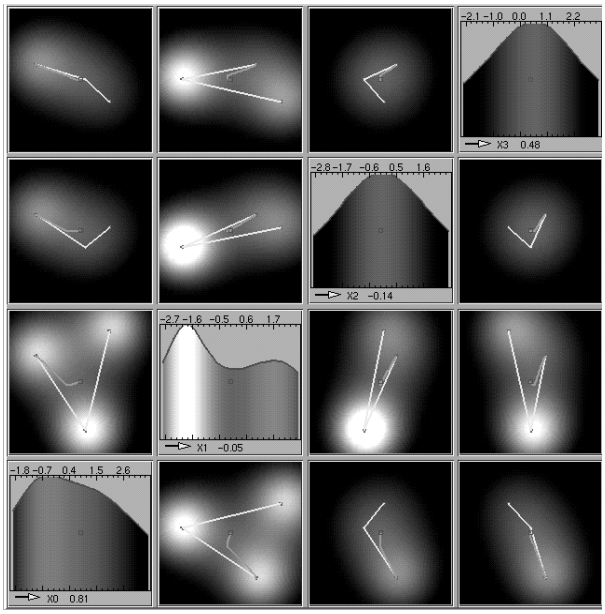

FIGURE 3. HyperSlice applied to four-dimensional potential function

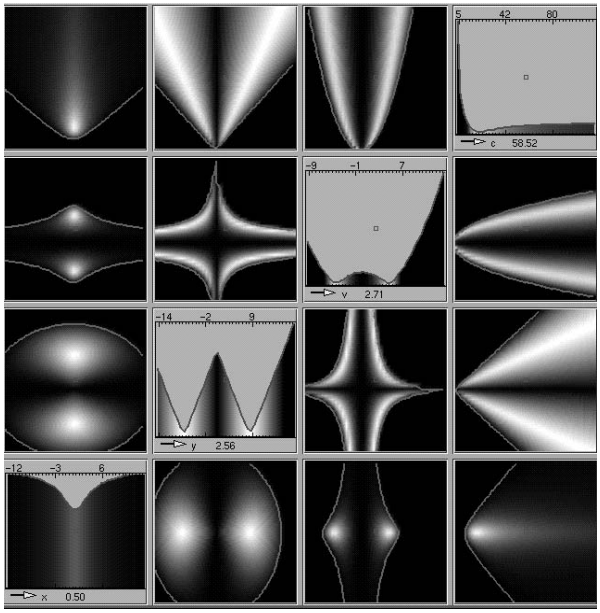

FIGURE 4. The HyperSlice of $\epsilon(x, y, v, \gamma)$.