HyperSlice

Visualization of scalar functions of many variables

Jarke J. van Wijk

Netherlands Energy Research Foundation ECN
P.O. Box 1, 1755 ZG Petten
The Netherlands

Robert van Liere

Centrum voor Wiskunde en Informatica CWI
P.O. Box 4079, 1009 AB Amsterdam
The Netherlands

Abstract

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 variable space. Several interaction techniques, for navigation, the location of maxima, and the use of user-defined paths, are presented.

1 Introduction

1.1 Problem

Scalar functions of several variables are often used in science and engineering. These functions can be denoted as $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_{n-1}, x_N)$, where $\mathbf{x}$ is a point in N-dimensional space, and x_i is a variable of the i-th dimension. 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 parameters. It 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, it obviously requires that the function can be evaluated fast enough for interactive use.

The complexity of the representation 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 [1].

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 representation 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 functions 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. A crude classification for multi-dimensional function representations is:

- One popular class of representation techniques involves hierarchy : select a small number of dimensions and display these within a space of higher dimension. Young, Kent and Kufeld [2] 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 [3, 4].

- In the Exvis project [5] 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 [6, 7].

- Scatterplots matrices [8] 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 row 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 argues that scatterplots 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 [3] is an important exception.

Scatterplot matrices provide simple representations of discrete data. An advantage is that the different 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.

We argue that a simple representation augmented with fast interaction tools, based on direct manipulation, can provide additional insight in continuous scalar functions of many variables.

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 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. Although significant progress has been made, current techniques for volume rendering are too slow for direct manipulation. Also, 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. Furthermore, interaction in three-dimensional space 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. Here rendering is fast, visual interpretation is easy, and interaction is direct, because of the one to one relation between the screen space and variable 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.

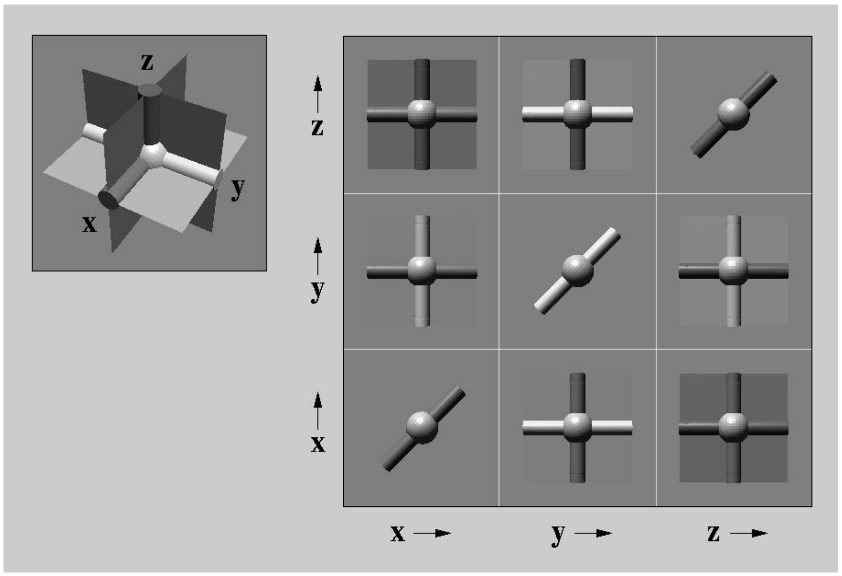

Figure 1: The concept of HyperSlice for $N = 3$

The central concept, presented in section 2, is the representation of a multi-dimensional function as a matrix of orthogonal two-dimensional slices. This representation lends itself very well to a variety of interaction techniques. In section 3 navigation, the location of extrema, and the use of user defined paths is summarized. Next implementation issues (section 4) are discussed and an application (section 5) is presented. Finally, in section 6 the results are discussed and suggestions for further research are done.

2 Representation

In this section we present the HyperSlice representation, after an introduction of some basic notions and definitions.

We assume that the focus of the user is on a single N-dimensional point of interest which we call the current point, and denote as $\mathbf{c} = (c_1, c_2, ..., c_N)$. 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 and x_l vary and the other x_i are equal to c_i, or:

$$\begin{aligned} x_k &\in R_k \,, \\ x_l &\in R_l \,, \\ x_i &= c_i, i \neq k \text{ and } i \neq l \,. \end{aligned}$$

The horizontal axis of the slice is aligned with x_k, and the vertical axis with x_l. The function values are shown as a rectangular grid of filled cells. We used grey values for the cells, which are derived from the function value via a linear transfer function. The current point in the center of the slice is marked with a square, and other annotation of the axes can be displayed at request. 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 again with x_k, while the vertical axis is aligned with $f(\mathbf{x})$.

The next question is which slices are to be displayed. We require that all dimensions are treated identically, so we display the value of the function for all pairs of variables. This leads us to the HyperSlice representation. A HyperSlice is a matrix of panels i, j with $1 \leq i, j \leq N$. Ranges R_i are enumerated along the horizontal and vertical axes. Panels on the diagonal contain graphs G_i, panels at off-diagonal positions contain slices $S_{i,j}$. As an example, figure 1 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. Colors are used to annotate the three principle axes (red, yellow, blue) and slices (cyan, orange, green). The generalization of the HyperSlice representation to higher N is straightforward.

Several relations exist between the panels:

- all off-diagonal slices $S_{i,j}$ are the same as slices $S_{j,i}$, rotated over 90 degrees;
- the function value $f(\mathbf{c})$ displayed in the center of a panel is the same for all panels;

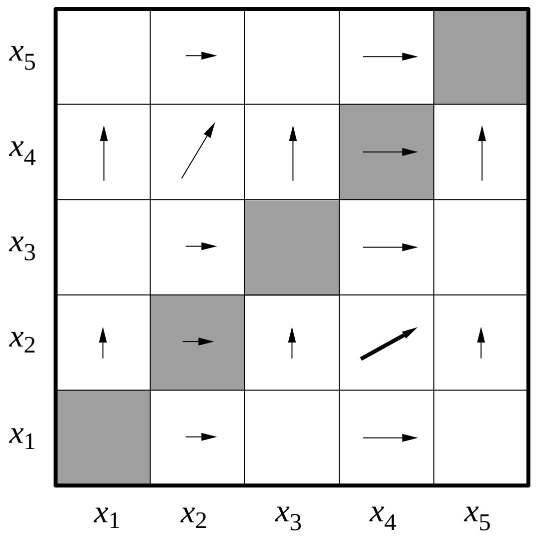

Figure 2: Effect of dragging a slice

- the values along a horizontal line through the center of the panel are the same for all panels in the same column, and also the values along a vertical line are the same for all panels in the same row (see fig. 1).

This HyperSlice representation allows the viewer to observe the sensitivity of f to changes in one and two dimensions. It is difficult, if not impossible, to reconstruct a complete, multi-dimensional mental image from the separate graphical representations. However, this representation does enable the user to view the multi-dimensional space around a point in a simple and intuitive way. The user can locate features such as extrema and hyperplanes. Because all dimensions are presented simultaneously and in various combinations, the chance that important relations are overlooked is small. Another interesting property is that for $N = 1$ the HyperSlice reduces to the standard representation: a single graph.

The main strength of the HyperSlice representation is that it lends itself very well to interaction via direct manipulation, which is the subject of the next section.

3 Interaction

3.1 Navigation

The HyperSlice representation shows f only around the current point **c**. Probably the most important aspect of user interaction is therefore the change of **c**. By changing **c** the user steers through multi-dimensional space in search for interesting features of the function, where the visual representation supports his navigation. A direct and simple solution is feasible with the HyperSlice concept. The user can point at a panel, press a mouse-button, and drag the visual representation. If the user drags a slice $S_{k,l}$ over a displacement $[d_k, d_l]$, then the current point **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. 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. Furthermore, for all slices $S_{i,j}$ other than $S_{k,l}$, one or two of the modified dimensions of the current point are not represented by a horizontal or vertical axis. One could say that these dimensions are perpendicular to these slices. A change in such a dimension does affect the slice shown: the slices move perpendicular to the image plane.

If the graph G_k 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.

In practice this mechanism is used in various ways:

- If in one of the panels an interesting spot is detected (e.g. an optimum) the user can drag this spot to the center of the panel;

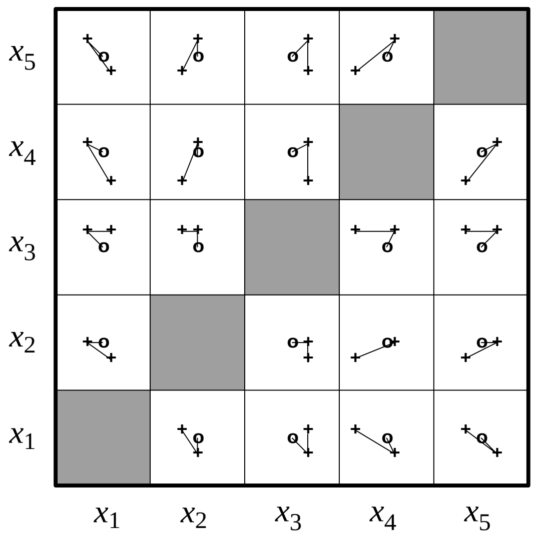

Figure 3: User defined path

- The multi-dimensional space can be scanned along a single axis by dragging the graph in a diagonal panel;
- It is very instructive to look at a slice $S_{i,j}$ while dragging the slice $S_{k,l}$, with i, j, k, and l all different. The effect is that the slice $S_{i,j}$ is moved along the dimensions k and l, which are both perpendicular to this slice.

The widths w_i of each range R_i can also be changed. In the current implementation all w_i can be scaled simultaneously with the same factor with zoom-in and zoom-out buttons at a control panel. Also, they can be adjusted per dimension.

3.2 Optimization

In the previous section the emphasis was on user controlled changes of the current point. However, the value of **c** can also be adjusted automatically, according to some criterion. Currently, one such option has been implemented: a tool to simplify the location of maxima. When enabled, a gradient path from **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 **c** is changed. This allows the user to detect for instance saddle areas: here 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 **c** along the gradient path to the maximum.

3.3 Paths

It is easy to get lost during the exploration of hyperspace. To prevent this, the user can define paths and mark interesting points. 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 $\mathbf{m}_i\mathbf{m}_{i+1}$. Figure 3 shows a path with three points. The current point is indicated with a circle.

A set of standard editing operations is available to the user. A new path can be created, the current point can be marked and added to the path. Further, each mark can be selected by point and click. The user can set **c** to this selected mark, delete the mark, move the mark, or insert a new mark after this mark. The visualization tool can thus be used as a multi-dimensional drawing tool.

In addition to the path specification, the user has a control panel available to animate **c** using this path. The user can request to move **c** along the path, stepwise or continuously, in forward or backward direction. Again, if **c** changes, all panels are recomputed and redisplayed. The gradient path can also be displayed along with the animation.

Path specification and animation has proved to be very powerful in practice. The metaphor of a drawing tool is easy to learn and understand. The possibility to mark interesting points and to jump back and forth allows fast comparisons. The use of the path to animate **c** enables the

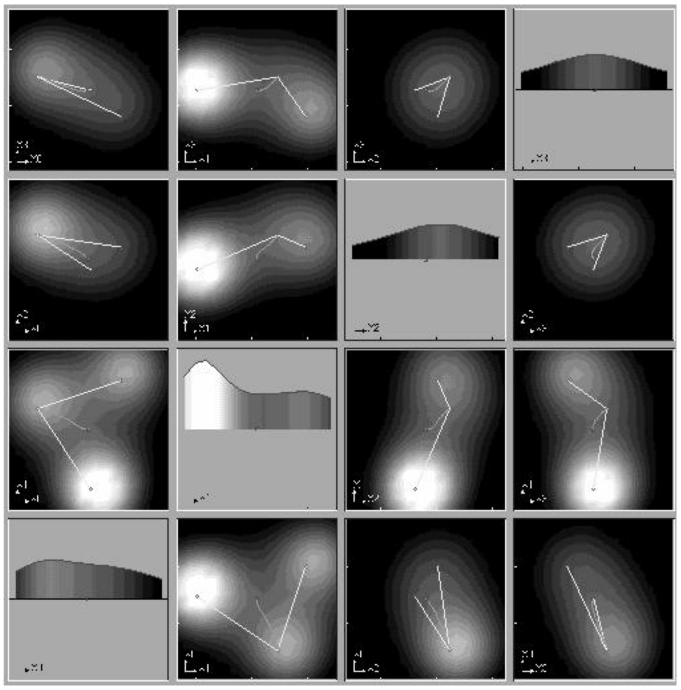
Figure 4: HyperSlice applied to four-dimensional potential function

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.

4 Implementation

For the display of a slice each dimension i is discretized into a number of samples M_i, resulting in a evenly spaced grid of coordinates.

The computational requirements of HyperSlice depend on the number of dimensions N and the sampling rates M_i. If all M_i are equal, the number of function evaluations for the slices is equal to $M^2(N^2 - N)/2$. Some values for the number of evaluations are given in Table 1.

N	M	evaluations
5	10	1000
5	25	6250
10	10	4500
10	25	28125

Table 1: Number of evaluations $f(\mathbf{x})$ for several values of N and M

A number of techniques can be used to improve performance. Various progressive and adaptive refinement techniques can be used. If the user does not interact with the representation, the computing process can continue to calculate the slices at higher sampling rates. During the dragging of a slice, the resolution of other slices can be lowered. Further, the presentation lends itself well to parallel processing, because each slice is independent. However, if the computation of the function $f(\mathbf{x})$ is not too demanding, a high level of interaction up to smooth animations can easily be achieved on a serial processor.

HyperSlice is written in C++ using the Forms library of user interface objects and SGI's graphics library. It runs on all SGI platforms.

5 Application

To test the method we applied it to a simple synthetic application: a potential function that results from a set of multi-dimensional point objects. Each object has a position $\mathbf{p}_i$ and a weight w_i. The potential function $f_i(\mathbf{x})$ of a single object is defined by:

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

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

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

This function can be used for any number of dimensions, which makes it highly suitable for test purposes. Figure 4 shows an image as it appears on the screen for four dimensions. For all data related information (graphs and slices) a Gouraud shaded, grey-scale coloring scheme is used. The current value is depicted as a small red box in the center of a panels. A gradient path (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 thus connects the three objects.

Table 2 summarizes the performance achieved with this synthetic application. The first column provides various sampling rates, while the second and third column provide the number of frames per second on a SGI Indigo R4K with Elan graphics and a SGI 4D/310 with VGX graphics. The first workstation has a fast processor and relatively slow graphics, the second a slow processor and fast graphics. At both workstations animation rates can be achieved, provided that the sampling rate is not chosen too high. The table shows that CPU processor performance dominates rendering performance, even for this relatively simple application.

M	R4K+Elan	R3K+VGX
10	9.1	8.5
18	5.7	4.0
26	4.8	2.2
42	1.8	0.9
58	1.1	0.5

Table 2: Frame per second for different sample rates and architectures.

In the future we will apply HyperSlice to less synthetic problems. The most challenging is the problem of parameter estimation in chemical reactions [9]. These parameters are e.g. unknown reaction constants in the kinetic equations. Estimates of these parameter values are input to a model of the reaction. The results of the model are judged, and compared with experimental data. These type of problems are called “inverse problems” and we expect HyperSlice to be a useful tool for steering the numerical solver towards areas of interest of the multi-dimensional parameter space.

6 Summary

A new method is presented for the visualization of scalar functions of many variables. The function is represented as a HyperSlice: a matrix of orthogonal two-dimensional slices. This representation is simple and easy to understand, and symmetric for all variables. The use of two-dimensional slices allows for fast rendering, and, most important, easy interaction via direct manipulation, due to the one to one relation between screen space and variable space.

To support the user in the search for interesting features, several interaction techniques, for navigation, the location of maxima, and the use of user-defined paths were presented. Many other techniques are conceivable. The handling of user-defined constraints and of rotation is under development.

Other areas for further research are the improvement of the performance, extension of the visual representation, the coupling of dedicated visualization and input tools, and the inclusion of sampled data.

Acknowledgement

The authors would like to thank Frits Post (Delft University of Technology) for his constructive criticism during the writing of this paper.

References

[1] P. Hanrahan, J. Kajiya, W. Kreuger, P. Schroeder, and J. Wilhelms. State of the art in volume visualization. In *SIGGRAPH ’91 Tutorial Course Notes, volume 8.* July 1991.

[2] F.W. Young, D.P. Kent, and W.F. Kuhfeld. *Dynamic Graphics for Exploring Multvariate Data,* pages 391–424. Wadsworth Inc., 1988.

[3] S. Feiner and C. Beshers. World within worlds. In *Proceedings Visualization ’92*, pages 283–290. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[4] T. Mihalisin, J. Schwegler, and J. Timlin. Hierarchical multivariate visualization. In *Proceedings Interface ’92*, 1992.

[5] S. Smith, G. Grinstein, and R.D. Bergeron. Interactive data exploration with a supercomputer. In *Proceedings Visualization ’91*, pages 248–254. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[6] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Jour. Amer. Stat. Assoc.*, pages 361–368, 1973.

[7] J. Beddow. Shape coding of multidimensional data on a microcomputer display. In *Proceedings Visualization '90*, pages 238–246. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[8] W.S. Cleveland. *The Elements of Graphing Data*. Wadsworth Inc., 1985.

[9] R. van Liere. Computational steering: a case study. *CWI Quarterly*, 5(3):207–218, 1992.