Hypervolume Visualization: A Challenge in Simplicity

C. L. Bajaj V. Pascucci
Department of Computer Sciences and TICAM
University of Texas, Austin, TX 78733

G. Rabbiolo
Department of Mathematics
Purdue University, West Lafayette, IN 47907

D. R. Schikore
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory, Livermore, CA 94550

Abstract

Hyper-volume visualization is designed to provide simple and fully explanatory images that give comprehensive insights into the global structure of scalar fields of any dimension. The basic idea is to have a dimension independent viewing system that scales nicely with the geometric dimension of the dataset and that can be combined with classical approaches like isocontouring and animation of slices of nD data. We completely abandon (for core simplicity) rendering techniques, such as hidden surface removal or lighting or radiosity, that enhance three dimensional realism and concentrate on the real-time display of images that highlight structural (topological) features of the nD dataset (holes, tunnels, cavities, depressions, extrema, etc).

Hyper-volume visualization on the one hand is a generalization of direct parallel projection methods in volume rendering. To achieve efficiency (and real-time performance on a graphics workstation) we combine the advantages of (i) a hierarchical representations of the hyper-volume data for multiresolution display and (ii) generalized object space splatting combined with texture-mapped graphics hardware acceleration.

The development of a system that implements display techniques for multidimensional datasets requires careful design of both algorithms and user interfaces that scale linearly with the dimension n of the input geometric space. This is a major challenge since straightforward generalizations of standard techniques that are suitable for display of 3D data yield exceedingly intricate interfaces. For example, a view manipulation graphical user interface is usually based on a rotation of the object about Cartesian rotation axes, with possibly unit quaternions internal representations for the rotation group. Unfortunately the number of independent rotation axes grows quadratically with dimension(three in 3D to six in 4D to ten in 5D to fifteen in 6D space). Going back to the basics of parallel projections, we develop an alternative scheme that is very simple to implement and immediately gives a view manipulation graphical user interface that scales linearly with the dimension. One can still utilize matrix or quaternion or higher dimensional rotational group representations, internally for calculations.

The main results of our paper are thus both a multiresolution direct rendering algorithm and scalable graphical user interface that provides insightfull global views of scalar fields in any dimension, while maintaining the fundamental characteristics of ease of use, and quick exploratory user interaction.

1 Introduction

We introduce a new technique for informative visualization of scalar fields embedded in n-dimensional spaces. Examples of scalar fields defined over more than three variables, are gated MRI volume scans of heart motion, time varying data from computational fluids dynamics, molecular vanDerWaal energies as a function of molecular configurations (bond angles).

Our main contributions are:

1. the design and implementation of a new graphical user interface for interacting with parallel projections of n-dimensional scalar fields;

2. the design and implementation of a higher dimension generalization of the traditional splatting algorithm for 3D volume rendering of scalar fields;

This paper extends the research on visualization techniques which provide "global views" of scalar fields independent of the dimension of their embedding space. In this paper we directly render n-dimensional views of the global scalar field.

A number of approaches have been attempted to visualize higher dimensional objects [16, 14, 1]. *The grand tour* technique [2] is based on the idea of projecting the n-dimensional datasets onto a 2-dimensional subspace that is moved along random or selective paths. By visually perceiving coherence in the contiguous 2D images the user can get an idea of the actual structure of the nD object.

Bill Hibbard et al. [13] developed Vis5D for visualizing scalar fields defined over 4D grids. They assume that the last dimension is the time evolution of the dataset, so that they simply animate the display of the isosurface, volume rendering or planar slices. Hence their approach is fundamentally to animate traditional scientific visualizations. For their purpose they achieve good results but the technique cannot be generalized for higher dimensional datasets.

Hanson and Heng [9] introduced a technique to present 3D scalar field by means of 4D elevation models in the same way 2D scalar fields can be show as 3D terrains. Rotating the 4D pseudocolored elevation model the user can *see* its structure enhanced by the illumination scheme developed in [11]. The approach has been later generalized [10] to be suitable for display of more general 4D geometric objects and made more efficient to provide the speed necessary for good user interaction.

Laur and Hanrahan [15] accelerate the 3D splatting [20] algorithm adopting an octree hierarchical representation of the volume data. We generalize this approach showing also how the relative storage overhead of the full 2^n-tree hierarchy decreases as the dimension of the embedding space increases.

The *HyperSlice* approach [19] introduced by van Wijk and van Liere uses an array of 2D slices of high dimensional data. Various interaction approaches are applied for different user objectives. While this approach provides intuitive user interaction, for higher dimensions it becomes an increasingly difficult problem to fuse the multiple slices and establish an understanding of more complex patterns in the data.

On important aspect in developing a visualization tool for n dimensional data is the design of an interface simple enough to make the user interaction reasonably simple and intuitive. Duffin and Barret [6] addressed this problem by presenting a simplified 2D user interface to specify an n-dimensional orthonormal rotation matrix.

The approach introduced in the present paper can be considered a good complement to the previous approaches listed above. A common aspect of all such approaches is that it can provide a realistic and detailed representation of “local” (in time or space) feature of the scalar field. After looking at many sequences of pictures the trained user attempts to form in his own mind a “global picture” of the dataset. The approach proposed here tries to avoid relying (as much as possible) on the geometric abstraction capabilities of users, providing them directly with global projections of higher dimensional spaces with real time interaction in a reasonably similar way to how they would explore their own physical world.

There are two main challenges in developing such an exploratory approach to visualization:

- to make the user interaction be sufficiently intuitive, simple and “linearly” scalable with the dimension
- to efficiently render such a large amount of data (note that the size of the dataset grows exponentially with the dimension n of the embedding space

2 Hypervolume Projection Transformation

In this section we discuss how a 2D “view” of an nD object is geometrically defined and how such a definition impacts the user interface for view selection. We consider parallel projections. It is well known since the last century (see fundamental theorem by K. Pohlke and H. A. Schwarz in [7]) that given the image of a reference coordinate axes of a parallel projection, the projection itself is completely defined. A short informal proof of this fact is as follows. Consider the parallel projection as in figure 1. In the (x, y) view plane we draw the projection of the reference axes (X, Y, Z) of the 3D space. The unit vectors of the three axes are projected respectively onto the vectors $\vec{l}_1, \vec{l}_2, \vec{l}_3$ and the projection of the origin O is given by $o + \vec{l}_O$. The projection of the point $P(a, b, c)$ is thus given by:

$$o + \vec{l}_P = o + \vec{l}_O + a \cdot \vec{l}_1 + b \cdot \vec{l}_2 + c \cdot \vec{l}_3. \qquad (1)$$

This vectorial equation defines the parallel projection as the linear mapping $\Pi(\Re^3 \rightarrow \Re^2)$ that maps the 3D point (X, Y, Z) onto the 2D point (x, y):

$$\Pi(\Re^3 \rightarrow \Re^2) : (X, Y, Z) \mapsto (x, y);$$

$$\begin{cases} x = l_O^x + X \cdot l_1^x + Y \cdot l_2^x + Z \cdot l_3^x \\ y = l_O^y + X \cdot l_1^y + Y \cdot l_2^y + Z \cdot l_3^y \end{cases} \qquad (2)$$

where l_i^j is the j^{th} component of the vector $\vec{l}_i$.

In the system (2) we fix the triple (X, Y, Z) to compute the corresponding 2D point (x, y). Symmetrically we fix a particular pair (x, y) and determine all the (X, Y, Z) that satisfy the system (2). In the latter case the set of all the solutions is a line parallel to the vector:

$$\vec{\pi} = \left\| \begin{matrix} l_1^x & l_2^x & l_3^x \\ l_1^y & l_2^y & l_3^y \end{matrix} \right\|$$

$\vec{\pi}$ is the direction of projection that defines the parallel projection. Note that changing the three vectors $\vec{l}_1$,$\vec{l}_2$ and $\vec{l}_3$ in all possible ways, we obtain all the possible parallel projections. Moreover it is easy to show that a valid view is given by any surjective linear mapping Π. This implies that the only constraint that the triple $\vec{l}_1, \vec{l}_2, \vec{l}_3$ needs to satisfy is:

$$rank \begin{pmatrix} l_1^x & l_2^x & l_3^x \\ l_1^y & l_2^y & l_3^y \end{pmatrix} = 2$$

Simple additional linear constraints over the triple $\vec{l}_1$,$\vec{l}_2$,$\vec{l}_3$ guarantees the parallel projection to be orthographic (isometric, dimetric or trimetric) or oblique (cavalier, cabinet or generic) [17].

2.1 The general mapping $\Pi(\Re^n \rightarrow \Re^2)$

In the previous section we have discussed the methodology for defining a parallel projection from 3D space to the 2D

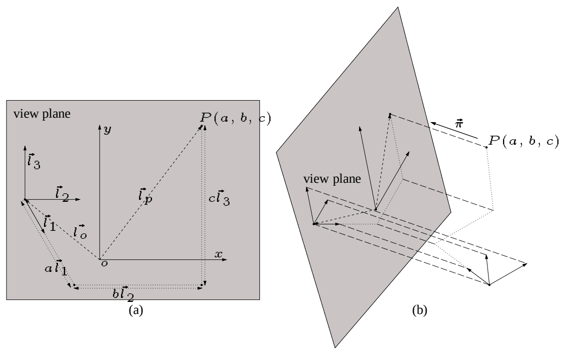


Figure 1: Parallel projection definition from 3D to 2D. (a) Relationship between 3D and 2D coordinates of a point. (b) Parallel projection of the 3D reference system onto the 2D view plane.

view plane. In this section we show how this allows generalization of parallel projection transformation from nD space to a 2D plane. We also show how one can grow the dimension n of the object space without increasing the complexity of the parallel projection definition scheme.

Let $\{\vec{e}_1, \ldots, \vec{e}_n\}$ denote the canonical basis in the n-dimensional object space $\Re^n$. The parallel projection

$$\Pi(\Re^n \to \Re^2) : (X_1, \ldots, X_n) \mapsto (x, y)$$

is a linear transformation and is therefore completely determined by the n vectors $\vec{l}_i = \Pi(\vec{e}_i), \quad i = 1, \ldots, n$. In fact by linearizing the image under Π of a point $P(X_1, \ldots, X_n) = \vec{e}_1 \cdot X_1 + \vec{e}_2 \cdot X_2 + \cdots + \vec{e}_n \cdot X_n$ is given by:

$$\begin{aligned}\Pi(P) &= \Pi(\vec{e}_1 \cdot X_1 + \vec{e}_2 \cdot X_2 + \cdots + \vec{e}_n \cdot X_n) \\ &= \Pi(\vec{e}_1) \cdot X_1 + \Pi(\vec{e}_2) \cdot X_2 + \cdots + \Pi(\vec{e}_n) \cdot X_n \\ &= \vec{l}_1 \cdot X_1 + \vec{l}_2 \cdot X_2 + \cdots + \vec{l}_n \cdot X_n\end{aligned}$$

As in the 3D case, the mapping defines a system of two linear equalities:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} l_1^x & \cdots & l_d^x \\ l_1^y & \cdots & l_d^y \end{pmatrix} \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} \tag{3}$$

Changing the vectors $\vec{l}_i$ we obtain all the possible projection matrices. For a projection matrix to be "valid" it is sufficient that it is full rank (otherwise we would project onto a line or onto a point). Changing the vectors $\vec{l}_i$ in all the valid ways we produce all the possible 2D views of the dataset.

To grow the dimension n of the object space means to add new $\vec{l}_i$ vectors and hence columns to the projection matrix. What in the 3D case is called the "direction of projection" is in general the kernel K of the mapping Π. Since the projection is given by a full rank $2 \times n$ matrix the kernel of Π is a $(n-2)$-dimensional linear space. In 3D we have that points aligned along the direction of projection are projected onto the same 2D point. In the nD space we have that two points are projected onto the same 2D point if and only if they are contained in the same $(n-2)$-dimensional affine space parallel to the kernel of the projection Π.

2.2 Occlusion

The nice property of the 3D case is that the kernel of the mapping Π is a 1-dimensional affine space. Hence, one can define a total order among the 3D points projected onto the same point in 2D. This automatically yields a sound definition of occlusion (or visibility ordering) between points in 3D space. A point P_1 occludes a point P_2 if (i) $\Pi(P_1) = \Pi(P_2)$ and (ii) and P_1 is "nearer" than P_2 or more formally P_1 has smaller rank than P_2 in the order within their common projection ray. For dimensions four or higher, a unique total order cannot be defined at least in a sound geometric sense and independent from the selected coordinate system. Artificial (partial) orders can be easily imposed on the points of the $(n-2)$-flat that is projected onto a single 2D point. Moreover, in 3D, given a direction of projection, one can define two alternative occlusion orders that give two views of an object in a fixed position, say front view and a back view. In 4D or higher dimensions things get immediately more intricate since one needs to define a coordinate system within the kernel of the mapping Π to define a certain occlusion order. This involves two main difficulties:

- Already in 4D, one gets an infinity of different occlusion orders, each giving a different view of the object (keeping the position of the object fixed). This implies additional degrees of freedom to explore and hence longer interaction times.
- One needs to define a reference system within the kernel, but by definition, the entire kernel is projected into a single point. So we need either some separate view that "shows" the kernel or a numerical interface to control the occlusion order.

Apart from the complexity introduced by an occlusion order, the more fundamental issue is the meaning of such an occlusion and what enhancement if any, the occlusion yields to the images displayed. In the 3D case, occlusion certainly enhances the realism of the images and better highlights silhouettes of the viewed objects. However a case can be made is that it also hides important features of the dataset. In our case, since we are looking especially at scalar fields to provide views that displays its topological structure, we have chosen not to provide an occlusion information in our rendering.

2.3 The graphical user interface

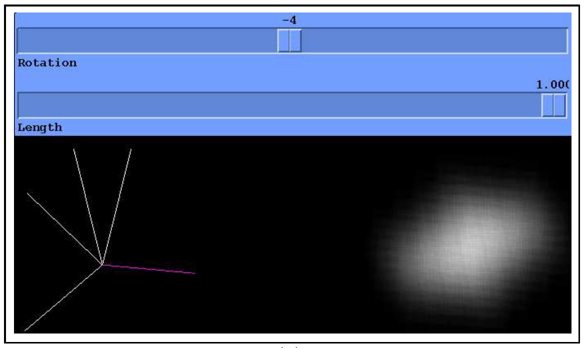
¿From the general mapping defined in the previous section, we derive the a graphical user interface for n dimensional object exploration that differs from the classical rotation-based interface. usually the basic view transformation that allows one to change the view of an object is rotation (see section 8.2.6 of [8]), since translation and scaling allow one to only change the “focus” of a view, keeping the display substantially the same. from a users perspective interaction with rotations has two main difficulties:

- since one rotation is defined by 2 coordinate axes (the axes that span the rotation plane) we have that in nd space the number of independent coordinate rotations are $\binom{n}{2}$. when one looks at a simple rigid body configuration space of a 3d object where the number n of coordinate axes is six (three translations plus three rotations) one needs to explore 15 different planes of rotation.

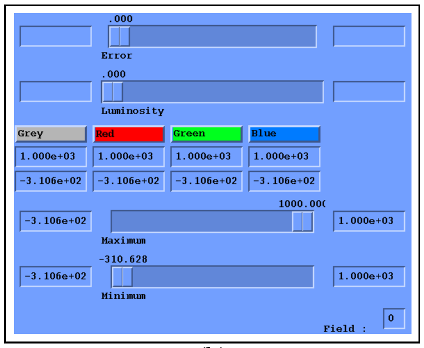
- the user is usually provided with the ability to rotate an object in the coordinate planes so that a rotation in a generic (non-coordinate) plane needs to be obtained by combination of the elementary transformations. this task gets really confusing in more than three dimension since one cannot rely on navigation experience acquired in physical 3d space.

A classical rotation-based interface grows quadratically with the dimension n of the embedding space getting quickly unsuitable for simple and fast interaction if $n > 3$. we employ an alternative approach that scales linearly with the dimension n making the interface more suitable for a higher dimensional approach (in the next subsection we show how this interface can be enriched with rotational interaction when needed). The main idea is that the user can modify the image of the reference system instead of changing the position of the object in the embedding space. As shown in the previous section, this approach provides sufficient degrees of freedom to explore all the possible views of the object. in the same time the approach reduces substantially the number of parameters that the user needs to deal with.

Consider the reference system of a five-dimensional space as in figure 2(a). The user can select with the mouse any of the axes (highlighted in red) and then rotate/stretch it in any position. For more accurate interaction, some buttons and sliders are provided, allowing the user to perform the same operation even if there are overlapping axes or the precision required cannot be achieved with a simple pick-and-move operation. In this way the user can explore all the possible view with much less redundancy. Since the user deals with one axis at a time the complexity of the interface is only equal to the dimension n of the embedding space. In a six dimensional configuration space the user just needs to adjust the length and orientation of the six vectors $\vec{l}_1$ $\vec{l}_2$ $\vec{l}_3$ $\vec{l}_4$ $\vec{l}_5$ and $\vec{l}_6$ projections of the embedding space unit vectors. This is simpler that rotating in fifteen possible planes. It also may be more intuitive for axes that do not correspond to the physical extent of the object (e.g. time or rotational degrees of freedom). Of course when the user needs to rotate the object s/he


-4
Rotation
1.000
Length

(a)


.000
Error
.000
Luminosity

Grey	Red	Green	Blue
1.000e+03	1.000e+03	1.000e+03	1.000e+03
-3.106e+02	-3.106e+02	-3.106e+02	-3.106e+02

-3.106e+02 1000.00(1.000e+03
Maximum
-3.106e+02 -310.628 1.000e+03
Minimum
Field : 0

(b)

Figure 2: (a) User interface for selection of viewing parameters. The axes on the left can be directly selected and stretched or rotated. The image of the standard splat is show on the right. The sliders on top allow fine adjustment of the stretching and rotation parameters. (b) User interface for selection of transfer function parameters.

can still do that directly as in the classical approach, as detailed next.

2.3.1 Rotation

In this section we show how one can easily integrate in our image-space approach the classic object-space rotation. Consider (without loss of generality) a rotation in the X, Y plane by an angle θ. It maps the X, Y coordinate to X', Y' by the rule:

$$(X, Y) \mapsto (X', Y') : \qquad \begin{cases} X' = \cos\theta \cdot X - \sin\theta \cdot Y \\ Y' = \sin\theta \cdot X + \cos\theta \cdot Y \end{cases} \tag{4}$$

while all the other $n - 2$ coordinates remain unmodified. To obtain the equivalent transformation in our image-based projection definition, we substitute equation (4) into the generalization of (2) to obtain:

$$\Pi(\Re^n \to \Re^2) : (X, Y, Z, \ldots) \mapsto (x, y);$$

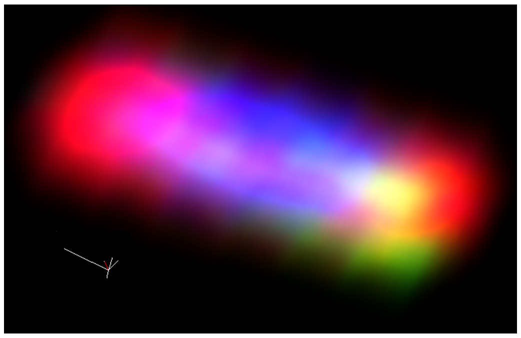


Figure 3: 5D interaction energy scalar field (Red=attraction, Blue=repulsion, Green=free movement). The axes configuration is reported on the bottom left (the stretched axis corresponds to a rotational degree of freedom).

$$\begin{cases} x = l^x_O + l^x_1 \cdot (\cos\theta \cdot X - \sin\theta \cdot Y) + \\ \qquad\qquad l^x_2 \cdot (\sin\theta \cdot X + \cos\theta \cdot Y) + l^x_3 \cdot Z + \cdots \\ y = l^y_O + l^y_1 \cdot (\cos\theta \cdot X - \sin\theta \cdot Y) + \\ \qquad\qquad l^y_2 \cdot (\sin\theta \cdot X + \cos\theta \cdot Y) + l^y_3 \cdot Z + \cdots \end{cases} \tag{5}$$

that can be rewritten as:

$$\begin{cases} x = l^x_O + (l^x_1 \cdot \cos\theta + l^x_2 \cdot \sin\theta) \cdot X + \\ \qquad\qquad (-l^x_1 \cdot \sin\theta + l^x_2 \cdot \cos\theta) \cdot Y + l^x_3 \cdot Z + \cdots \\ y = l^y_O + (l^y_1 \cdot \cos\theta + l^y_2 \cdot \sin\theta) \cdot X + \\ \qquad\qquad (-l^y_1 \cdot \sin\theta + l^y_2 \cdot \cos\theta) \cdot Y + l^y_3 \cdot Z + \cdots \end{cases} \tag{6}$$

Hence we provide a slider to control the rotation parameter θ and at each rotation step replace $\vec{l_1}$ with $\vec{l'_1}$ and $\vec{l_2}$ with $\vec{l'_2}$ where:

$$\begin{cases} \vec{l'_1} = (\ \ l^x_1 \cdot \cos\theta + l^x_2 \cdot \sin\theta, \ \ l^y_1 \cdot \cos\theta + l^y_2 \cdot \sin\theta)^T \\ \vec{l'_2} = (\ -l^x_1 \cdot \sin\theta + l^x_2 \cdot \cos\theta, \ -l^y_1 \cdot \sin\theta + l^y_2 \cdot \cos\theta)^T \end{cases} \tag{7}$$

In this way, the user is provided with the same intuitive ability to rotate objects as in a classical 3D user interface.

3 Hyper-Volume Splatting

This section first provides a working example of the Hyper-Volume splatting approach and then details the splatting algorithm that we have implemented with the three following properties:

- the use of a transfer function that highlights the basic structural features of the scalar field;
- the use of a multiresolution hierarchical approach to speed up the drawing when is provided a user specified bound on the tolerated error;
- the use of a splatting algorithm that takes advantage from texture mapping graphics hardware.

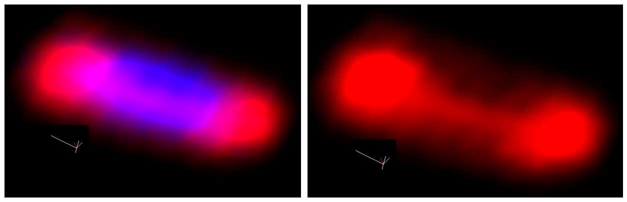


Figure 4: 5D interaction energy scalar field (Red=attraction, Blue=repulsion, Green=free movement).
Same view as is figure 3 but highlighting only some of the energy components.

3.1 Example of Hyper-Volume Splatting(5D Molecular Interaction Potential)

Consider a pair of molecules, a small ligand (methanol) and a large receptor (Ecballium Elaterium Trypsin Inhibitor[1]), of which one wants to study the possibility of docking. At this purpose one needs to understand how the interaction energy between them changes as they change relative position. In particular we regularly sample the configuration space of the ligand translations along the x,y,z axes and rotations around the x and y axes (assuming rough symmetry of the ligand with respect to the z axis). For each sampled position of this five dimensional space one gets a particular value of the interaction energy (sum of electrostatic interaction and Van der Waals interaction components) defining a scalar field sampled over a 5D regular grid. Figure 3 shows the direct rendering of the 5D scalar field highlighting in Red regions of attracting energy, in Blue region of repulsion energy and in Green region in free movement of the ligand. The display is performed directly by projection form 5D space to 2D space without any slicing/isocontouring stage so that the information contained in the dataset is preserved in its globality. The axes reported on the bottom left of the picture show how one of the degrees of freedom (a rotation) is stretched more than the others to enhance better its influence with respect to the overall scalar field structure. In this case it is clear from the two large red spots that correspondingly to high and low values of that degree of freedom we get more attraction values than for intermediate values (such rotation are probably more advantageous for a docking of the ligand with the receptor).

Progressively removing all the color but the red as shown in figure 4 one can also see how these two large red regions are connected by a narrow tunnel. Deeper understanding of the scalar field structure is of course provided by interactive navigation in the dataset structure. For example figure 5 shows a second view of the dataset of figure 3 in which the axis corresponding to the second rotational degree of freedom is also stretched (as in the reference system on top). From this view one can see that the two large red regions are in turn divided each into two. On the left picture one can no-

[1]The Ecballium Elaterium Trypsin Inhibitor can be found in the file 2eti.pbd available form the Protein Data Bank `http://pdb.pdb.bnl.gov/`

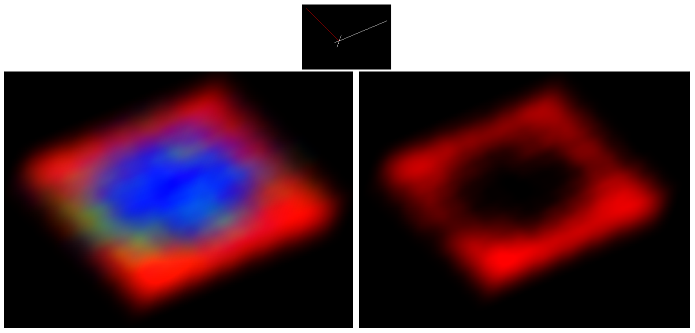


Figure 5: 5D interaction energy scalar field (Red=attraction, Blue=repulsion, Green=free movement).
Same scalar field as in figure 3 but from a different view.

tice an interesting small site in green where the ligand can move along the interface with the receptor without being subject to a repulsion force. Again one can show only the attraction component (in red) and see clearly that in the central region the energy is completely repulsive (see right image). Note that this kind of check by partial color removal is necessary because some red spots might be hidden within the blue region.

3.2 Transfer Function

The definition of a "good" transfer function is highly dependent on the type of scalar field displayed and the features that one needs to highlight. In low dimensional cases interesting techniques have been developed to support the automatic selection of transfer functions which emphasize the important structures of a scalar field [12, 4]. In our current implementation we use the three color components to highlight regions of the field that encompass values in different ranges. Using the interface component in figure 2(b) one can interactively select the range of the scalar field values associated to each color and the relative intensity of each color component. The user clicks on one color button to select the currently modified component and then uses the sliders to determine the associated range in the scalar field and scale factor in luminosity. Interactively the view is updated accordingly to the modified parameters. In a particular the image generation is defined as follows. The red color component $R(p)$ in the pixel p of the image is given by the integral:

$$R(p) = L_r \int_{p+K} F_r \, dk$$

where the domain of integration K is the kernel of the projection Π (in 3D is the projection ray through p), dk is the $n-2$ dimensional differential element, L_r is the luminosity of the red component and F_r is the scalar field value normalized in the (min_r, max_r) range associated to the red color component. Similar formulas can be written for the blue and green component providing the complete coloring scheme for a given view.

3.3 Efficient Splatting

The splatting algorithm is particularly simple and efficient in the case of parallel projections since all the splats have the same shape: they differ only in color intensity and eventually in scale factor (see the hierarchical representation in the next subsection). In this case the display algorithm has two main stages:

1. compute the shape of the standard splat or footprint;
2. draw each voxel by copying the standard splat scaled by color intensity and size.

Note that we are not considering the ordering the voxels to be splatted since we do not perform occlusion between voxels with overlapping images.

3.3.1 Splat Computation

The input data we are displaying is a decomposition of the space in elementary volume regions or voxels. At the center of each voxel the scalar field is sampled and assumed constant within the voxel. In this framework each splat is the projection of a n-dimensional cube (the voxel) of constant transparency value (the scalar field value) onto the 2D image space. From spline theory we get that the luminosity distribution of the splat is a bivariate box spline [5]. This fact allows us to compute the splat luminosity distribution exactly or to control the error of an approximated version we might use instead. In particular we observe that the splatting algorithm applied to a volume of constant intensity is indeed an approximation of a bivariate box spline. The level of approximation depends on what splat one uses and on the number of voxels in which the value is decomposed. This allows use to pursue a bootstrapping technique by using the splatting algorithm to generate a good splat to be used in the actual rendering of the scalar field. Note that this approach, again transforms the problem of drawing a good splat by projecting a cube (in this case n-dimensional), into a simple reuse of the splatting algorithm. Of course in the initial splat drawing stage instead of an exact splat we may use a simple square. To obtain an exact drawing of the initial splat we would need a square of size equal to one pixel (see [5]) but in practice a fairly larger one is sufficient since in the successive use of the splat its initial footprint will be shrunk to the necessary size. Figure 6 show the splat obtained for different axes configurations in four, five and six dimensional spaces.

Note also that in the case of the hierarchical approach the splats of any level in the hierarchy have exactly the same shape. They need only be scaled in size and color intensity.

3.4 Hierarchical Representation

One major problem that arises while dealing with multidimensional scalar fields is that the size of the dataset grows exponentially with the dimension of the embedding space.

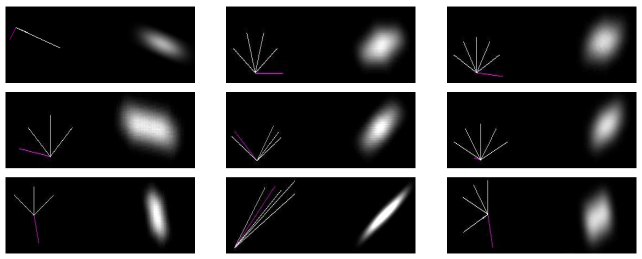


Figure 6: Standard splat footprint for different axes orientations in four dimensions (a), five dimensions (b) and six dimensions (c).

For example, an n-dimensional scalar field sampled on regular grid with k samples in each dimension one has k^n samples. A regular grid in the 6-dimensional rigid body configuration space with only 64 samples in each direction has already $2^{36} \simeq 68$ billion samples.

To deal with such large datasets we adopt a 2^n-tree hierarchical representation where n is the dimension of the embedding space. We build the hierarchy in a bottom up coarsening scheme by merging at each step groups of 2^n adjacent voxels and averaging their function values, with precomputed error bounds. In the display stage we recursively visit the 2^n-tree nodes in a Depth First Traversal from the coarser level and stop when the user specified error bound is satisfied (the error value is set by the user with the help of the top slider in the interface shown in figure 2(b)). In this way, the user is allowed to trade accuracy for speed in a fully controlled manner.

3.4.1 Hierarchy Storage Overhead

There are at least two possibilities in storing the 2^n-tree hierarchy: (i) store the complete 2^n-tree in an array, independent from the sample values (ii) store the 2^n-tree in a 2^n-linked list to avoid multiple storage for neighboring voxels with equal sample value. In general it is not clear which approach is more convenient. It could be even better to have, instead of a 2^n tree, a bin-tree where each binary division is performed along one of the n coordinate directions [21, 18]. Our choice to store the full 2^n-tree is derived from the following considerations that show how the full hierarchy storage overhead decreases as the dimension of the dataset increases.

Consider a regular grid of total size M embedded in the nD space. Assume for simplicity that the grid has the same number m of samples in all the n directions (we have $M = m^n$) and $m = 2^h$ for some h. Note that the assumption made simplify the following formulas without altering the result that we shell derive. Since the coarsening stage from one level to the next in the hierarchy is based on grouping 2^n adjacent cells we have that the number of cells is reduced at each level by a factor of 2^n. Overall the storage M^* of the complete 2^n-tree is:

$$M^* = \sum_{i=0}^{h} \frac{\left(2^h\right)^n}{\left(2^n\right)^i} = \sum_{i=0}^{h} \left(2^n\right)^{h-1} = \sum_{j=0}^{h} \left(2^n\right)^j$$

Using a geometric series formula[2] we have that the relative storage overhead is given by:

$$\frac{M^*-M}{M} = \frac{\frac{(2^n)^{h+1}-1}{(2^n-1)} - (2^n)^h}{(2^n)^h} = \frac{(2^n)^{h+1}-1-(2^n)^h(2^n-1)}{(2^n)^h(2^n-1)} = \frac{(2^n)^h-1}{(2^n)^h(2^n-1)} < \frac{(2^n)^h}{(2^n)^h(2^n-1)} = \frac{1}{(2^n-1)}$$

The overhead due to the hierarchical representation can be bounded by a term that decreases exponentially with the dimension n so that the hierarchy storage overhead is very small with respect to the input dataset especially for $n > 3$.

3.5 Hardware Acceleration

Once the standard splat is computed we store its image in the texture map memory. Each splat is then rendered as a 2D textured polygon. In this way we take full advantage from the hardware acceleration of modern graphics workstations. We can render a large number of splats quickly achieving almost interactive rates for fairly complex datasets. In our hierarchical implementation we also need to compute different splats for different levels of resolution. To perform this operation we use mipmaps so that we automatically obtain the best scaling in size of the splat simply by drawing a larger textured polygon onto the screen.

The ability to interactively view and manipulate three dimensional textures (available on high end graphics workstations) could also be potentially used. The general mapping $\Pi : \Re^n \rightarrow \Re^2$ can easily be modified to a map $\Pi : \Re^n \rightarrow \Re^3$ to produce hypervolume splats, which can be then be interactively explored. The full potential of this exploration will however be only realized when true volumetric displays become available.

4 Further Enhancements and Future Directions

In addition to the coupling with classical visualization techniques such as isocontouring and slicing, we are also exploring the enhancement of the hypervolume view with additional computed structural information such as the scalar topology diagram, a one-dimensional roadmap of the nD scalar field [3]. Due to the high-dimensionality of the data, it remains an open issue how to highlight fundamental structural feature of the scalar field without occluding a large portion of the displayed view. This problem may be cast into a nD embedded graph simplification problem. We are also investigating automated colormap definition techniques driven by the goal of identifying topologically interesting features.

[2] Remember that $\sum_{i=0}^{k} x^i = \frac{x^{k+1}-1}{x-1}$

References

[1] ANDREWS, D. F. Plots of high-dimensional data. *Biometrics 28* (1972), 125–136.

[2] ASIMOV, D. The grand tour: a tool for viewing multidimensional data. *SIAM Journal on Scientific and Statistical Computing 6*, 1 (Jan. 1985), 128–143.

[3] BAJAJ, C. L., PASCUCCI, V., AND SCHIKORE, D. Visualization of scalar topology for structural enhancement. In *Proceedings of IEEE Visualization '98 (to appear)* (1998).

[4] BERGMAN, L., ROGOWITZ, B., AND TREINISH, L. A rule-based tool for assisting colormap selection. In *Visualization '95 Proceedings* (Oct. 1995), G. M. Nielson and D. Silver, Eds., pp. 118–125.

[5] DE BOOR, C., HÖLLIG, K., AND RIEMENSCHNEIDER, S. *Box Splines*. No. 98 in Applied Mathematical Sciences. Springer-Verlag, 1993.

[6] DUFFIN, K. L., AND BARRETT, W. A. Spiders: A new user interface for rotation and visualization of N-dimensional point sets. In *Proceedings of the Conference on Visualization* (Los Alamitos, CA, USA, Oct. 1994), R. D. Bergeron and A. E. Kaufman, Eds., IEEE Computer Society Press, pp. 205–211.

[7] FIEDLER, G. *Die darstellende Geometrie*. XXX, 1871.

[8] FOLEY, J. D., VAN DAM, A., FEINER, S. K., AND HUGHES, J. F. *Computer Graphics: Principles and Practice, 2nd ed.* Addison-Wesley, Reading MA, 1990.

[9] HANSON, A., AND HENG, P. Four-dimensional views of 3D scalar fields. In *Proceedings of IEEE Visualization '92* (1992), pp. 84–91.

[10] HANSON, A. J., AND CROSS, R. A. Interactive visualization methods for four dimensions. In *Proceedings of IEEE Visualization '93* (San Jose, CA, Oct. 1993), G. M. Nielson and D. Bergeron, Eds., IEEE Computer Society Press, pp. 196–203.

[11] HANSON, A. J., AND HENG, P. A. Illuminating the fourth dimension. *IEEE Computer Graphics and Applications 12*, 4 (July 1992), 54–62.

[12] HE, T., HONG, L., KAUFMAN, A., AND PFISTER, H. Generation of transfer functions with stochastic search techniques. In *Visualization '96 Proceedings* (Oct. 1996), pp. 227–234.

[13] HIBBARD, W. L., ANDERSON, J., FOSTER, I., PAUL, B. E., JACOB, R., SCHAFER, C., AND TYREE, M. K. Exploring coupled atmosphere-ocean models using Vis5D. *The International Journal of Supercomputer Applications and High Performance Computing 10*, 2/3 (Summer/Fall 1996), 211–222.

[14] HOLLASCH, S. R. Four-space visualization of 4D objects. M.sc. thesis, Arizona State University, Aug. 1991.

[15] LAUR, D., AND HANRAHAN, P. Hierarchical splatting: A progressive refinement algorithm for volume rendering. In *Computer Graphics (SIGGRAPH '91 Proceedings)* (July 1991), T. W. Sederberg, Ed., vol. 25, pp. 285–288.

[16] NOLL, M. A. A computer technique for displaying n-dimensional hyperobjects. *Communications of the ACM 10*, 8 (August 1967), 469–473.

[17] PASCUCCI, A., AND PASCUCCI, V. Uso del calcolatore nalla produzione, elaborazione ed archiviazione di proiezioni parallele. In *Atti del convegno L'immagine nel rilievo* (1992), C. Cundari, Ed., edizione Gangemi.

[18] SHAFFER, C. A., JUVVADI, R., AND HEATH, L. S. A generalized comparison of quadtree and bintree storage requirements. *Image and Vision Computing 11*, 7 (1993), 402–412.

[19] VAN WIJK, J. J., AND VAN LIERE, R. Hyperslice. In *Proceedings of IEEE Visualization '93* (San Jose, CA, Oct. 1993), G. M. Nielson and D. Bergeron, Eds., IEEE Computer Society Press, pp. 119–125.

[20] WESTOVER, L. Footprint evaluation for volume rendering. *Computer Graphics 24*, 4 (Aug. 1990), 367–376.

[21] WISE, K. Generalized comparison of bintree and 2^n-tree storage requirements. Technical Report 053, University of Bath, Department of mechanical Engineering, December 1997.