A Radial Focus+Context Visualization for Multi-Dimensional Functions

Sanjini Jayaraman, Chris North
Center for Human Computer Interaction
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, USA
sjayaram@vt.edu, north@cs.vt.edu
http://infovis.cs.vt.edu/

Abstract

The analysis of multidimensional functions is important in many engineering disciplines, and poses a major problem as the number of dimensions increases. Previous visualization approaches focus on representing three or fewer dimensions at a time. This paper presents a new focus+context visualization that provides an integrated overview of an entire multidimensional function space, with uniform treatment of all dimensions. The overview is displayed with respect to a user-controlled polar focal point in the function's parameter space. Function value patterns are viewed along rays that emanate from the focal point in all directions in the parameter space, and represented radially around the focal point in the visualization. Data near the focal point receives proportionally more screen space than distant data. This approach scales smoothly from two dimensions to 10-20, with a 1000 pixel range on each dimension.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces – Graphical user interfaces; I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction Techniques.

Keywords: Visualization, multidimensional functions.

1 INTRODUCTION

Multi-dimensional functions are commonplace in many disciplines. Insight into their nature and behavior is critical for success. For example, in engineering design, system output performance is affected by several input parameters. Such a system can be viewed as a multidimensional function mapping the input parameters to the output performance of the system. Designers need to analyze and discover patterns among inputs that affect system performance to formulate design decisions. The multi-dimensional nature of such complex functions makes them difficult to understand and analyze. A visual overview of the entire data space would help designers to analyze the trends in system performance and study design tradeoffs. Focusing on more details of a localized region can then enable fine-tuning performance.

IEEE Visualization 2002 Oct. 27 - Nov. 1, 2002, Boston, MA, USA
0-7803-7498-3/02/$17.00 © 2002 IEEE

Multidimensional functions map values of multiple independent variables (dimensions) to values of dependent variables. A *d*-dimensional function has *d* independent variables. This paper focuses on single-valued functions with one dependent variable, that are defined within a multidimensional box bounded by minimum and maximum values on each dimension. Mathematically,

$$y = f(x_1, x_2, \ldots, x_d), \text{ where } x_i \in [\min_i \ldots \max_i].$$

Multidimensional functions are typically expressed in one of two forms:

- Mathematical *formulas*: The relationship between the independent and dependant variables is well defined as a mathematical expression. The value of the dependent variable can be directly calculated from the values of the independent variables. This form is often used in mathematical modeling applications. For example, network communication functions model throughput as a function of several factors such as delay, jitter and bandwidth of the communication channel.
- Multidimensional *arrays* of sampled values: The bounded multidimensional space defined by the independent variables is discretely sampled on a uniform grid. Dependent variable values are sampled on these grid points, and stored in a multidimensional array. Values between grid points can be approximated with interpolation. This format is used when formulas are unknown or not easily specified. For example, such data is often collected from scientific experiments, complex simulations, or advanced models such as neural networks.

For one- or two-dimensional functions, visualization is straightforward [CMS99]. One-dimensional functions can be viewed with simple line graphs (maps independent variable to x, dependent variable to y). Two-dimensional functions can be viewed with height fields (maps independent variables to x and y, dependent variable to z) or colored heat maps (maps dependent variable to color). Three-dimensional functions require non-trivial approaches such as volume rendering or transparent 3D isosurfaces.

Beyond three- or four-dimensional functions, visualization becomes very difficult because such spaces are beyond the physical world and therefore difficult to depict and conceptualize. Typically, slicing or projection is used to visualize only three or fewer of the dimensions at a time. *Slicing* selects constant values for the remaining non-visualized dimensions. *Projection* with aggregation accumulates (e.g. averages) function values in the subspaces defined by the remaining dimensions.

However, showing a limited number of dimensions or limited regions of the space prevents the user from gaining an overview or holistic picture of the functional behavior throughout the space. Users must mentally integrate multiple views or navigation sequences, and entire regions of the space can be missed. This makes it difficult, for example, for system engineering designers to decide how to adjust system parameters to improve output performance or find stable regions of performance.

This paper presents PolarEyez, a new focus+context visualization for multidimensional functions with uniform treatment of all dimensions. Its polar radial layout provides an integrated contextual overview of the entire function space from the perspective of a focus point within the space. Data is revealed in more detail near the focus. The overview helps guides users to interesting regions in the space, and users can navigate the focus to examine regions in more detail.

2 RELATED WORK

It is important to distinguish between visualizations intended for two different, but related, classes of data: multidimensional functions and multidimensional relations. Relational data generally contains a relatively small number of defined tuples scattered throughout its multidimensional space, and does not necessarily distinguish between independent and dependent variables. Functional data is significantly denser, because every point in the multidimensional space defined by its independent variables has a dependent variable value. Functional data in the form of fully sampled arrays gets large quickly. A d-dimensional function with a range of r samples per dimension results in r^d values (e.g. 10 samples on 10 dimensions is 10 billion data points). Continuous functions have infinite data. Differences between functional and relational data lead to different types of users tasks, and different approaches for visualization.

When visualizing multidimensional functions, the goal is typically to view patterns of the dependent variable across the multidimensional space of the independent variables. Hence, independent variables should be mapped to spatial attributes in the visualization (spatial substrate [CMS99]). This requires arranging the multidimensional space on the 2D screen space in some fashion. Three primary approaches have emerged, each of which recomposes slices of the data in different arrangements. HyperSlice [VV93] is similar to the scatterplot matrix, showing a 2D heat-map slice for all pairs of independent variables. Each slice can be navigated through the other dimensions. Mihalisin [MTS91] uses similar 2D slices, but attempts to display all possible slices eliminating the need to navigate the slices. Slices are organized in a hierarchical 2D grid. Worlds-within-Worlds [FB90] nests slices (typically 2D height fields) within outer 3D coordinate frames. The location of the inner slice origin within the outer frame determines the constant values for the outer dimensions in the slice.

While these approaches have been a significant advance, serious problems remain. First, these slicing and projection approaches do not provide an integrated overview of the space and treat dimensions non-uniformly. Users can only view the function with respect to a few dimensions at a time, and must navigate to view other portions of the space. Second, scalability in number of dimensions d and range per dimension r is limited because d and r both compete for horizontal screen space in these approaches. For example, in Mihalisin's approach, $r^{d/2} < screenWidth$. This severely limits d (to approximately 4 or 6), and additional dimensions cannot be shown or must be projected.

Visualizations for multidimensional relations, such as Parallel Coordinates [Ins97], Star Coordinates [Kan00], TableLens [RC94], Spotfire [AW95], and Attribute Explorer [TSD96], have more freedom to explore alternate mappings. However, these approaches are typically not well suited for multidimensional functions. The dense nature of functional data overwhelms these mappings, or creates serious occlusion. For example, Figure 1 shows a TableLens visualization of a 5-dimensional function with 9 samples per dimension. Most of the screen space is spent encoding all possible combinations of values for the independent variables (columns 1-5). Vertical space in the dependent variable column (column 6) is overcrowded, because d and r compete for vertical space, attempting to squeeze 9^5 values into the column.

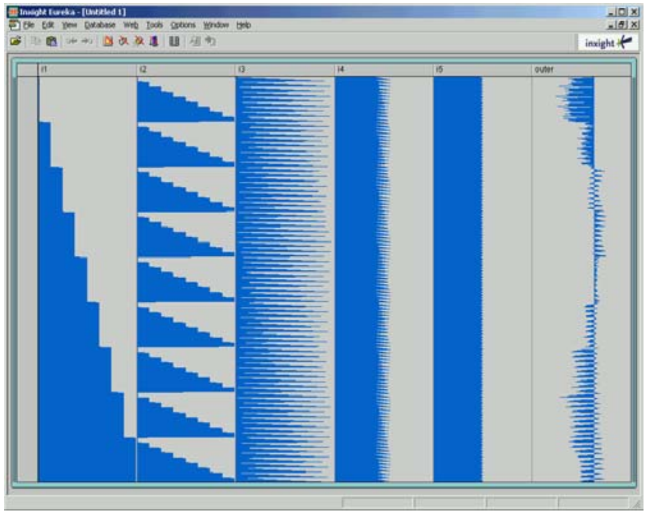
Inxight Eureka - [Untitled 1]
File Edit View Database Web Tools Options Window Help
inxight
i1
i2
i3
i4
i5
outer

Figure 1: TableLens visualization of a 5-dimensional function with 9 samples per dimension. Columns 1-5 represent independent variables, column 6 represents the dependent variable.

For visualizing functions, it is more efficient to map independent variables to spatial attributes, without separate visual marks for each independent variable value. Our approach looks similar to the Star Coordinates radial arrangement of axes, but does not use the vector summation concept.

3 POLAREYEZ VISUALIZATION

PolarEyez is a new focus+context visualization for multidimensional functions that uniformly integrates all dimensions into a single view. The visualization arranges the dimensions in radial fashion around a polar focus point (Figure 2), and maps function value to a color scale at every point in the space. For 2-dimensional functions, it is identical to a heat map. However, instead of adding views of 2D slices, additional dimensions are added directly into the radial arrangement. This provides an integrated overview of the space from the perspective of a navigable focus point within the space.

The PolarEyez approach conceptually supports any number of dimensions and arbitrary range bounds on each dimension. For simplicity, the following mappings will be described for 3D functions defined within a bounding cube. The concepts scale up naturally, and examples with greater dimensionality will be shown later.

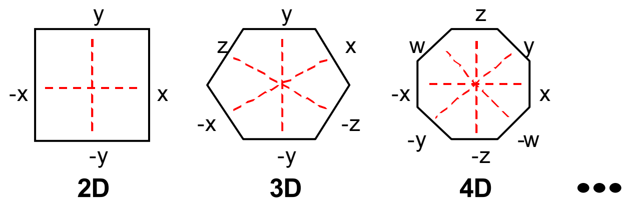

Figure 2: Basic spatial arrangement in PolarEyez with increasing number of dimensions.

3.1 Conceptual Mapping

Focal point: First, a focal point is chosen within the multidimensional function space. Initially, the focal point is selected as the center of the bounding cube. The function value is sampled at this point, mapped to a color scale, and displayed as the focal point in the center of the visualization. Our current color scale (Figure 3) ranges from bright red for the most negative value, to black for zero value, to bright green for greatest positive value. White is used for undefined values and regions outside the bounding cube.

Rays: Then, many rays that emanate from the focal point in all directions and extend to the cube boundary are chosen in the function space. These rays are then arranged radially around the focal point in the visualization (Figure 4). Essentially, rays in the multidimensional space are rotated around the focal point onto a 2D plane of the visualization. Function values are sampled along the rays in multidimensional space, mapped to the color scale, and displayed along the rays in the visualization. A uniform distribution of rays is chosen, such that their destination points on the bounding cube forms a grid-like pattern. Enough rays are chosen such that the visualization is completely filled with colored rays. Hence, each pixel on the perimeter of the visualization is the destination point of some ray emanating from the focal point.

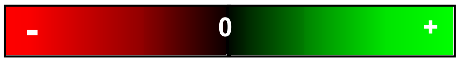

Figure 3: Approximate color scale for mapping function values.

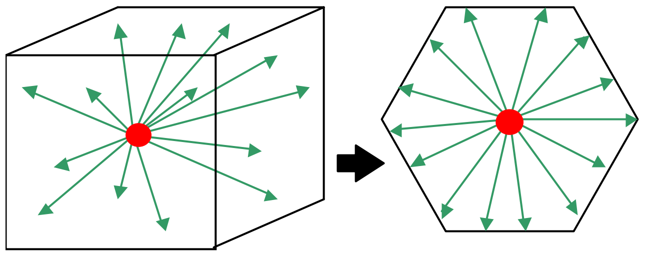

Figure 4: Focal point with rays in all directions map from function space to visualization space.

A major challenge of this approach is to organize and order the rays around the focal point in the visualization. Two heuristics are used.

Faces: The first heuristic groups rays according to the face of the bounding cube that they intersect. Hence, rays are grouped with their rotationally nearest primary axis. Each group of rays defines a pyramid whose base is a face of the cube, and whose pinnacle point is the focal point. Each group is represented in the visualization as a triangular pie slice (Figure 5a). The exterior edge of the pie slice is flat to represent the corresponding face of the cube. In the visualization, this forms a hexagon (for 3D functions). Each face of the hexagon represents a face of the cube. The pie slices and their faces are identified by the primary dimension that they bound. The two pie slices representing the positive and negative directions of a single dimension (e.g. +x, and –x) are displayed on opposite sides of the focal point in the visualization. Hence, lines formed by opposite rays in multidimensional space are preserved in visualization space.

Paths: The second heuristic orders rays within each pie slice according to their destination points on the cube face. A path is defined on the cube face that linearly orders the ray destination points along the hexagon face. There are many possible alternatives for this path. Our current approach uses a straightforward scan-line path across the cube face (Figure 5b). The ray at the center of the cube face, and parallel to the primary dimension of the face, is located in the center of the pie slice. A spiral path, a potential alternative, orders rays according to increasing angle from the primary dimension of the face. Note that in the general case a face of a *d*-dimensional hyper-cube is itself a *(d-1)*-dimensional hyper-cube.

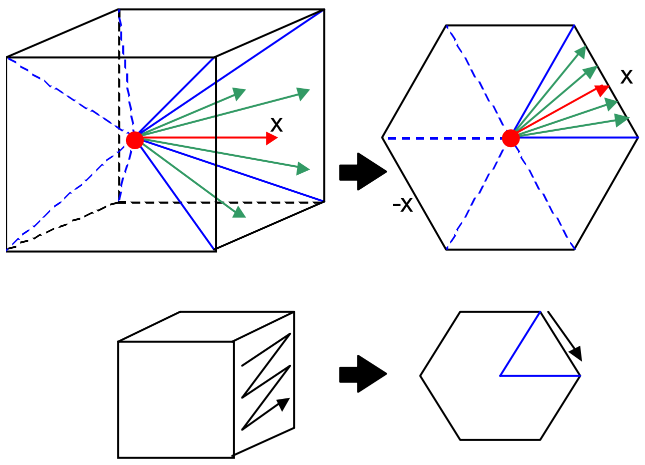

Figure 5: Rays are (a) grouped by face, and (b) ordered within a face by a scan-line path.

Figure 6 shows an example visualization of a 3D function. The cyclical nature of the function and the high and low spots can be seen. The most prominent cycles are on the diagonals where all parameters have near equal absolute values. The radius is approximately one cycle.

To generalize beyond 3D functions, the mappings from multidimensional function space to 2D screen space are summarized as (with colored reference to Figure 5):

- Focal point maps to focal point (in Figure 5, red).
- Ray maps to ray (green).
- Hyper-cube maps to a polygon with equal sides and angles (black).
- Hyper-cube face maps to polygon face (black).
- Hyper-pyramid maps to pie slice (blue).
- Function value maps to color.

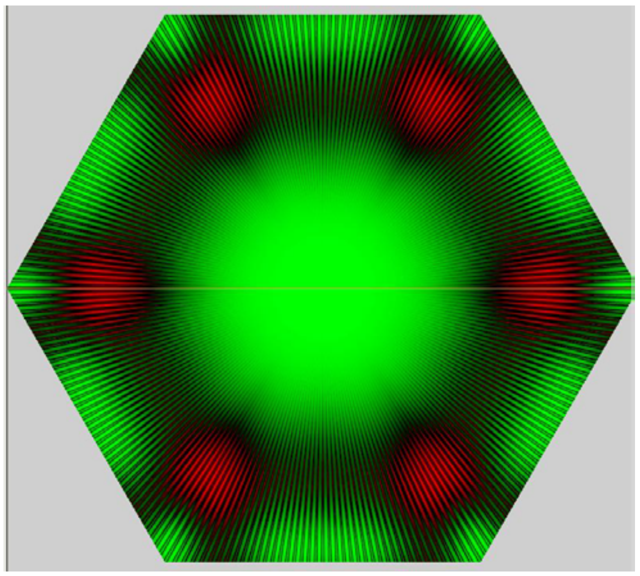

Figure 6: Visualization of *$\cos(x^2) + \cos(y^2) + \cos(z^2)$* with focal point at the origin.

3.2 Aggregation and Focus+Context

Since function data may exist between the rays in the functional space, it is necessary to aggregate this data onto the nearest ray so that all data is represented in the visualization. That is, each ray aggregates (e.g. averages) data around it. Hence, each ray in function space is actually a narrow pyramid. Together, these narrow pyramids completely fill the entire cube. In visualization space, each ray is actually a narrow triangular pie slice.

This aggregation, combined with the radial layout, creates the smooth focus+context [Spe01] effect. This enables users to view details of a localized region without losing the context of the overall functional space. Less aggregation (more detail) occurs near the focal point, and more aggregation (less detail) occurs distant from the focal point. Furthermore, aggregation increases smoothly with increase in distance from the focus, creating a smooth transition between detail and context.

Proof of this concept is helpful. Intuitively, the surface of a cube of width *w* in function space is visually encoded on the perimeter of a hexagon of width *w* in 2D visualization space (technically, a linear mapping on the width occurs to map distance in function space to distance in screen space). As width *w* increases, the surface area of the cube increases more rapidly than the perimeter length of the hexagon. Hence, as width *w* increases (and therefore distance from the center focal point increases), more area in the function space must be aggregated and encoded into relatively less perimeter in visualization space (Figure 7).

Mathematically, the surface area of a cube of width *w* is $6w^2$. In general, the hyper-surface area of a *d*-dimensional hyper-cube is the total volume of its $2*d$ $(d-1)$-dimensional hyper-faces, or $2dw^{d-1}$. The perimeter length of a polygon of width *w* can be approximated by the circumference of a circle of diameter *w*, which is πw. Hence, the aggregation factor, which measures the amount of data encoded per unit screen space, is a function of *w*:

$$aggregation = 2dw^{d-1} / \pi w \approx dw^{d-2}$$

Therefore, for greater than 2 dimensions, aggregation increases as a function of the distance from the focal point. With many dimensions, aggregation increases rapidly with *w*. Note that for $d=2$, aggregation is constant (e.g. a heat map).

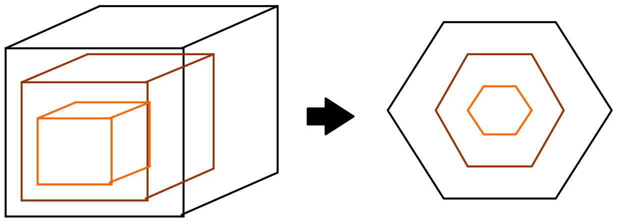

Figure 7: Surfaces of concentric cubes map to perimeters of concentric polygons. Outer cubes require more aggregation.

3.3 Interactive Exploration

As users move the mouse over the visualization, a tooltip shows the coordinates of the point in multidimensional space and the function value.

Users can navigate the visualization's focal point within the multidimensional function space by either directly selecting a point in the visualization or by entering the desired coordinates. The visualization places the new focal point at the center, and reorients the polar overview from the perspective of the new focal point. This enables users to view more details of a desired region using the focus+context technique. Navigating the focal point nearer to the edge of the bounding cube will cause some rays to be shorter than others. Rays emanating toward a nearer face will naturally intersect the bounding cube sooner, and causes portions of some pie slices to appear truncated. This helps users recognize nearness to the boundaries, and orient themselves within the space.

Users can also filter out uninteresting pie slices from the overview. This provides more detail to pie slices of interest by evenly distributing extra angular screen space to these slices. This enables users to focus on interesting regions, or eliminate independent variables that have little effect on function behavior. This feature could be extended to enable users to independently control the angular space devoted to each pie slice by directly resizing them.

Users can issue simple queries to highlight points that meet a desired criterion in bright yellow. Users can specify ranges for dependent and independent variables to highlight. For example, this helps users to analyze whether desired functional values occur in clusters or are spread throughout the space.

4 EXAMPLES

4.1 Mathematical Functions

Figure 8 shows the visualization of a function used in Mihalisin [MTS91], $x^2 + u^2 + y^2 + {}^2v$. The integrated approach used by PolarEyez demonstrates the pattern in a simpler fashion than slicing.

Figure 9 shows the visualization of the function:

$$y = \sum cos(x_i^2)$$

with 4 and 12 dimensions, and focal point at the origin. The function is clearly cyclical and symmetric. The bright green region in the center indicates strong spike in the function where all variables are near 0. Smaller regions of high and low values are distributed. The bounds of the space is approximately 15 cycles wide. The spoke pattern indicates the prominence of the cyclical pattern near the primary axes. The decreasing period of the ripple pattern proceeding outward indicates the effect of the exponent within the cosine term. Figure 9c shows the change in the pattern when navigating to $x_1=2.5$.

4.2 Engineering Data

To study potential engineering designs for an aircraft control subsystem, researchers developed a mathematical model of the phenomenon. The design had 5 primary parameters (labeled x_1 through x_5) that affected system performance. A complex neural network simulation was implemented and trained. Sampled performance data was derived throughout the 5-dimensional space. Visualization of the simulation output helps in several design tasks. In general, target regions of high performance are desired. However, system stability is a priority due to potential error in simulation and system operation. Furthermore, to guide continued experimentation and aid in model verification, general understanding of simulation output is needed.

Figure 10a shows the initial visualization of the simulation output. Parameter x_5 is critical to system performance, and negative values of this parameter results in almost certain poor performance (red). Parameters x_3 and x_5 both have unusual regions of uniformly poor performance in their negative sides. This might indicate a problem in the simulation for further study. x_1 and x_2 are fairly symmetric. Small hot regions of good or poor performance are scattered throughout the other dimensions. The current focus region is a candidate for fairly stable good performance, with green surroundings for a reasonable margin. However, the green area near the extreme negative region of x_3 also appears stable, and mousing over the region indicates higher performance values.

To verify this potential stability, the query highlight feature is used. The user queries a narrow range for performance value that straddles an observed value in that area. All points meeting that criteria are automatically highlighted in bright yellow (Figure 10b). The solid yellow region is quite stable, with the exception of some holes near the $x3$ axis itself. The existence of these holes is strengthened by the fact that the yellow on the other dimensions is away from the origin, and hence not near the $x3$ axis. Negative values for x_1, x_2, x_4 appear best. Navigating to that region does reveal a stable high performance area (Figure 10c).

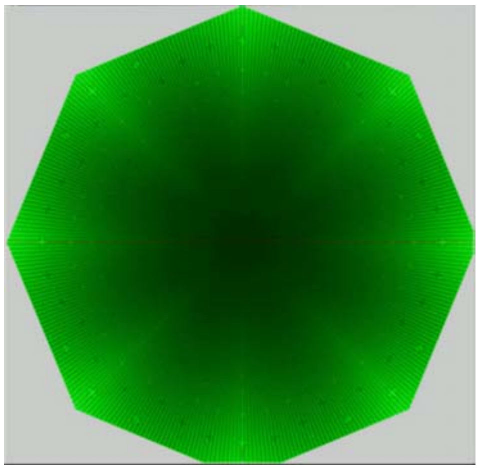

Figure 8: Visualization of $x^2 + u^2 + y^2 + {}^2v$.

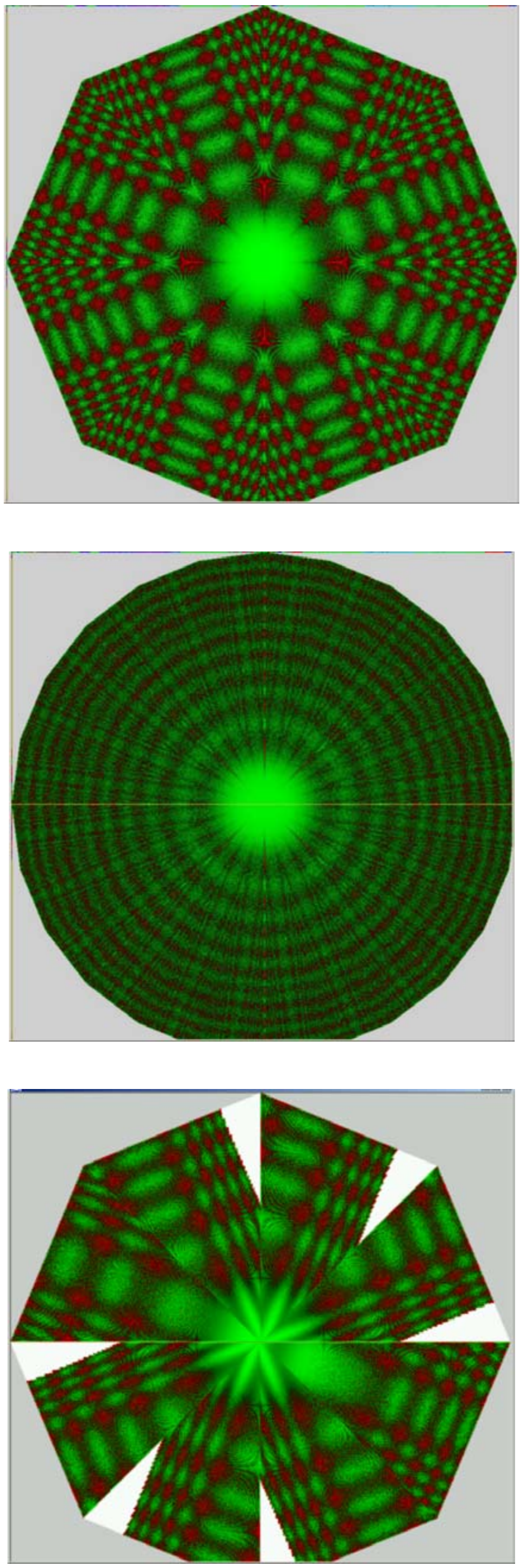

Figure 9: Visualizing $y = \sum cos(x_i^2)$.
(a) 4D, (b) 12D, (c) 4D navigating to $x_1^2=2\pi$.

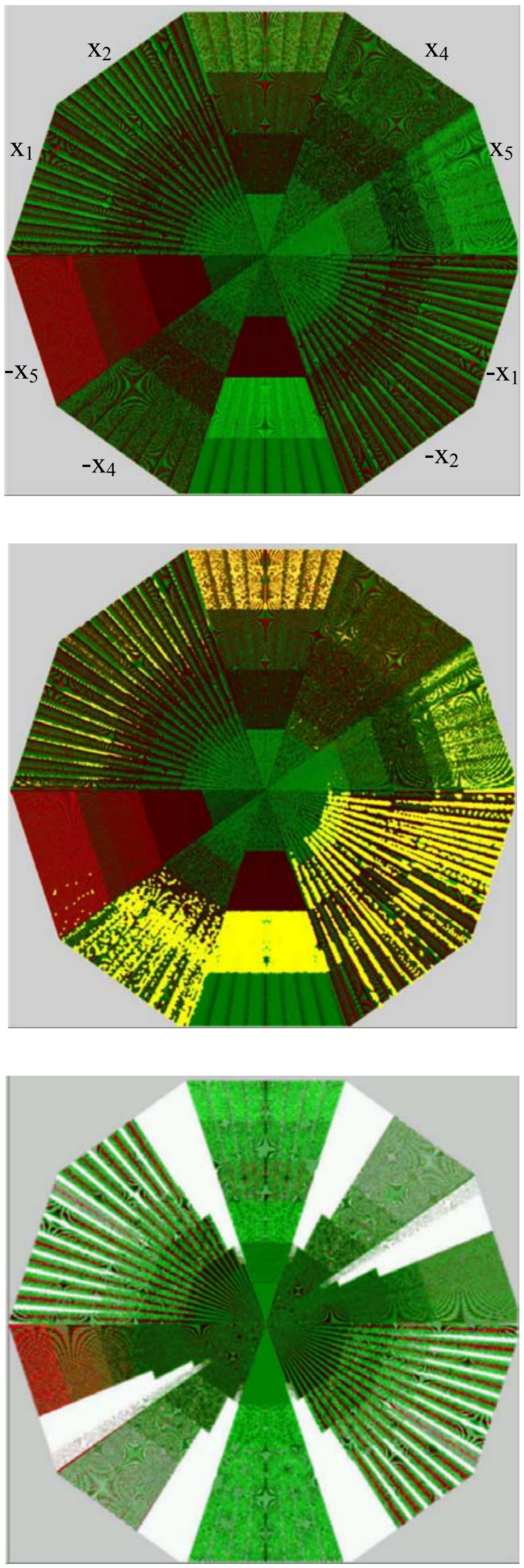

Figure 10: Visualizing 5 dimensional engineering data.
(a) initial overview, (b) query highlight, (c) navigation.

5 ALGORITHM

The algorithm to implement the PolarEyez conceptual layout is complex. This is because it is difficult to display polar concepts on Cartesian rectangular pixels. Two different algorithms were implemented:

- Map function to screen: This approach computes the rays in multidimensional function space, and then draws each ray on the screen. The challenge of this approach is handling the anti-aliasing and overlapping of the rays on the screen.
- Map screen to function: This approach reverses the process. For each pixel, it computes the multi-dimensional space represented by that pixel, and colors the pixel accordingly. The challenge of this approach is calculating the space represented by each pixel.

The Screen-to-Function Algorithm

```
For each pixel p(x,y):
1. Map pixel p to region r in function space:
   a. Map pixel p to point pt in function space:
        i. Determine pie slice s containing pixel p using
           angles w.r.t. focal point
       ii. Determine concentric polygon face containing
           p using radius w.r.t. focal point
      iii. Determine concentric cube face represented
           by polygon face using radius and slice s
       iv. Calculate point pt of pixel p on cube face
           using scan-line path:
           • Calculate fraction of distance of p along
             polygon face, using angles
           • Map to distance along path on cube face
           • Follow distance on cube face to locate pt
   b. Approximate region r around point pt:
        i. Map pixel width to fraction of path length
           using approximate number of pixels on
           polygon face
       ii. Include that fraction of path around pt in r
      iii. Map pixel width to radial depth of cube face
           using approximate number of pixels on pie
           radius and dimension range.
       iv. Include that depth in r
2. Compute aggregate function value v for region r:
   a. If sampled array data: average the samples
      contained in region or interpolate samples around
      region r
   b. If math formula: sample and average the center
      and corners of region r
3. Map value v to color c using color scale
4. Plot color c in pixel p
5. Repeat for mirror pixel p2 on opposite side of focal
   point, reusing pt and r calculations from step (1)
   where possible
```

Figure 11: The algorithm for generating PolarEyez visualizations using the screen-to-function variation.

We currently prefer the latter approach because it handles each pixel only once. The algorithm is presented in Figure 11. When the function is specified using the mathematical formula method, aggregation is difficult and requires mathematical integration. The algorithm approximates this with some sub-sampling. However, in many dimensions, significant portions of the space are missed which causes the visualization to appear noisy or choppy. A better algorithm is needed for aggregating functional spaces.

5.1 Implementation

The visualization is implemented in Java. Users can provide data in either form, mathematical formula or sampled data array, and specify range bounds for each dimension. Inputting approximate minimum and maximum function values enables color mapping without requiring a second pass in the algorithm to calculate these values, and enables some customization of color map bounds.

On a <1 GHz Pentium PC, the visualization takes a few seconds to generate. A mirroring technique is used to reduce redundant calculations and speed up the visualization. Specifically, computations for a pixel such as angle and radius are reused for the symmetric pixel in the diametrically opposite pie slice.

6 DISCUSSION

A major advantage of this visualization approach is that it presents an integrated overview of the entire functional space on all dimensions simultaneously. The overview helps users identify regions of high or low function values, frequency of particular values, clusters, etc. This relieves users from mentally integrating separate 2D slices in short-term memory as required by previous approaches. The polar nature of the overview enables users to view patterns in function value proceeding away from the focal point in all directions, and estimate distances to interesting phenomena. Representing the overview from the perspective of a point in the space is somewhat egocentric and natural for users to grasp, as confirmed by informal feedback from test users. Interestingly, the flat edges of the polygon pie slices are helpful for orienting users, and provide a natural analogy to the squares and cubes of heat maps. We had previously explored a completely circular layout, but caused confusion for users.

Another advantage is scalability. This approach scales up smoothly. Each additional dimension simply adds two more slices to the pie, gradually narrowing all slices. It scales to approximately 10 to 20 dimensions (20 to 40 pie slices). Beyond that, very narrow pie slices can still provide some useful information. For 2D functions, this approach reduces cleanly to a simple heat map. Furthermore, the number of dimensions *d* scales independently of the bounding range *r* of the dimensions. This is because dimensions and range are mapped to separable dimensions in the visualization. Dimensions are represented circumferentially, and dimension range is represented radially. The range for each dimension receives a screen-width (e.g. 1000 pixels) of space regardless of number of dimensions.

7 FUTURE WORK

While the focus+context approach provides a detail view within the overview, a more detailed view is needed. We have explored a radical variation of the visualization in which the focal point is stretched out into a horizontal line, and the rays are organized along the line pointing vertically (Figure 12). This significantly expanded the detail of the focus region, at the expense of the extremities. However, users were too confused by this mapping, and could not overcome the belief that it was depicting 2D slices. A more natural approach might be to radially expand the focus in a fisheye-like manner [Kea98]. Hence, the radius could have a non-linear mapping from function to visualization. Alternatively, 2D slices are excellent detail views for examining specific correlations. It would be interesting to explore the use of PolarEyez as an overview for controlling separate 2D slices in an overview+detail fashion [CMS99].

Additional work is needed to explore improved aggregation algorithms, alternate ray arrangement heuristics, more efficient algorithms for real-time navigation, and support for multi-valued functions. Finally, while informal user feedback has been positive and has guided the design, formal user studies are needed to rigorously evaluate this approach.

8 CONCLUSIONS

This paper contributes a novel layout and navigation strategy for visualizing multidimensional functions. A polar focal point with rays emanating in all directions compresses the multidimensional space into a circular layout. This approach has several key characteristics:

- Provides an integrated overview of the entire bounded function space, from the natural perspective of a point within the space.
- Treats all dimensions uniformly, without employing conventional slicing schemes, enabling visualization of variation in all dimensions simultaneously.
- Provides focus+context, with smooth seamless transition from detail to overview.
- Smoothly scales up to 10-20 dimensions, with approximately 1000 pixel range on each dimension.

9 ACKNOWLEDGEMENTS

Thanks to B.F. Goodrich Inc., Black&Decker Inc., Vijay Varadarajan, and Srinivasan Vasudevan for data and valuable feedback.

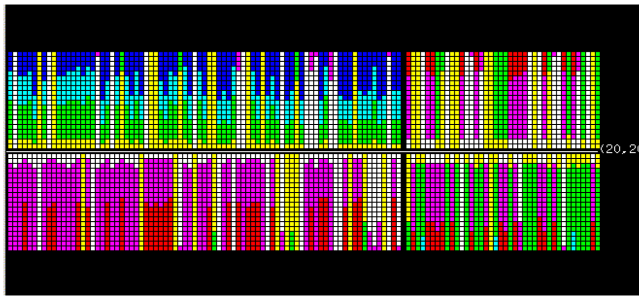

Figure 12: Stretching the focal point into a horizontal line, with rays extending vertically, to exaggerate the focus+context effect.

10 REFERENCES

[AW95] Ahlberg, C., Wistrand, E., "IVEE: An Information Visualization and Exploration Environment", *Proc. IEEE Information Visualization Symposium '95*, pp. 66-73, (1995).

[CMS99] Card, S., Mackinlay, J., Shneiderman, B., *Information Visualization: Using Vision to Think*, Morgan Kaufmann, (1999).

[FB90] Feiner, S., Beshers, C., "Worlds within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds", *Proc. ACM UIST '90*, pp. 76-93, (1990)

[Ins97] Inselberg, A., "Multidimensional Detective", *Proc. IEEE Information Visualization Symposium'97,* pp. 100-107, (1997).

[Kan00] Kandogan, E., "Star Coordinates: a Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions", *LBHT Proc. IEEE Information Visualization Symposium 2000*, pp. 9-12, (2000).

[Kea98] Keahey, A., "The generalized detail-in-context problem", *Proc. IEEE Symposium on Information Visualization*, IEEE, (October 1998).

[MTS91] Mihalisin, T. Timlin, J., and Schegeler, J., "Visualizing multivariate Functions, Data, and Distributions", *IEEE Computer Graphics and Applications*, 11(13), pp. 28-35, (1991).

[RC94] Rao, R., Card, S., "Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information" *Proc. ACM CHI'94*, pp. 318-322, (1994).

[Spe01] Spence, R., *Information Visualization*, Addison-Wesley, (2001).

[TSD96] Tweedie, L., Spence, R., Dawkes, H., Su. H., "Externalizing Abstract Mathematical Models", *Proc ACM CHI'96*, pp. 406-402, (1996).

[VV93] Van Wijk, J., Van Liere, R., "HyperSlice: Visualization of Scalar Functions of Many Variables", *Proc. IEEE Visualization '93*, pp. 119-125, (1993).