Editor: Theresa-Marie Rhyne

Visualizing Multiple Fields on the Same Surface

Russell Taylor

University of North Carolina at Chapel Hill

A long-standing problem in the field of visualization is the ability to display many overlaid data sets. Recently, researchers have developed successful techniques that show multiple 3D fields by layering sparse similar glyphs.

My colleagues and I at the University of North Carolina have investigated this problem for a number of years. Our driving application has been the display of data sets from scanned-probe and scanned-electron microscopes, which led us to concentrate on the display of multiple scalar fields. Although several of our first approaches were initially successful, they failed to scale to more than three or four data sets. So that others won't be tempted to follow the same course, I begin by presenting the argument for, preliminary successes with, and reasons for the eventual downfall of our initial approach. I then describe the key concept that enables successful techniques.

Our initial approach

Our initial (incorrect) thesis was that the largest number of data sets could be displayed using the following methods (which are obviously not orthogonal):

- map each data set to a different surface characteristic (color, shape, albedo, and transparency),
- apply different visualization techniques for each data set (color, contour lines, blending in textures, and so on), and/or
- generate textures or glyphs whose different characteristics (scale, feature shape, color, density, and so on) depend on the data sets.

Because we had a height field from the scanned-probe microscopy data to start with, we used apparent surface height to show this measured height and investigated techniques to display additional fields on this height field. (More on this matter later.)

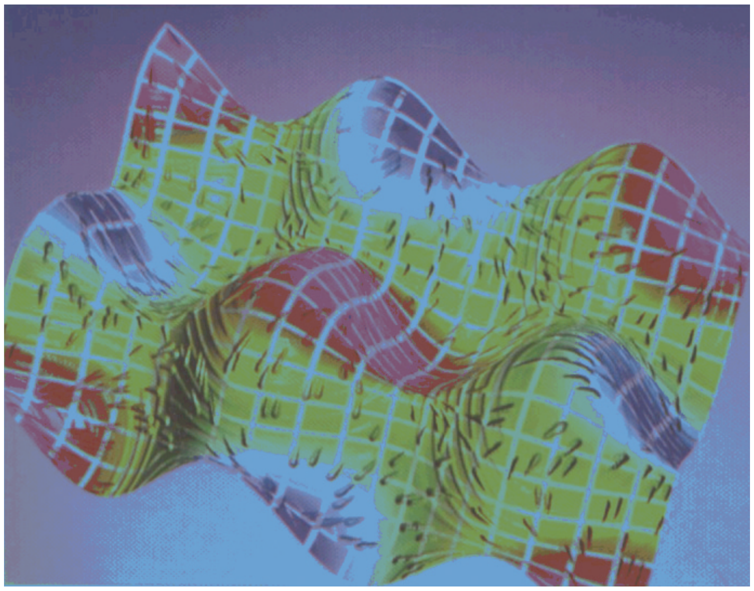
Promising signposts leading nowhere

We were encouraged to modify different surface characteristics and use different techniques by our early successes in combining small numbers of these techniques.

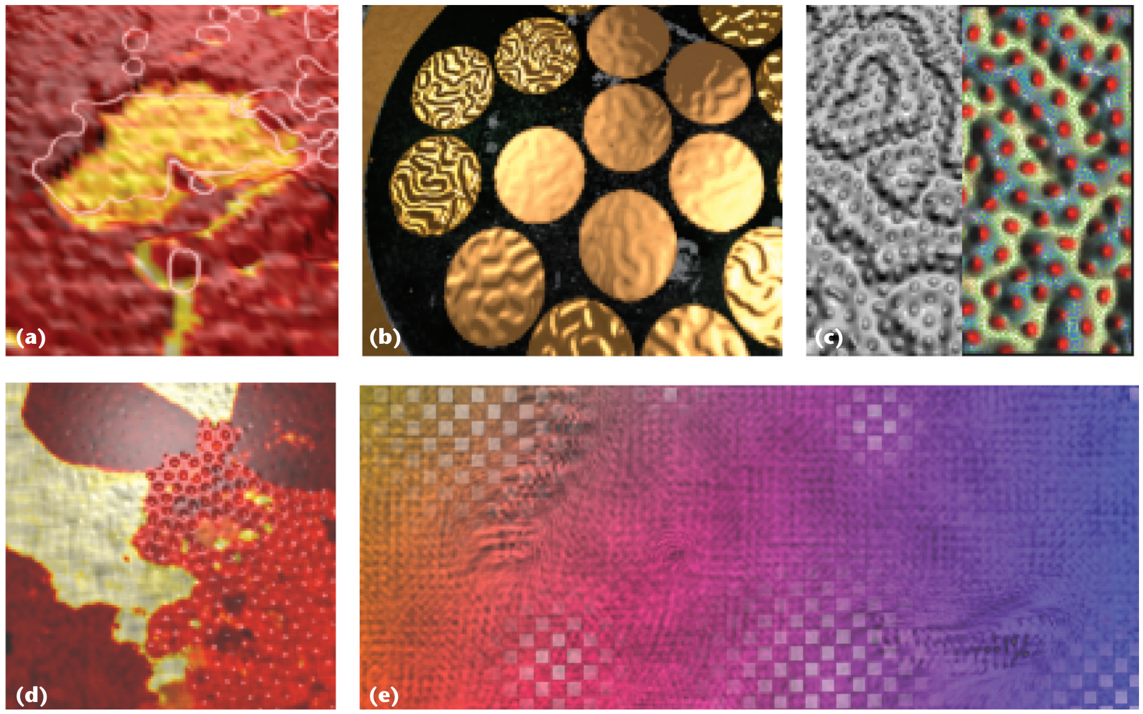
The discovery of Roger Crawfis' beautiful visualization of multiple data sets[1] encouraged us to try layering different techniques. This visualization (see Figure 1) shows several scalar fields and a vector field displayed on a single surface.

We were led to texture synthesis by the data-driven synthesis of reaction-diffusion[2] and spot-noise textures.[3] These techniques enable the adjustment of texture scale, feature shapes, and colors by coupling the parameters of the generating functions to data sets. We were further encouraged by the perceptually based texture synthesis techniques of Healey and colleagues.[4]

In the end, we were disappointed to find that the original thesis was incorrect. No matter which sets of techniques we tried, no matter which sets of surface characteristics we varied, no matter which group of texture characteristics we used, we couldn't display more than three or four data sets before it became impossible to separate the effects of one technique from the effects of the others.


Courtesy of Roger Crawfis

1 Visualization showing several scalar fields and a vector field displayed on a single surface.



2 Some of our attempts to visualize multiple data sets by applying a different technique to each data set. This worked only up to three or four data sets.

Wandering in the wilderness

We were tempted to display multiple data sets using a different technique for each. Having access to the University of North Carolina-designed PixelFlow graphics computer let us rapidly explore per-pixel shading of surfaces using Renderman-like shaders,[5] including online data-driven texture generation. We explored many combinations of the following techniques:

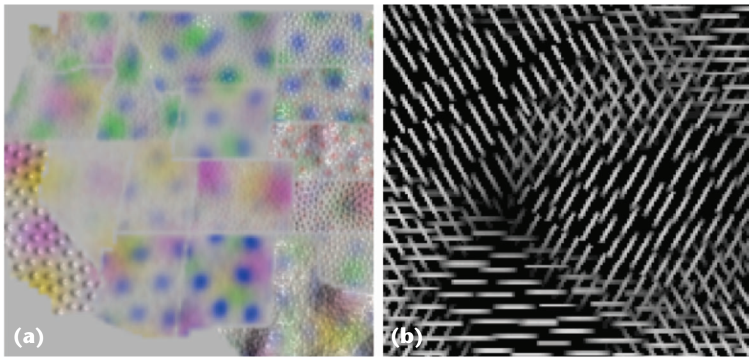
- color,
- transparency (which failed for the reasons outlined by Interrante and Rheingans[6,7]),
- contour lines,
- surface albedo (reflectance function variations),
- textures that appeared more pronounced where a data set was higher,
- bump-mapped textures whose height was modulated by data sets,
- spot noise texture generation using kernels with data-dependent characteristics,
- reaction-diffusion texture generation using data-dependent parameters for their generating functions, and
- apparent surface height (which we always mapped to measured height).

Figure 2 shows images produced during our attempts to combine multiple techniques. Along each line of investigation, the initial results were positive. We could combine two or three visualization techniques with good effect (barring unfortunate choices of parameter settings). Figure 2a shows the combination of height field, hue, and contour lines to show surface height, friction, and areas of high adhesion. Figure 2b shows a bump map and surface shininess to display the amount of gold and copper present in each strand of a cable. Figure 2c shows the beginnings of trouble: the two bump-map textures start to obscure one another until we add color to make it easy to distinguish between them. Figure 2d is more problematic: color, the presence of a cross-hatch texture, and the presence of dimples show three data sets. The presence of the dimples makes it difficult to determine the texture's strength. Figure 2e also has trouble: hue, the presence of a checkerboard texture, and the kernel orientation in a spot-noise texture each display different fields. Where the checkerboard pattern is prevalent, it's difficult to determine the orientation of the texture underlay.

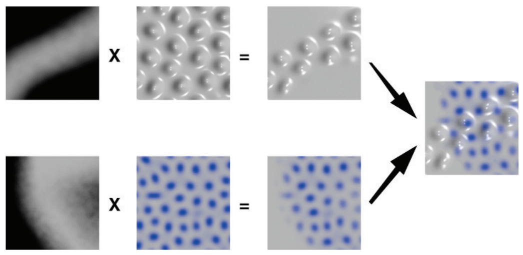
We tried different parameter settings, different layering orders, different combinations of techniques, perceptually linear mappings for the attributes, and different couplings between data sets and techniques—all to no avail. Each time, the visualizations failed to work when more than a few techniques were applied at once. In each case, the problem was masking between the different techniques. High spatial frequencies in one technique masked the values in other techniques. Techniques that were layers on top either masked or scrambled layers below.

A new approach

We achieved a comprehensible display of the largest number of simultaneous scalar fields by using several overlapping instances of the same technique. Two students in the group came up with the most successful techniques. Alexandra Bokinsky proposed using sets of spots and bumps of different colors and scales to encode different data sets. Chris Weigle proposed using sets of slivers with different orientations to encode different data sets (see Figure 3, next page). I was skeptical in each case—the ideas didn't fit the original thesis—but each technique proved effective.



3 **Data-Driven Spots (a) uses spots of a specific size and color for each data type. Oriented Slivers (b) uses slivers of a certain orientation for each data type. The spots or slivers showing each data set are distinguishable yet do not obscure those showing other data sets.**



4 **In Data-Driven Spots, each layer of spots is modulated by the presence of a data set and then combined with other layers.**

Data-Driven Spots

Data-Driven Spots (DDS) uses the perceptual channels of color, scale, and motion (which could be used to display data) to produce multiple visually separable texture layers. Each layer is composed of a single type of sparse, distinct glyphs that display one scalar field—spots by their intensity and bumps by their height. Figure 4 shows how we combined two data sets into a single image. Figure 3a shows the display of several census data sets overlaid on a map of the western United States. The results of user studies showing the effectiveness of this technique and a description of the power of combining animation with DDS will be described in Bokinsky's dissertation (expected to be completed in August 2002), so I won't spoil the surprise here.

Oriented Slivers

Figure 5 shows the method of creation for each layer of slivers in a multilayer presentation. The multiple layers are blended, producing an image displaying overlapped data sets (Figure 3b). Thin, well-separated slivers let multiple orientations show through in each area. The resulting image lets us estimate the value of each data set at a given point on the surface. Weigle describes this technique, as well as the results of user studies to determine the number of orientations to use, elsewhere.[8] We can use hue either to label one or more sliver orientations (to bring them to the viewer's attention) or as in Figure 6 to display another data set in the background at constant luminance (this also sharpens the judgment of intensity levels for the mid-range and dark regions of slivers).

The key idea

In traditional data visualization techniques, we vary each attribute (hue, intensity, orientation, texture, and so on) across its range of values to indicate variation in a data set. Hue is used this way in Figure 6 and both height and hue are used this way in Figure 1. Sometimes, we can vary more than one attribute to show the same data set (height and color may both be mapped from the same scalar field, for example). Both DDS and Oriented Slivers take a new approach. Oriented Slivers uses the orientation attribute not to encode data but to differentiate between sets of slivers. The intensity of each sliver encodes the value of a single field at its location. DDS uses color and scale not to encode data but to differentiate between sets of dots. The intensity (for each dot) or apparent height (for each bump) encodes the value of its associated data set at that location. Here's the general principle: one or more attributes differentiate between layers of sparse, similar glyphs and each layer displays a single field by varying other attributes. There's a switch from layering different techniques to layering distinguishable instances of the same or similar techniques. More formally, the algorithm is

1. Using one or more attributes, producing a nominal code for a set of glyphs.
2. Sparsely placing glyphs within a layer so that it's possible to see through to other layers. Avoiding regularity in placement to reduce aliasing.
3. Adjusting new attributes of each glyph in a layer (attributes not used in the nominal code) to display a single data field's value at the glyph's location.

Kirby and Laidlaw developed one of the most successful visualizations of multiple fields on a surface that I've seen.[9] Figure 6 shows the image from their paper that displays the largest number of data sets. Their image follows this algorithm (layered on top of a hue field). It uses what are often data-encoding dimensions (basic figure shape and saturation) to separate data display into layers of glyphs and then uses other data-encoding dimensions (anisotropic distortion and scale, orientation) to convey the data values within each layer. Here, the sparseness of top layers is violated in one layer—it's difficult to see past the black blocks near the left edges of the cylinder. This is probably due to the decreased image resolution and image size in this reprinting of the figure.

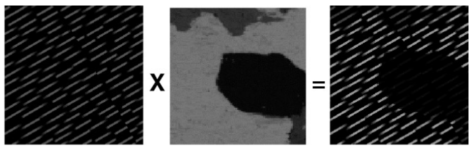
Figure 1 also follows this algorithm (layered on top of a height field and a hue field): the dimensions of figure shape (a droplet and a ripple) separate layers while orientation encodes the data values within each field.

Combining with other techniques

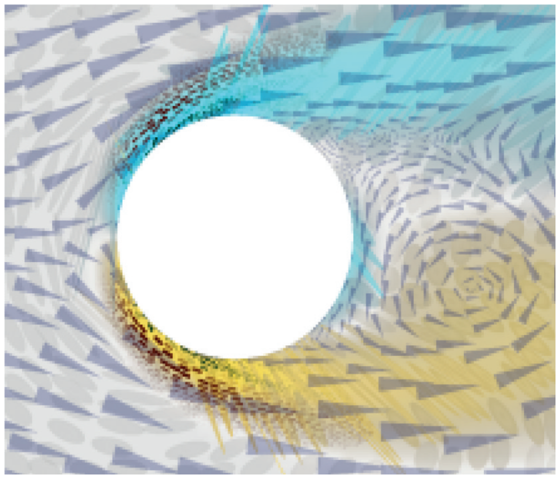
As I mentioned before, it's possible to include a varying-hue backdrop layer to display another scalar field behind the glyph layers. This isn't possible when hue is one of the attributes used to differentiate between glyph layers. Care must be taken to avoid changing the apparent contrast between the background layer and glyphs if glyph intensity is a data-carrying attribute.

The display of height seems to combine well with many techniques, forming the manifold upon which they're displayed. This is true for the layering techniques advocated here. Of course, dangers exist. First, the diffuse and specular reflections used to indicate surface shape cause luminance changes that can completely hide or distort the perception of the glyphs' luminance. Second, if one or more glyphs distort the surface (bumps, for instance), the scale of the distortion due to glyphs must be kept far from the scale of height changes due to data display to avoid confusing the two. However, there's an upside. Adding a uniform texture to surface that's otherwise shown as a uniform-color height field can actually improve the perception of the surface's shape—adding more data display to a surface might make it easier to see the surface. Ware discusses this at some length in his book *Information Visualization: Perception for Design*.[10]

Our experiences showed that the attributes of one visualization technique become confounded with those of other techniques that also have high spatial-frequency components. I strongly believe that attempts to combine this technique with texture techniques, complex glyph techniques, or other techniques that include high-frequency components will fall prey to this. We must take great care to select glyph sets that don't confound one another, enabling effective layering.



5 In Oriented Slivers, each layer of slivers is modulated by the presence of a data set before being combined with other layers.



6 Kirby and Laidlaw used ideas gleaned from studying artists to develop this visualization showing two scalar fields, two vector fields, and a tensor field in the same image.

Closing thoughts

I've not mentioned the most obvious methods of presentation for multiple data sets: side-by-side display of multiple images and sequential-in-time display of each data set. Figure 2a gives the case for displaying the data in the same image (rather than side by side images): although the friction and adhesion maps are of a similar shape, they're somewhat distorted and offset with respect to each other. In looking at side-by-side images, this slight discrepancy might go unnoticed (and noticing slight discrepancies in expected results has provoked many a scientific insight). The case for layering versus time-sequential display of multiple fields is harder to make. Displaying nine scalar fields pairwise requires 36 transitions, so a user would have to watch the display for up to a minute to see all comparisons using two-second fades between data sets—the viewer of a layered image can switch to the desired comparisons at will. Also, layering techniques can be used with time-varying data (which we get from our microscopes as they scan). With layers, but with neither spatially parallel nor time-sequential display, all of the data sets are laid out at once in the same image, causing areas where lots of data sets are high or low to stand out. Of course, which of these is better depends on the task at hand. We aim for exploratory qualitative visualization of data sets to let users quickly verify the expected and notice the unexpected.

There's a spatial trade-off when using sparse glyphs—they don't cover the entire surface and so can miss features of interest. Note that the data can cut off the glyphs so that boundaries are visible (the Nevada state boundary in Figure 3a shows this). We can address this undersampling in at least three ways. First, we can increase the display resolution and size so that the glyphs are denser with respect to the underlying data set. Second, users can zoom into the data set while maintaining fixed glyph size and spacing (looking at less of the data but at higher resolution). Finally, animating the glyphs by placing them all in uniform linear motion across the surface causes each area of the surface to be sampled over time by some glyph in each layer (this would probably be confusing if vector fields are displayed, but it's effective for scalar fields).

Note that I judge tensor, vector, and scalar fields to be incomparable. To visualize a vector field, you want an integral pair of parameters that shows direction and magnitude so that users can easily perceive the flow. To visualize two independent scalar fields, you want a separable pair of parameters that lets users judge the value of each field (and possibly compare the relative magnitudes of the two). Because of this qualitative difference, I don't count vector fields as equivalent to two scalar fields or tensor fields as equivalent to more than two scalar fields, nor do I count them the same as one scalar field—they're simply different.

Our efforts to develop techniques for displaying many data sets on a surface have led me to the conclusion that the key lies in producing layers of sparse, discrete glyphs with each layer displaying one field. This was the breakthrough realization that underlies the two successful techniques developed by our group, and it matches successful results shown by others. If the glyphs are similar enough between layers, it seems easier to estimate the relative magnitudes of different data sets (this is especially true of Oriented Slivers). If the glyphs in each layer vary in the same manner (Oriented Slivers and DDS), users only have to learn one interpretation strategy to decode all the data sets. If the glyphs in different layers are sufficiently different (all techniques but Oriented Slivers), it becomes easy to see the boundaries of regions covered by each data set. If the data sets include vector or tensor fields along with scalar fields, then more complicated glyphs are required. In this case, we must carefully select glyphs for different layers that are distinguishable from each other and yet are able to show variation compared to glyphs in their own layer. ■

Acknowledgments

The work described here was performed by graduate students in computer science at the University of North Carolina at Chapel Hill. In addition to Alexandra Bokinsky and Chris Weigle, the team included Aron Helser, Renee Maheshwari, Amy Henderson, Daniel Rohrer, Thomas Hudson, Adam Seeger, and Jeffrey Juliano. Chris Healey from NCSU and Mark Hollins from UNC Psychology provided valuable guidance and lively discussions. Herman Towles, Anselmo Lastra, and John Thomas kept PixelFlow running until the work was completed. The UNC nanoscale science team provided data sets and driving problems for us to solve.

The work presented here was made possible by an award by the National Science Foundation to support High Performance Computing and Communication (ASC-9527192) and by an award from the National Institutes of Health National Center for Research Resources (5-P41-RR02170).

Readers may contact Russell Taylor at the Department of Computer Science, CB#3175, Sitterson Hall, University of North Carolina, Chapel Hill, NC, 27599-3175, email taylorr@cs.unc.edu.

References

1. R.A. Crawfis and M.J. Allison, "A Scientific Visualization Synthesizer," *Proc. IEEE Visualization 91*, IEEE CS Press, Los Alamitos, Calif., 1991.
2. A. Witkin and M. Kass, "Reaction-Diffusion Textures," *Proc. Siggraph 91*, ACM Press, New York, 1991, pp. 299-308.
3. J.J. van Wijk, "Spot Noise, Texture Synthesis for Data Visualization," *Proc. Siggraph 91*, ACM Press, New York, 1991, pp. 309-318.
4. C.G. Healey and J.T. Enns, "Large Datasets at a Glance: Combining Textures and Colors in Scientific Visualization," *IEEE Trans. Visualization and Computer Graphics*, vol. 5, no. 2, Apr.–Jun. 1999, pp. 145-167.
5. M. Olano and A. Lastra, "A Shading Language on Graphics Hardware: The PixelFlow Shading System," *Proc. Siggraph 98*, ACM Press, New York, 1998, pp. 159-168.
6. V. Interrante, H. Fuchs, and S.M. Pizer, "Conveying the 3D Shape of Smoothly Curving Transparent Surfaces via Texture," *IEEE Trans. Visualization and Computer Graphics*, vol. 3, no. 2, Apr.–Jun. 1997, pp. 98-117.
7. P. Rheingans, "Opacity-Modulating Triangular Textures for Irregular Surfaces," *Proc. IEEE Visualization 96*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 219-225.
8. C. Weigle et al., "Oriented Texture Slivers: A Technique for Local Value Estimation of Multiple Scalar Fields," *Proc. Graphics Interface 2000*, 2000, pp. 163-170.
9. R.M. Kirby, H. Marmanis, and D.H. Laidlaw, "Visualizing Multivalued Data from 2D Incompressible Flows Using Concepts from Painting," *Proc. IEEE Visualization 99*, IEEE CS Press, Los Alamitos, Calif., 1999, pp. 333-340.
10. C. Ware, *Information Visualization: Perception for Design*, Morgan Kaufmann, San Francisco, 2000.

Readers may contact department editor Theresa-Marie Rhyne by email at tmrhyne@ncsu.edu.

Correction to May/June 2002 Visualization Viewpoints

Due to misunderstanding and rush, I made a serious mistake in my article "Visualizing Multiple Fields on the Same Surface" published in the May/June 2002 Visualization Viewpoints column. I published key elements of Alexandra Bokinsky's PhD dissertation without her knowledge or permission, and before she had a chance to publish them herself. I would like to apologize to Alexandra Bokinsky for my actions.

I would also like to ask that all future references to the Data-Driven Spots visualization technique by other authors point to her dissertation,[1] not to the Visualization Viewpoints column.

Sincerely,
Russell M. Taylor II

Reference

1. A. Bokinsky, *Data Driven Spots: A Layered Visualization Technique for the Interactive Display of Multiple 2D Scalar Variables*, PhD dissertation, UNC tech. report 02-033, UNC-Chapel Hill , 2002.