# DISPLAY OF FUNCTIONS OF THREE SPACE VARIABLES AND TIME USING SHADED POLYGONS AND SOUND*

W. M. COUGHRAN, JR.† AND ERIC GROSSE†

**Abstract.** This talk will describe the visualization tools used in our scientific computing group to look at data and functions in two and three space variables. Emphasis is given to aspects that differ from the prevailing style elsewhere, and the points made will be illustrated with a videotape of representative example of the tools in use.

Aside from a few inherently interactive tools such as brushing scatterplots and choosing viewpoints, we emphasize images recorded frame-at-a-time onto videotape. Sound works effectively for presenting scalar information in sync with field displays, for adding tick marks on the time axis, and for more subtle stretched data displays.

**1. tensor/scatter tools.** We have been involved in the construction of algorithms and software for simulating complex physical systems for many years. As simulations in two and three (or more) spatial dimensions and time become more commonplace, manipulating and understanding the results have become important aspects of the overall scientific-computing problem.

It is necessary to switch from a subroutine library mentality to making use of self-descriptive file formats when large simulation tools (run on specialized computer hardware) are involved. We introduced such file formats for tensor-product and scattered data elsewhere[5]; unlike specialized binary formats, our approach uses simple ASCII files that can be processed using an AWK-like paradigm[1]. In cases where speed of input/output becomes a serious issue, we accelerate the ASCII file using [10]. Such an approach has allowed us to build a family of simple tools that can be applied in numerous combinations using standard UNIX™ pipes.

We provide tools to: compute the domain and range of the data; scale data; generate variation-diminishing and least-squares splines in multiple dimensions; provide ubiquitous function plots; generate orthographic and perspective color-level plots; compute sounds suitable for inclusion in scientific videos to demark time and some other scalar parameters. In later sections of this paper, we describe in more detail some of the other tools that we have found valuable. We have found that building small tools has made it possible to move from one hardware platform to another with minimal effort.

**2. Slice and dice.** One of the more innovative aspects of our function displays is the *dashed surface*. The scattered data smoothing program *loess*[3, 9, 4] produces not only a data model but also an estimate of standard error at each point. The obvious generalization of error bars would be transparent offset surfaces, but this gets rather cluttered. Instead, dice the surface into patches that get trimmed as the error increases. This is analogous to a univariate function plot in which the function curve is dashed in areas of uncertainty. The technique is straightforward to implement, though one needs to be sure to scale the trimming so that patch *area*, not diameter, is proportional to standard error.

So far, the tools described have mostly been applicable to functions of two space variables. Animation is used for a third variable, such as time in a simulation of transient phenomena, smoothing parameter in data analysis, or continuation parameter in a stability study.

* Typeset on June 7, 1991.

† AT&T Bell Laboratories, Murray Hill NJ 07974 USA. wmc or ehg@research.att.com.

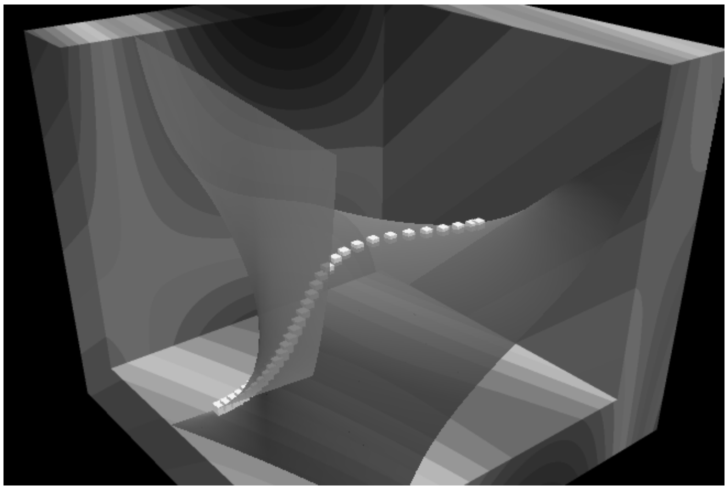

FIG. 1. *Continuation example, illustrating icons on transparent isosurface, with notched cube and color contours as reference frame.*

For functions of three space variables, our goal is to emulate the outstanding pictures one finds in a good anatomy text. First we want to cut away parts of the three-dimensional domain, not just with a single cutting plane but making a rectangular notch, sometimes stopping the cut along a curved internal surface. Sometimes this means a blood vessel or the like is left intact in the excavated region. Second, color and surface texture increase contrast on the internal surfaces thus revealed. Finally, distinctive objects (*icons*) are inserted to point out important features.

So far we have taken only the first steps toward this goal. In one VLSI simulation we were recently involved with, the operating envelope of a device was to be computed by holding one voltage fixed, varying a second by an ordinary continuation procedure, and searching for a turning point. Then we varied the second voltage and traced out turning points in the first voltage using predictor-corrector continuation applied to an augmented nonlinear system. As Rheinboldt has analyzed in detail[15], this amounts to tracing a curve on a certain manifold and we have gotten some insight into the behavior of the continuation procedure by displaying this manifold.

A segment in the videotape shows a continuation example of this kind; a still from this is (inadequately) shown in Figure 1. We start with a scalar function of three variables. Although we plan to borrow a BSP-tree program[13], *sculpt*, to specify arbitrary slicing, for now we make do with a simple notched cube, and display by color bands the values of the function on the visible faces. This gives an overall impression of the function to the trained eye, and provides a frame of reference for the isosurface, which is the equilibrium manifold for this continuation problem, and the key item of interest.

Successive turning points found in the numerical continuation are marked by small cubes. The surface is slightly transparent so that the continuation path can be at least roughly seen from any angle. (This presents a technical difficulty, because the z-buffer algorithms used for hidden surface elimination do not properly deal with transparent surfaces in general. In this particular use, we can arrange to compute the pieces of the isosurface in back to front order and sidestep the difficulty.) Both the isosurface and the cubes are given a non-spectral color to contrast with the background. Even so, a proper vantage point is essential for a clear view of the action. So in the animation we pick a camera position using the surface normal at the continuation point just computed, thus encoding some curvature information in the camera sweep.

At present we compute the isosurface by the customary linear interpolation. By adding surface normals at polygon vertices, obtained from a trivariate quadratic variation diminishing tensor spline approximation, visible artifacts are much reduced. This gives a smooth appearance without the need for a very fine grid. However, it does not get around the fundamental limitations of linear interpolation near singular points. We have successfully used Rockwood's program[16] on an experimental basis, but need to adapt it to generate patches instead of points before we can use it routinely.

**3. Sound and color.** Yet more information can be conveyed in this animation of the continuation process by using the sound track. In our example, we used a snare drum to indicate each Newton step, and a bass drum to indicate a converged turning point. Upon playing the videotape, we noticed for the first time by hearing an abrupt change in the sound track that the convergence got into trouble just as the continuation switched variable 1 to variable 2. Examination of the detailed numerical output confirmed this, giving us a valuable pointer of where to direct further numerical efforts.

In other animations of simulation results, by listening to the sound track we have discovered restarts in the differential equation solver that had earlier gone unnoticed. This was caused by a phenomenon related to stretching the $x$ axis in line drawing graphics. Sometimes small features that can't be seen when the entire dataset is displayed at once become apparent when the scale is stretched and only a small segment of the dataset is displayed at a time.

Even when sound does not lead to a discovery in the data as dramatic as this, it can serve a useful role as a tick mark for the time axis. This could alternatively be achieved by printing the time or, better, drawing a clock face, in the corner of the screen. Sound has the advantage that the viewer's eye is not drawn away from the main action. Animations go by rather quickly until one has viewed them many times, so this advantage of sound (like the head-up display in aircraft) can be valuable.

For further discussion of synchronizing sound and video, see [6]. That paper also describes a numerical procedure for automatically choosing colors for contour bands. Nonlinear least squares applied to psychophysical metrics can eliminate a manual task that is otherwise tedious and arbitrary. For preliminary efforts to correct loudness similarly, see [7].

With all the fascination of elegant renderings and snazzy sound, it is easy to lose sight of the fact that simple line drawings are still the most important graphical tool we have. Probably 80% of our displays come from a tiny library of function and data plotting routines[11]. These generate files in an equally tiny language that can be trivially converted to PostScript, pic, or whatever other graphics is locally available. For dependency graphs and other such figures, some useful tools are MetaPost[12] and dag[8]. PC-based drawing programs are also attractive, but do not lend themselves to data-assisted creation and updating.

One line drawing that deserves wider use is *brushing scatterplots*[2]. This technique for displaying sets of multivariate data points presents a matrix of all pairwise scatterplots and allows the mouse to paint points in one plot while simultaneously highlighting the corresponding points in other views. By this means, a number of patterns and outliers can be quickly identified.

**4. Hardware-related choices.** Not so long ago, graphics applications tended to be tied to specific hardware. The reason was not lack of standards in calling sequences and file formats, but rather in fundamental differences of hardware architecture and the need to squeeze the last ounce of performance out of a system in order to get acceptable speed. On one system, color-band contour plots would be produced by operating on the color lookup tables; on another system, using a texture map proved more effective. Fortunately, the situation has changed and we are now blessed with comparable functionality in all the high-end graphics devices. Even the lower-cost devices have plenty of performance for common visualization needs. However, hardware considerations continue to force important architectural decisions in graphics software.

High resolution color workstations (1280 by 1024 pixels, 8 bits deep) seem to be the current norm, but we prefer to work with smaller image buffers (640 by 480 pixels, 24 bits deep). This is a strategic decision; if you develop applications that depend on high resolution, until HDTV arrives you'll find it very difficult to produce a legible videotape. How often we hear the lament at conferences, "Well, you can't see it very well here, but back in my lab..."! Working at the lower resolution forces one to come to grips early on with proper antialiasing, integrating it well into the environment instead of leaving it as a last-minute hassle when making photos under deadline pressure.

The NTSC discipline also discourages the application writer from filling the screen with menu and dial widgets. We prefer to relegate these to an adjacent monochrome X display or, even better, to prune the user interface ruthlessly so no widgets are necessary. For example, we adopted a helicopter model for all our tools, so that the mouse is used as a virtual control stick for flying around the object on display. Immediate visual feedback works more smoothly than twirling a dials attached to each axis. As another example, Pat Hanrahan's *medit* program lets one move lights around by grabbing a reflection highlight with the mouse and moving it on the surface of a displayed object. To our eyes, a control widget on-screen is like chartjunk[17] in a line drawing; it may serve some valid purpose, but the overall effect is to distract from the data.

For portability, we have restricted ourselves to a three-button mouse as the only input device. There is some concern also that adding degrees of freedom overwhelms the average operator. An interesting lesson in the development of V/STOL aircraft was that pilots learned to take off more stably by being trained to only change a few throttle and attitude variables at a time.

The issue of 8 bits versus 24 bits is less decisive; there are effective dithering and color compression algorithms available to squeeze images into 8 bits. But these are global algorithms and lead to visible artifacts unless properly tuned, so we prefer the simplicity of 24 bits, made affordable by the astonishing drop in memory prices. The image needs to be rendered anyway at 24 bits (or more, since alpha and depth values are also commonly needed). The 640 by 480 resolution, in contrast, does yield a noticeable improvement in rendering times.

Another advantage of standardizing on NTSC is that displays are inexpensive enough to put on everyone's desk, and signals can be readily distributed by existing video switch technology. Graphics tends to be a tool that is just used a few times a day; when one has to run down the hall to a graphics lab, it tends to be used less than it should be. One solution is to put personal graphics workstations in each office, but that is expensive. Making the displays cheap allows one to purchase fewer, more

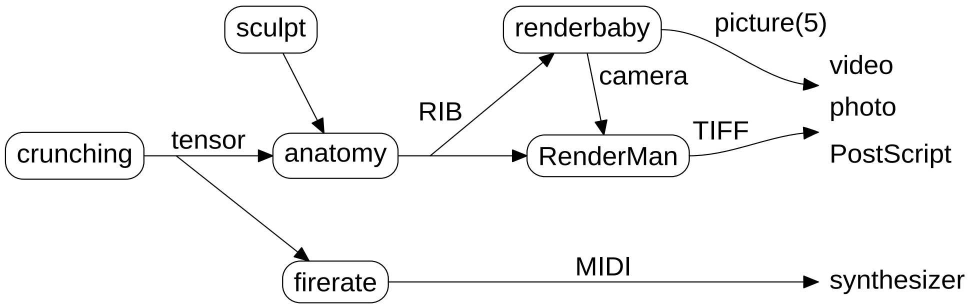


FIG. 2. *Software pipeline, with the "main path" consisting of batch-mode computation culminating in synchronized audio-videotape. The "anatomy" filter is composed of several independent programs reading spline coefficients in "tensor" format and producing geometric objects in "RIB" format.*

powerful graphics engines to be shared among more people, some of them just casual users who can't justify the cost of a dedicated workstation.

Even with an investment in high-end graphics workstations, the rendering time for many of our displays may be several seconds per frame. Rather than compromise on the graphics or spend great effort optimizing the rendering, we content ourselves with making a video in batch mode, recording animation frame-at-a-time. The numerical simulation that feeds the graphics is typically run in batch mode anyway, so this does not feel very restrictive.

Our top level pipeline is shown in Figure 2. RIB (RenderMan Interface Bytestream) [14] is a convenient language for describing geometry. It has enough generality that all visualization techniques we can currently contemplate using are covered; it makes gluing together the output of several independent programs easy; it can be rendered by high-quality, commercial software or (in restricted subsets) by simple home-brew drivers adapted to local hardware. TIFF and picture(5) are two image file formats. MIDI is a standard music file format.

The economics of publishing at the present time preclude the use of color for specialized scholarly journals. Since animation is at least as valuable to us as color, we choose to skip from PostScript™ drawings straight to video, bypassing the difficulties of printing accurate color.

We thank Allan Wilks for contouring code and for comments on a draft of this paper. PostScript is a registered trademark of Adobe Systems Inc. RenderMan is a registered trademark of Pixar. UNIX is a registered trademark of UNIX System Laboratories, Inc.

## REFERENCES

[1] A. V. AHO, B. W. KERNIGHAN, AND P. J. WEINBERGER, *The AWK Programming Language*, Addison-Wesley, New York, 1988.

[2] R. A. BECKER AND W. S. CLEVELAND, *Brushing scatterplots*, Technometrics, 29 (1987), pp. 127–142.

[3] W. S. CLEVELAND, S. J. DEVLIN, AND E. GROSSE, *Regression by local fitting: Methods, properties, and computational algorithms*, J. Econometrics, 37 (1988), pp. 87–114.

[4] W. S. CLEVELAND AND E. GROSSE, *Computational methods for local regression*, Statistics and Computing, 1:1 (1991).

[5] W. M. Coughran, Jr. and E. Grosse, *A philosophy for scientific computing tools*, SIGNUM Newsletter, 24:2/3 (1989), pp. 2–9.

[6] ———, *Techniques for scientific animation*, SPIE Proceedings, 1259 (1990), pp. 72–79.

[7] ———, *Seeing and hearing dynamic loess surfaces*, in Interface'91 Proceedings, Springer-Verlag, 1991.

[8] E. R. Gansner, S. C. North, and K. P. Vo, *DAG — a program for drawing directed graphs*, in Unix Research System Papers, Tenth Edition, Volume II, Saunders College Publishing, 1990, pp. 147–162.

[9] E. Grosse, *LOESS: Multivariate smoothing by moving least squares*, in Approximation Theory VI, C. K. Chui, L. L. Schumaker, and J. D. Ward, eds., Academic Press, New York, 1989, pp. 299–302.

[10] ———, *How shall we connect our software tools*, in Visualization'91 Proceedings, IEEE Computer Society Press, 1991.

[11] E. H. Grosse and W. M. Coughran, Jr., *The pine programming language*, Numerical Analysis Manuscript 83-4, ATT Bell Laboratories, 1991.

[12] J. D. Hobby, *A MetaFont-like system with PostScript output*, Tugboat, the TEX User's Group Newsletter, 10 (1989), pp. 505–512.

[13] B. Naylor, *SCULPT an interative solid modeling tool*, in Graphics Interface '90, Canadian Information Processing Society, 1990, pp. 138–148.

[14] Pixar, *The RenderMan Interface, Version 3.1*, 1989. 3240 Kerner Blvd, San Rafael CA 94901.

[15] W. C. Rheinboldt, *Computation of critical boundaries on equilibrium manifolds*, SIAM J. Numerical Analysis, 19 (1982), pp. 653–669.

[16] A. Rockwood, *Accurate display of tensor product isosurfaces*, in Visualization '90 Proceedings, A. Kaufman, ed., IEEE Computer Society Press, 1990, pp. 353–360.

[17] E. Tufte, *Visual display of quantitative information*, Cheshire, Conn., Graphics Press, 1983.