Computer-Aided Design and Applications
© 2008 CAD Solutions, LLC
http://www.cadanda.com

Multivariate Visualization on Parametric Surfaces

James R. Miller

University of Kansas, jrmiller@ku.edu

ABSTRACT

Attribute Blocks are the basis of a multivariate visualization scheme that has been shown to be effective for two-dimensional data sets such as those common in Geographical Information System (GIS) applications. In this paper, we present an extension of this technique into three dimensions. Specifically, we show how the variation of multiple attributes defined continuously across a parametric surface can be explored using the Attribute Block technique. Any combination of geometric and/or non-geometric attributes can be visualized in this way. We present the method and illustrate it with a few simple examples including the visualization of purely geometric surface differential properties as well as the visualization of non-geometric properties such as the variation of temperature and stress across the surface. The method has educational potential as an aid to help students understand surfaces and surface parameterizations. It also has potential as a surface inspection tool to highlight areas of interest from a differential property point of view that might not be obvious from standard surface displays.

Keywords: multidimensional visualization, programmable shaders, GPU, CAD education.
DOI: 10.3722/cadaps.2008.142-152

1. INTRODUCTION

Multidimensional multivariate visualization refers to the visualization of *n* attributes as a function of their position in *d*-dimensional space [1], [8], [15]. We have previously described a technique we called *Attribute Blocks* and demonstrated its use in the $d=2$ case using examples drawn largely from geographical applications [9]. In this paper, we extend the technique to 3D by mapping through the parameter space of a parametric surface. The variation of an arbitrary number of attributes continuously defined across the surface can then be visualized.

The Attribute Block method employs an array of “lenses”, each capable of presenting a visual representation of a given attribute. We map the attributes into color ramps and display the corresponding color in the lens. The size of the lenses is independently controlled – either in pixel space or in model space – and the color will in general vary across the extent of the lens according to how the underlying attribute varies across the corresponding region of the surface.

The tool has been designed to be highly interactive and exploratory in nature. The size of the Attribute Block lens array can be dynamically altered, as can the assignment of attributes to lens array locations. The lens array is tiled across the surface, yielding a periodic sampling of each attribute across the surface. Altering the size of the lenses modifies the period; it is also possible to interactively adjust the phase shift in each direction by altering the assumed origin of the Attribute Block lens array. These and other tools described below allow the user to interactively explore the surface and its associated properties.

The method relies on the ability to decide on a pixel-by-pixel basis which attribute is to be used to generate the color of the pixel. We employ programmable vertex and fragment shaders on the GPU to accomplish this. All attributes defined on the surface are passed to the vertex shader which, among other things, arranges for them to be linearly interpolated across the generated graphical primitives so that the fragment shader can use an appropriate interpolated attribute when coloring the pixel.

We have used the system to illustrate to students the impact of certain types of control point operations and knot vector settings on surface differential properties and parameterization speed. We have also used it to show how various non-geometric properties whose values are known at discrete locations on the surface can be interpolated and used to visualize how the attributes continuously vary across a surface.

2. PREVIOUS WORK

The work described in this paper depends on our ability to present visualizations of multivariate data in 3D. The multidimensional multivariate research literature is much too large to review in a comprehensive fashion here. Some good entry points into this literature include [1], [5], [13]. We briefly highlight some of the major related techniques here.

Slocum [13] classifies techniques for the display of multivariate attributes based on whether multiple side by side images are to be visually compared or whether all attributes are combined in a single display. In the former case, one typically encodes one attribute per image. This multiple image approach requires the user to visually integrate the distinct images to visualize all the attributes at a given location. By contrast, when a single displayed image is used for the attributes, some scheme must be employed to encode all the attributes at locations throughout the single image. Several such techniques have been explored, most of which rely on color and/or texture mixtures.

One approach for the single image case that goes beyond simple color and texture usage employs special symbols, the characteristics of which (e.g., color, shape, size, topology) encode a collection of attributes at the location of the symbol. A common example of this technique in the context of visualization of surface differential properties is to display vectors at locations across the surface to represent, for example, partial derivatives and/or surface normals. One issue with this approach is that the display may get overly cluttered.

Multivariate dot mapping is an alternative to color and/or texture mixing for single image visualization of multiple attributes. Dot mapping originated as a technique to better visualize discrete phenomena when additional information could be exploited to illustrate how the phenomena varied throughout a region [13]. For example, instead of visualizing the population of a state by drawing the entire state using a color selected according to the state's total population, dots can be displayed throughout the state, the local concentration of which is chosen to reflect the actual non-uniform distribution of the population throughout the state.

Multivariate dot mapping employs differently colored dots for each attribute in an analogous fashion. This approach has been demonstrated to work well for the multiple discrete attribute case when the number of attributes is low, generally no more than three [11]. Our interest is in visualizing an arbitrary number of continuously (not discretely) varying attributes, hence this approach is not directly applicable. Nevertheless, our Attribute Block technique might be viewed as an analog of multivariate dot mapping in the continuous domain, particularly when the Attribute Block lens size is one pixel.

One of our example applications is the visualization of certain surface differential quantities. The specific ones chosen are somewhat arbitrary and are simply meant to demonstrate that any such quantity that can be computed from the underlying surface can be mapped and visualized in this way. We allow Gaussian and Mean curvatures [3] along with the lengths of various parametric differential vector quantities. Using the lengths of the vectors gave us a new way to demonstrate to students the well-known concept that specific parameterizations alter quantities like parametric speed, even though the geometry is not affected. They were also chosen because of their ability to highlight the presence or absence of non-zero derivatives – especially mixed partial derivatives – something that is frequently difficult to detect when looking at a conventional shaded image display.

The importance of calculating and managing differential surface properties is well established. Several researchers have studied the problem of approximating these quantities from triangular meshes (e.g., [4]). These methods are generally based on analyzing vertices in a neighborhood in order to arrive at approximations of relevant differential properties. Since we know the exact underlying surface, we have no need to approximate these quantities; rather we simply evaluate them as needed.

Designers in the automotive industry have long understood the importance of visualizing surface differential properties [2], [14]. For example, Dill's 1981 SIGGRAPH paper [2] explored the use of color images to display average, Gaussian, and principal curvatures of surfaces. As we do here, he used images of a torus with colors generated according to a specific curvature measure as a sort of training or calibration example to help users get a feel for the technique. He then showed the application of the method to more general surfaces. Our technique can be used to produce the same displays (see below, for example). Moreover, one of the advantages of the Attribute Block method is its ability to display several such quantities at once.

Somewhat more recently, Singh has reminded us of the importance that automotive designers place on the need for smooth surfaces so that, for example, the size and shape of highlights on a car are smooth and unbroken [12]. Achieving this requires that the surfaces be at least C^2 everywhere. The examples presented below do not do a good job of conveying C^k parametric continuity for any $k>0$, in part because the current technique is insensitive to span boundaries. We are currently studying ways to address this shortcoming. Finding a solution would be helpful so that such issues could be monitored and visualized interactively.

Finally, we rely heavily on the ability to program specialized functionality into vertex and fragment shaders inside a GPU. Others have done this in the context of geometric modeling as well. Kurfess, et al. [7] describe general characteristics of GPUs and GPU programs that have been found to be beneficial in geometric modeling and CAD applications. Their survey explains the software and hardware architecture of GPU-based systems, and describes how to take advantage of these capabilities.

Kanai [6] described a rendering algorithm for NURBS surfaces based on evaluating the B-spline basis functions on a GPU and computing positions and normal vectors on the surface on a per-pixel basis. The surface is initially tessellated into polygons, then as the polygons are scan converted, true surface positions and normal vectors are computed in the fragment shader and used to drive a lighting model.

3. ATTRIBUTE BLOCK BASICS

The Attribute Block technique is used to visualize *n* attributes defined continuously across a region presented in a single image. Coloring each pixel using some multivariate function of the attribute values at that location is ineffective for more than two or possibly three variables. Our approach is instead based on arranging all or an interactively selected subset of the *n* attributes into a dynamically configurable k_r x k_c array of visual representations called an *Attribute Block*. Each element of the Attribute Block array encodes a single attribute value. All attributes are defined everywhere; the Attribute Block array simply acts like a "screen door" of lenses, each allowing a single attribute value to be seen at the specific location. The dimensions of each "lens" – denoted b_r x b_c – are also dynamically adjustable. The current k_r x k_c array of b_r x b_c lenses is then tiled across the entire object on which the attributes are defined. Figure 1 illustrates a basic diagram of the Attribute Blocks used to create Figure 2. While the four cells of the Attribute Block in Figure 1 exhibit constant shading, the shading will in general vary across a cell to reflect the variation of the corresponding attribute. Figure 3 shows the legend used to generate Figure 2.

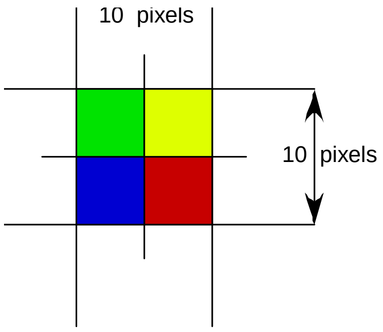

Fig. 1: Logical layout of Attribute Block.

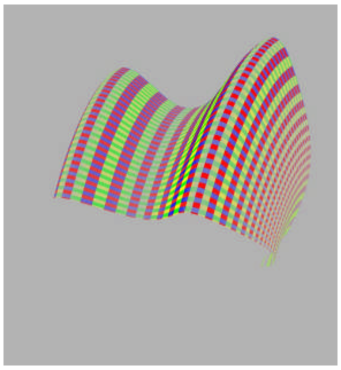
Fig. 2: A surface with tiled Attribute Blocks.

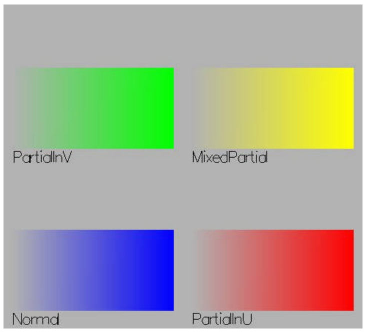

Fig. 3: Labeled legend for Attribute Blocks used in Fig. 2.

As first introduced in [9], the Attribute Blocks method was a tool locked to 2D geospatial coordinates, and the generation of the texture used to visualize attributes was constrained to the resolution of the display device. The major extensions presented here allow us to apply the method in 3D space and completely decouple the resolution of the generated texture from that of the display device. A multi-step display callback is now used along with OpenGL pbuffers to generate an Attribute Block texture at an arbitrary resolution that is then mapped onto a parametric surface. Other extensions to the basic technique as outlined in [9] include improved exploration tools such as an animation technique that allows the attribute block array to crawl across the surface, revealing features not apparent with static displays. There are also a large number of display options including allowing various combinations of lighting models; polygon display with and/or without Attribute Block textures; and edge displays to show the resolution at which the texture was generated.

The display of Figure 2 is generated by using the computed lengths of the cited vectors. The color legend varies from background color for the minimum length to the full indicated color for the maximum length. By default, the program determines the overall min and max for each attribute and then scales that range into the indicated minimum to maximum color range. This choice makes large attribute values stand out while small ones fade into the background. This behavior can be modified interactively. In addition, the user can redefine the default color assignment for each attribute on a per-surface basis.

The majority of the surface images in this paper are rendered without a lighting model. The purpose of the technique we describe here is to use color to represent values of attributes at various positions on a surface. Use of a lighting model often compromises our ability to visualize the attributes because it involves modifying pixel colors based not on attribute values, but rather on the position and strength of light sources. Fortunately, the structure of the Attribute Block displays tends to make the three dimensionality of the surfaces obvious, even when only a single attribute is being displayed (cf., Figures 4-6 below). The tool allows the lighting model to be turned on and off interactively.

As mentioned above, we can dynamically alter the assignment of attributes to locations in the $k_r \times k_c$ array; we can also interactively change the dimensions (i.e., the k_r and k_c). As a trivial special case, we can set $k_r=k_c=1$ and produce a display of a single attribute on the surface. We shall see other special cases below, most notably, setting $k_r=1$ and $k_c>1$ (alternatively $k_r>1$ and $k_c=1$) produces "Attribute Columns" (or "Attribute Rows"). We begin the next section with an example of $k_r=k_c=1$ and show how Gaussian and mean curvatures can be displayed on a surface. We then move on to show other Attribute Block arrangements and discuss how they can be used.

4. USING ATTRIBUTE BLOCKS WITH PARAMETRIC SURFACES

The example of Figure 4 shows a torus represented as a NURBS surface. Using an Attribute Block array with $k_r=k_c=1$, we compute and map Gaussian curvature across the surface, yielding a result like that in Dill's paper [2]. (Our color ramp is not exactly the same, but the displays are otherwise similar.) Figure 5 shows Gaussian curvature on another surface that will be used in some examples below. Figure 6 shows mean curvature on that same surface. We compute the Gaussian and mean curvatures using the methods outlined in [3].

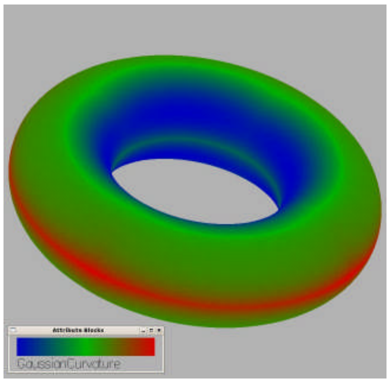

Fig. 4: Gaussian curvature on the NURBS representation of a torus.

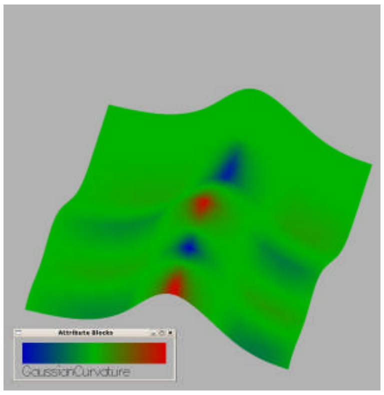

Fig. 5: Gaussian curvature of a NURBS surface.

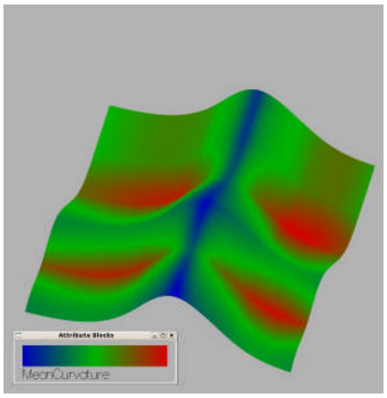

Fig. 6: Mean curvature of the surface of Figure 5.

One of the major advantages of the technique we are presenting here is that one need not be limited to viewing a single attribute at a time. An Attribute Block array can be constructed which allows an arbitrary number of attributes to be visualized simultaneously. Not surprisingly, the effectiveness of the technique is heavily influenced by color ramp selection. It also seems to be affected by how related the attributes are; however this issue needs more study. In the first few examples below, we will use a 2x2 Attribute Block array in which we present at each point on the surface the lengths of (i) the first partial derivative in *u*, (ii) the first partial derivative in *v*, (iii) the surface normal vector (computed as the cross product of the previous two), and (iv) the mixed partial derivative ($\partial^2 / \partial u \partial v$).

4.1 Visualizing Parametric Derivatives on a Polygon Represented as a NURBS Surface

It takes some training to get proficient with understanding Attribute Block displays. We will therefore begin by looking at an example of a polygon represented as a bicubic NURBS surface in order to begin to get a feel for what the displays can tell us. The knot vectors in *u* and *v* force interpolation of the first and last row and column of control points, but they are otherwise uniform. Figure 7 shows the Attribute Block array of Figure 3 tiled across the flat surface. (The polygon does not cover the display, hence the background appears as an outer gray frame.)

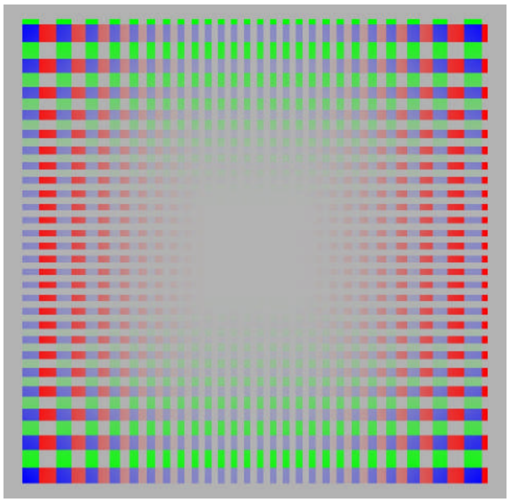

Fig. 7: A polygon represented as a bicubic NURBS surface with partial and normal vector lengths.

Several important impacts of the nonuniform knot spacing used to achieve end point interpolation are clearly displayed. Each Attribute Block cell is a fixed (square, in this case) size in the parameter space, but they appear larger towards the boundary due to the faster parameterization. This effect is actually encoded in two ways in this image: the blocks themselves are larger and not square, even though they correspond to fixed parameter space steps, and the coloring indicates that the lengths of the partial derivative vectors (depicted by the red and green cells) are greater near the ends. The mixed partial (depicted by yellow) is zero everywhere as it must be, and the normal vector length (blue Attribute Block cells) is largest where the partial derivative vectors used to compute it are largest.

The example of Figure 8 uses a larger Attribute Block array to explore second derivative effects. The mixed partial is still zero everywhere, of course. The array is 3x3, but we really are only seeing six non-blank cells. Notice that this makes it a little easier to track attributes across the surface. By visually scanning along a row (column), it is possible to observe that large second partial derivatives predict corresponding changes in the first partial in *u* (*v*). Given only the length, it is not possible to decide strictly from the display whether the first derivative should grow or shrink, but it does clearly predict change.

4.2 Visualizing Parametric Derivatives on a Torus Represented as a NURBS Surface

Figure 9 shows a torus represented as a NURBS surface along with an Attribute Block display in which the two first partial derivatives, the mixed partial, and the normal are presented. While somewhat easier to see in a larger image, it is fairly clear that the magnitude of the first derivative in *u* is largest along the outside and smallest along the inside diameter. The same holds true for the normal vector.

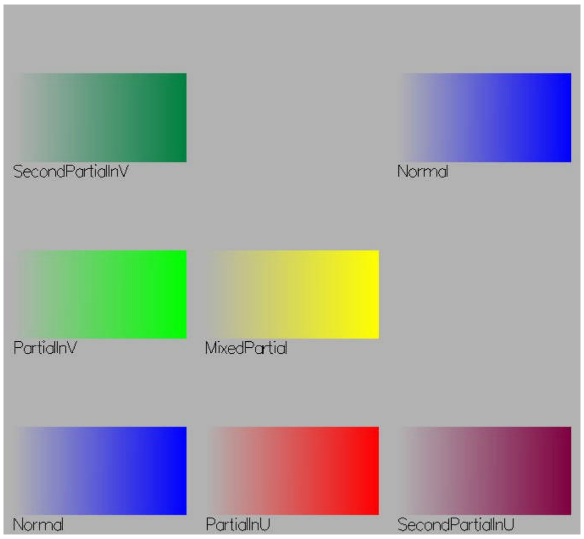

Fig. 8(a): The Attribute Block layout used in Figure 8(b).

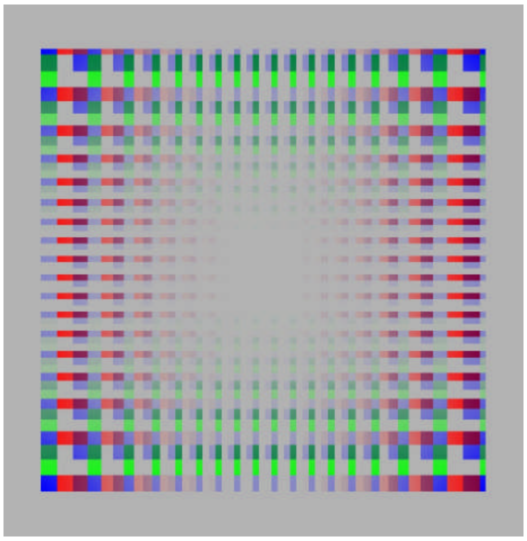
Fig. 8(b): First and second partials on the polygon.

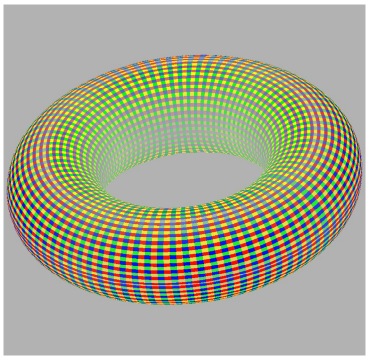
Fig. 9: Visualizing 4 attributes on a torus. (Uses layout of Figure 3).

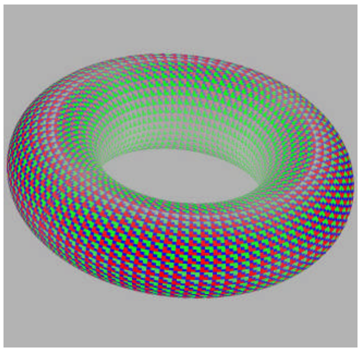
Fig. 10(a): Visualizing three attributes on the torus.

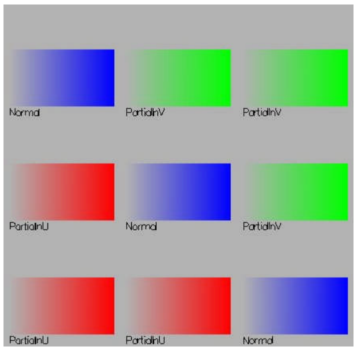

Fig. 10(b): The Attribute Block layout used in Figure 10 (a).

Sometimes it is easier to track attributes across the surface when they are arranged in patterns. The torus display of Figure 10 (a) uses the Attribute Block layout of Figure 10 (b) in which the two first partials are in the shape of orientable corners. The normal is arranged down the diagonal. The presence (and probably more significantly, the absence) of these parameters is very obvious in the display.

4.3 An Example with Non-Zero Mixed Partial Derivatives

The surface shown in Figure 11 was generated by perturbing the bicubic NURBS representation of the polygon shown in Figure 7. Specifically, one interior column of control points was displaced directly up along the polygon normal, then an interior row of control points was displaced up along that same vector. (The common control point was therefore displaced twice as far as all the others.) The surface of Figure 11(b) was created from that of Figure 11(a) by slightly moving the lower left and upper right control points away from the surface while keeping them in the *xy*-plane. The difference between the two surfaces is fairly obvious here, however similar and/or smaller such modifications are frequently difficult to detect visually.

Among other things, the sequence of shape modifications described above lead to some different differential surface properties. Specifically, the surface of Figure 11(a) has a zero mixed partial derivative everywhere. By contrast, the surface of Figure 11(b) has significant non-zero mixed partial derivatives in the vicinity of the two corners mentioned. See Figures 12(a) and 12(b). Both use the same Attribute Block layout of Figure 3.

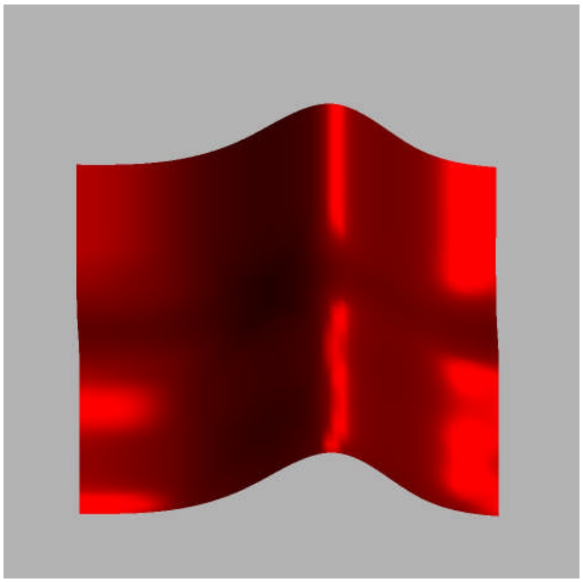
Fig. 11(a): A NURBS surface created from the surface of Figure 7 as described in the text.

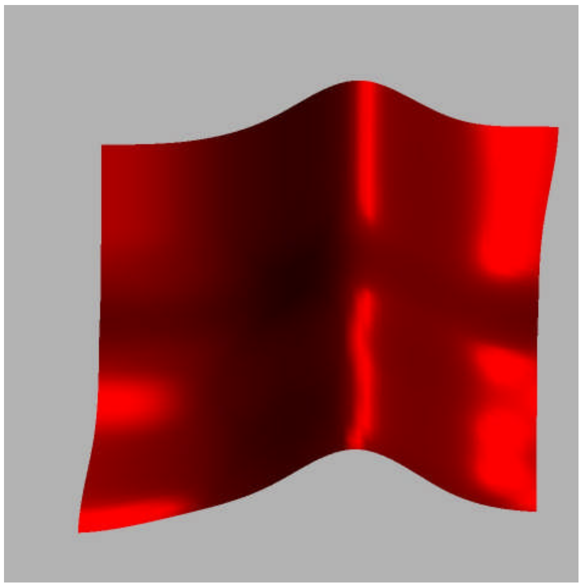
Fig 11(b): A slight perturbation of the surface of Figure 11(a) as described in the text.

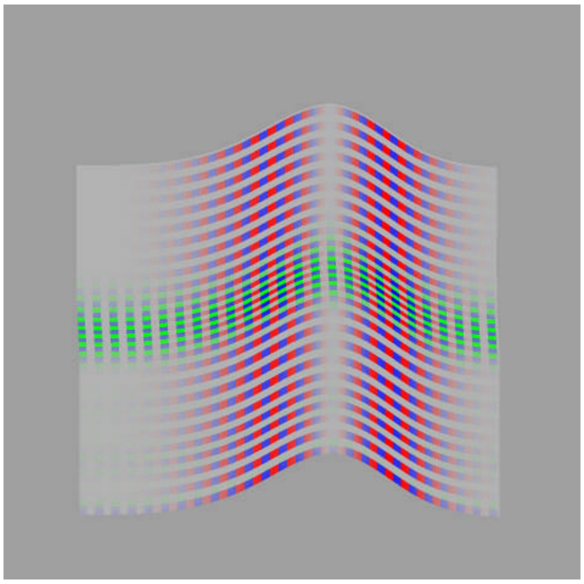
Fig. 12(a): The Attribute Block display of the surface of Figure 11(a).

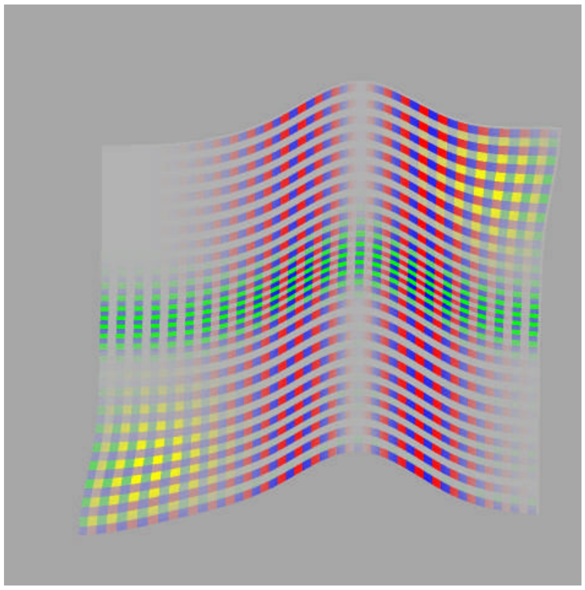
Fig. 12(b): The Attribute Block display of the surface of Figure 11(b).

4.4 Aircraft Surfaces

The example of this section illustrates the use of these methods on surfaces used as part of an aircraft fuselage. Figure 13 illustrates three surfaces (two large and one light-colored fillet between them) which will be trimmed to one another during the construction of a portion of a model of a fuselage. The dashed lines in Figure 13(b) show the NURBS span boundaries, not the polygons used for display.

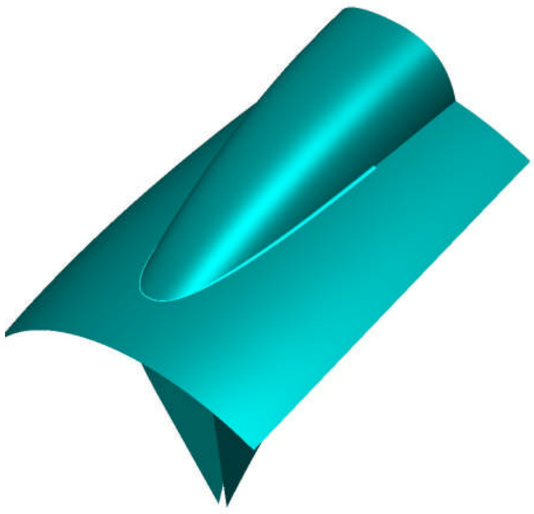
Fig. 13(a): Three surfaces from a fuselage.

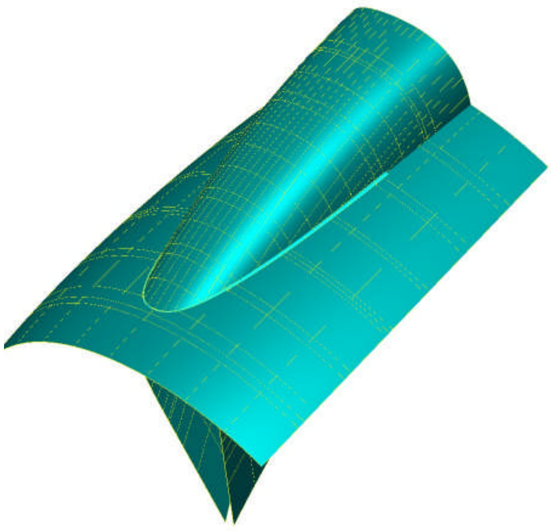
Fig. 13(b): Dashed yellow lines outline NURBS spans.

Figure 14 shows how we can simultaneously visualize curvature and mixed partial derivatives. Figure 14(a) shows Gaussian curvature along with the magnitude of the mixed partial derivative vector; Figure 14(b) is the same, except with mean curvature.

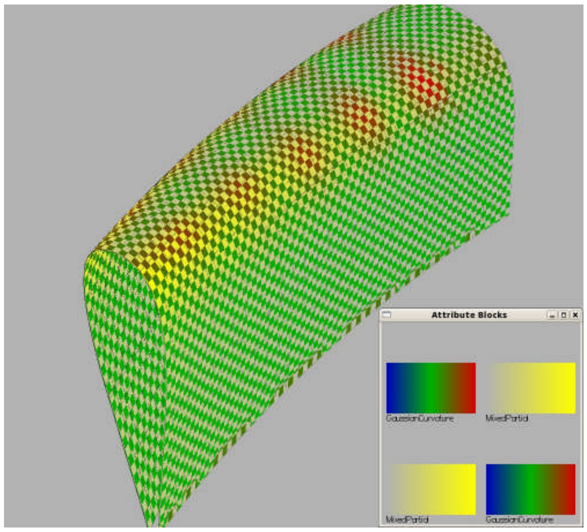

Fig. 14(a): Combined Gaussian curvature and mixed partial derivative vector length.

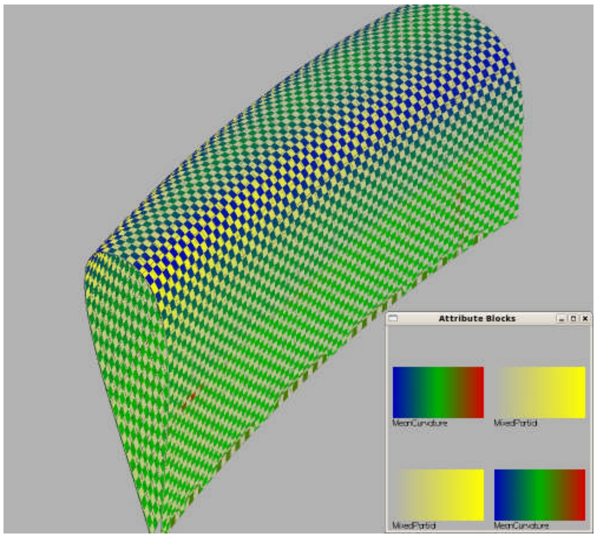

Fig. 14(b): Combined mean curvature and mixed partial derivative vector length.

It is clear from the figures that the mixed partial is (at least nearly) zero everywhere except along the edges of the ridge on top where it has maximum positive values. Figure 14(a) shows that the Gaussian curvature is (nearly) zero everywhere except in that same area where it also exhibits maximum positive values. Figure 14(b) indicates that mean curvature reaches its maximum (but negative) values there, and it is (near) zero elsewhere.

4.5 Visualizing Non-Geometric Attributes on a Parametric Surface

All the examples to this point involved displaying geometric attributes directly computable from the surface geometric description. The Attribute Block scheme is not restricted to quantities that are computable in this fashion, however. Any attribute for which a value is explicitly known throughout the surface or for which a value can be interpolated across it from various known positions can be incorporated.

As an example, consider the surface whose Attribute Block display is shown in Figure 15. Three non-geometric attributes have been defined and mapped to the surface. All were artificially generated, but are representative of the types of data one might want to visualize on a surface. For this example, we created "Spring Force" as a function of the signed distance this surface was from a reference surface. The simulated spring force presumably is a measure of the force pulling the surface back towards the reference surface. Blue colors in that scale indicate force along the normal vector; yellow colors indicate forces in the direction opposite to the normal vector. Additionally, we can define an arbitrary number of attributes whose values are known at fixed locations in the parametric surface domain and which are interpolated (using a simple inverse distance weighting algorithm here) to all points on the surface. Those two simulated attributes are "temperature" and "strain".

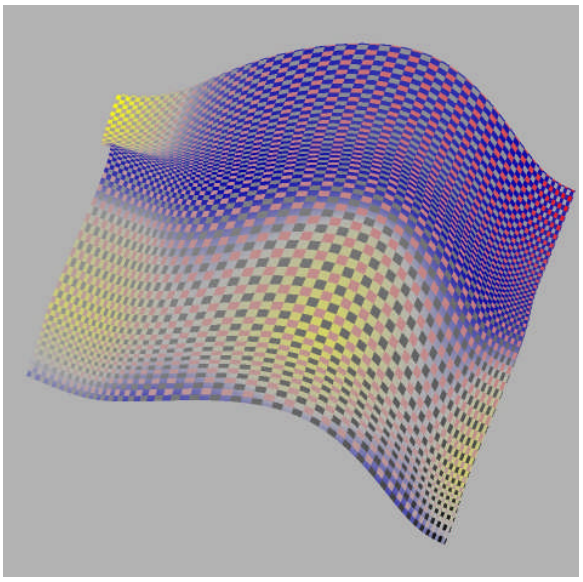
Fig. 15(a): An Attribute Block display with purely non-geometric attributes.

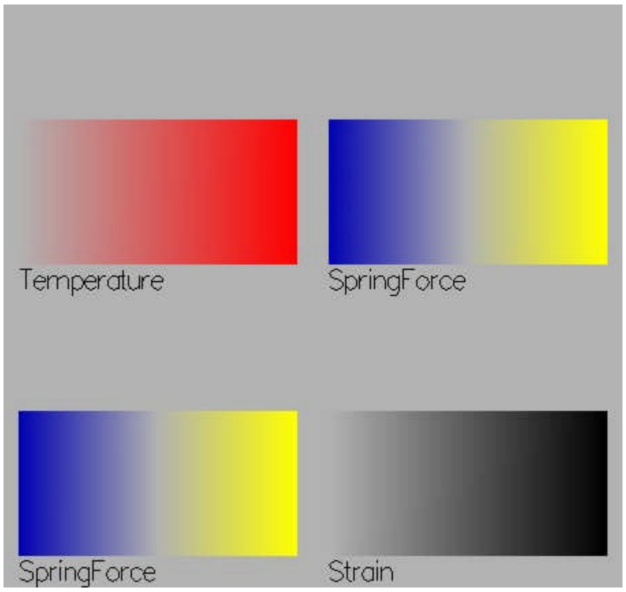

Fig. 15(b): The Attribute Block layout for Figure 15(a).

By examining the figure, it is immediately obvious what areas are being pushed up or down by the "spring force". In addition, the "strain" is clearly concentrated on the front lower right corner and falls off fairly rapidly. The "temperature" appears to achieve its maximum at the rear right corner, and seems to be a minimum at the other three corners.

It is sometimes useful to be able to track attribute variations across the surface without having to skip across Attribute Block cell boundaries. One could of course use a kr=kc=1 arrangement as we saw earlier in Figures 4-6. An alternative is the set just one of kr or kc to 1 and use an "Attribute Column" or "Attribute Row" display involving *n* attributes. Figure 16 illustrates an Attribute Column display with the temperature and spring force attributes from the example of Figure 14.

5. SOME IMPLEMENTATION NOTES

The system has been developed using OpenGL and its GLSL programmable shader language. We use the OpenGL NURBS tessellation facility, forcing it to sample the surface at fixed parameter space locations. We intercept the vertex calls and implement a multiple-pass algorithm. The first two passes are done only when a surface is first created or subsequently modified (e.g., control point or knot vector adjustment). They intercept each vertex callback and, rather than actually sending geometry to the graphics engine, they invert the *(x,y,z)* coordinates to *(u,v)* parameter values and then compute all the relevant differential properties (partial derivatives, normals, curvatures, etc.). Algorithms for inversion and differentiation operations were derived from those described in [10]. Any non-geometric attributes are then computed by interpolation from the known positions to the current *(u,v)* location and associated with the vertices as well. Passes 3 and 4 are executed during each display callback. The Attribute Block texture is created in pass 3 by having the fragment shader write its texture to an OpenGL PBuffer. The PBuffer contents are retrieved at the end of pass 3 and are applied as a texture map to the NURBS surface as it is actually being rendered during pass 4. This approach allows changes to the Attribute Block configuration, Attribute Block cell size, and all the other relevant display parameters to be performed at normal interactive rates.

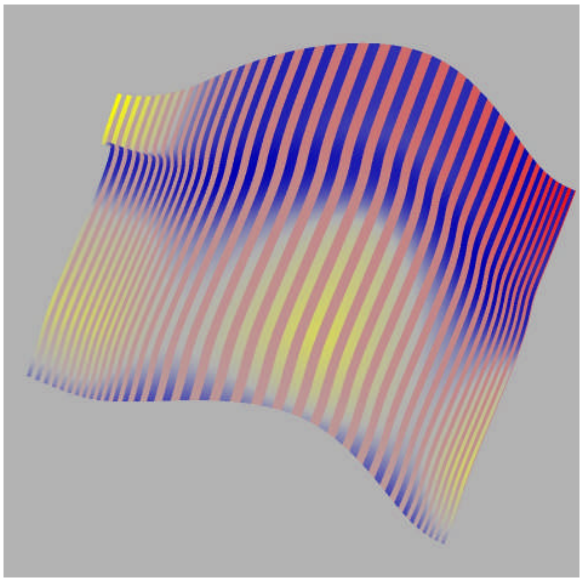
Fig. 16(a): An Attribute Column display of Spring Force and Temperature.

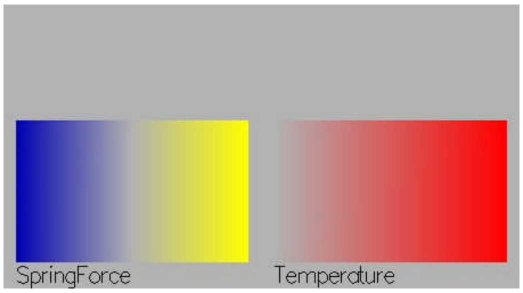

Fig. 16(b): The Attribute Block Layout used for Figure 16(a).

6. SUMMARY AND DISCUSSION

We have presented an extension to our earlier 2D Attribute Block method into the 3D domain. We rely on programmable vertex and (especially) fragment processors to determine on a pixel by pixel basis what attribute is to be used for a given pixel, and then to render the pixel according to the interpolated attribute value. We have shown several examples involving geometric and non-geometric attribute data mapped onto the surface.

There are a number of obvious areas for improvement and advancement. All the scales and images provide only relative indications of attribute values. One very straightforward improvement would be to add a numeric scale to the layout legends. This is a simple extension that is scheduled. A more significant related improvement would be to define more general color mappings. As of now, all color ramps are either (i) single ranges, mapping the attribute (minimum, maximum) range to a color in a single range defined by (RGB_1, RGB_2); or (ii) a double (diverging) map which maps the first half of a range (typically negative attribute values) into a color range (RGB_1, RGB_2) and the second half of the range (typically positive attribute values) into a color range (RGB_2, RGB_3). All mappings use simple linear interpolation. Defining multiple color ramp ranges and adding support for step function and/or nonlinear mappings would represent a significant improvement.

The importance of good color selection for attribute color mappings cannot possibly be overemphasized. Oftentimes a completely chaotic unintelligible display is made coherent by a simple change in one or more color ramps. Tuning colors for these ramps is an important part of any actual use of this technique in a given application. Interactive controls are available to alter the specification of each color ramp. This is not considered to be a routine interactive process, rather the controls are there to help the user when they start work on a new data set to interactively arrive at a reasonable set of colors. Having done so, the normal mode of operation is to transfer the colors to the configuration file for that data set so that it becomes the default on subsequent runs.

While we have done some informal user testing and evaluation, we clearly need a comprehensive user evaluation to be performed to assess the effectiveness of this system as compared to alternative approaches. We are currently in the early planning stages for such an effort.

7. REFERENCES

[1] de Oliveira, M. C. F.; Levkowitz, H.: From Visual Data Exploration to Visual Data Mining: A Survey, IEEE Transactions on Visualization and Computer Graphics, 9(3), 2003, 378-394.

[2] Dill, J.: An Application of Color Graphics to the Display of Surface Curvature, Proceedings SIGGRAPH '81, Computer Graphics, 15(3), 1981, 153-161.

[3] Farin, G., Curves and Surfaces for CAGD: A Practical Guide, Fifth edition, Morgan Kaufmann, 2002.

[4] Goldfeather, J.; Interrante, V.: A Novel Cubic-Order Algorithm for approximating Principal Direction Vectors, ACM Transactions on Graphics, 23(1), 2004, 45-63.

[5] Healey, C. G.; Enns, J. T.: Large Datasets at a Glance: Combining Textures and Colors in Scientific Visualization, IEEE Transactions on Visualization and Computer Graphics, 5(2), 1999, 145-167.

[6] Kanai, T.: Fragment-based Evaluation of Non-Uniform B-spline Surfaces on GPUs, Computer-Aided Design & Applications, 4(1-4), 2007, 287-294.

[7] Kurfess, T. R.; Tucker, T. M.; Aravalli, K.; Meghashyam, P. M.: GPU for CAD, Computer-Aided Design & Applications, 4(6), 2007, 853-862.

[8] Love, A. L.; Pang, A.; Kao, D. L.: Visualizing Spatial Multivalue Data, IEEE Computer Graphics and Applications, 25(3), 2005, 69-79.

[9] Miller, J. R.: Attribute Blocks: Visualizing Multiple Continuously Defined Attributes, IEEE Computer Graphics & Applications, 27(3), 2007, 57-69.

[10] Piegl, L.; Tiller, W.: The NURBS Book, Springer-Verlag, (second edition), Berlin, Heidelberg, 1997.

[11] Rogers, J. E.; Groop, R. E.: Regional Portrayal with Multi-Pattern Color Dot Maps, Cartographica, 18(4), 1981, 51-64.

[12] Singh, K.: Industrial Motivation for interactive shape modeling: A case study in conceptual automotive design, in ACM SIGGRAPH 2006 Short Course: Interactive Shape Editing, 2006, 3-9.

[13] Slocum, T. A.; McMaster, R. B.; Kessler, F. C.; Howard, H. H.: Thematic Cartography and Geographic Visualization, second edition, Pearson Prentice Hall, Upper Saddle River, New Jersey, 2005.

[14] Warn, D. R.: Lighting Controls for Synthetic Images, Proceedings SIGGRAPH '83, Computer Graphics, 17(3), 1983, 13-21.

[15] Wong, P. C.; Bergeron, R. D.: 30 Years of Multidimensional Multivariate Visualization, In Scientific Visualization: Overviews, Methodologies, & Techniques, G. M. Nielson, H. Mueller, and H. Hagen (editors), IEEE Computer Society Press, Los Alamitos, CA, 1997.