# Gridigrator:
# A Very Fast Volume Renderer
# for 3D Scalar Fields Defined on Curvilinear Grids

Al Globus
Computer Sciences Corporation
NASA Ames Research Center

Gridigrator is a software module which renders scalar volumes on curvilinear grids by casting rays along grid lines. Each ray deposits a scalar result on a vertex of the grid's face. This scalar is the result of a 'deposition' function applied to the data points the ray intersects. The resulting scalar field on a surface embedded in three dimensions may be visualized using standard techniques. This variation on ray casting scalar field volume rendering is very fast and can be trivially combined with polygon, line and point based visualizations to create images. Gridigration is a O(n) technique where n is the number of grid nodes. Gridigrator is much faster than traditional ray casting volume rendering on identical workstation hardware. Gridigrator on a workstation is somewhat faster than traditional ray casting volume rendering on a massively parallel supercomputer. Although interpretation of results can be difficult, we expect gridigration to be useful for quick look surveys of scalar fields to find features of interest. A generalization of gridigration based on a medical x-ray imaging analogy is briefly discussed as a promising avenue for further research.

## Introduction

Gridigration is best understood by comparison with traditional ray casting volume rendering (RCVR). A volumetric object may be rendered by casting rays from each screen pixel through the object. The volume is sampled along each ray. A transfer function is applied at each sample point. The results of the transfer function along each ray are composited to determine pixel properties. The resulting image is view-dependent and takes a long time to calculate because the data must be interpolated many times and, particularly for curvilinear grids, the intersections of the rays with the data set elements are difficult to calculate quickly. A more thorough description of RCVR through curvilinear volumes, including extensive references, may be found in [Usel91].

If one wishes to create a single image containing traditional graphical elements (polygons, line, points) and RCVR, it is necessary to know the graphic element location before RCVR begins. Each ray must check for polygon intersections as it is cast and compositing of lines and points must be done in proper depth order [Levo88]. Thus, embedding geometric visualizations known a priori in the image is possible, but one loses the ability to quickly rotate and zoom since the view-dependent volume visualization is too slow. View independence may be achieved by texture mapping the RCVR image onto a polygon located appropriately in space, although the author is not aware of anyone using this approach for scientific visualization.

W. Krueger describes a generalization of RCVR using the formalisms of transport theory from physics [Krue90]. Transport theory allows rays to travel along paths that are not straight. Similarly, the gridigration technique casts rays along grid lines which are, in general, curved.

**An X-Ray Model of Volume Rendering**

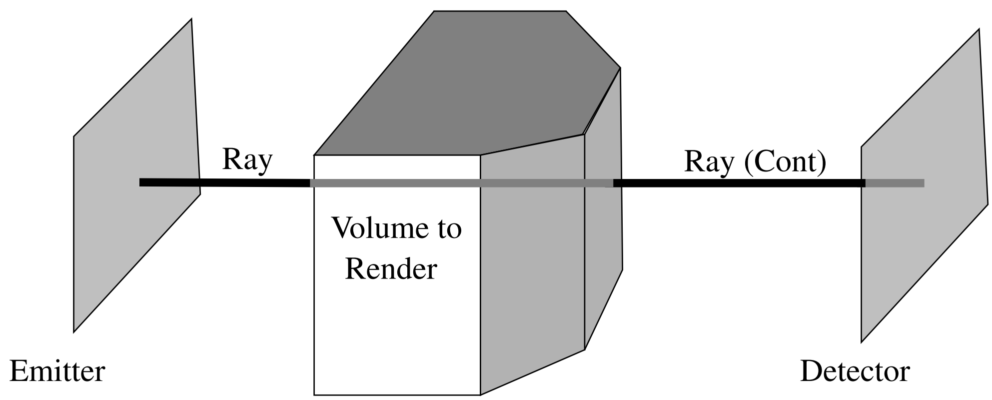


Figure 1: X-Ray Model of Volume Rendering

The next step to understanding gridigration is the medical x-ray imaging analogy. Consider an x-ray lab. Rays are emitted from a machine, pass through the patient's body, are modified by the bones and flesh inside the body, and are detected by a plate placed behind the patient. In this model there is a ray emitter (the x-ray machine), a ray detector (the plate), and rules to determine the path rays take (x-rays are always straight). In software, although not in Gridigrator, both the emitter and the detector may be placed anywhere in physical space. Thus, the resulting visualization is view-independent. Since the detector may be composed of a set of polygons in physical space, the resulting images can be easily combined with other visualizations in visualization systems such as AVS [Upso89], FAST [Banc90] and others. Assuming the rays deposit a scalar field on the detector, the polygons may be color mapped and shaded using the common scalar field visualization techniques. This deposition function combined with color mapping is equivalent to a combination of RCVR transfer and compositing functions. The x-ray model is also discussed in [Darm91].

**Gridigration**

Consider a scalar field defined on a single grid zone in generalized curvilinear co-ordinates, i.e., a 3d scalar array with a corresponding 3d array of x,y,z locations where both arrays are indexed by integers i,j,k. A grid face is the set of vertices where one index is held constant at an extreme value and the other indices vary over their entire range. A grid line is the set of vertices where two indices are held constant and the other varies over its entire range.

To perform gridigration, the ray emitter is co-incident with one grid face and the detector the opposite face. Rays follow the grid lines connecting vertices in the emitter and detector. This operation is performed for each set of opposing grid faces. The rendering calculations are fast since grid lines are very easy to traverse and no interpolation of the data is necessary. A deposition function maps the scalar data values on the grid line to a scalar on the face vertex. Thus, the output is a scalar field on a two dimensional surface embedded in 3-space. Results may be rendered as color mapped, shaded polygons using fast workstation graphics hardware. The resulting image incorporates information from all data points in the scalar field. Since the volume renderer's output is a set of view-independent color-mapped polygons, it may be trivially combined with the results of other visualization techniques.

Interpreting the resulting images requires an intimate knowledge of grid topology, but most scientists and engineers know their grids very well. Not all grid topologies lend themselves to straightforward interpretation using this technique, but many standard grid types have straight lines along one index. In other common grids, grid lines extend radially outward from a simple shape. In these cases, interpretation of gridigrator images is fairly straightforward, although more difficult than with RCVR.

**Method**

Gridigrator is implemented as a module in Silicon Graphics Inc.'s Explorer, a scientific visualization environment. The module was written in C++ [Stro91]. Existing Explorer modules are used to render Gridigrator's output, to input data, to select subsets of the data to gridigrate, to generate and render other volume visualizations (e.g., isosurfaces), etc.

The gridigrator module takes a 3D scalar curvilinear lattice and a deposition function specification as input and produces six 2D scalar curvilinear lattices embedded in 3-space and their extreme values as output. A lattice is the generalized array data structure in Explorer. Each of the 2D lattices is co-located with one face of the 3D lattice. Each pair of 2D lattices corresponding to opposite faces have the same scalar field but different grids. The scalar fields are generated by the deposition function applied to grid lines. To perform gridigration, each data node must be visited three times - once for each grid direction. The module we implemented also visits each node in the six output lattices to determine output extrema to aid color-map generation.

The deposition function is presently limited to integration, minimum value, or maximum value along a grid line. Minimum and maximum deposition functions simply find the extreme value along each grid line. Integration is the line integral along a grid line. This may be calculated by summing the line integral between each connected pair of nodes along the grid line. Integration between two nodes is performed using the trapezoid rule, i.e.,

$$I = \frac{(sn1 + sn2)}{2}(LineLength) \qquad (1)$$

where $I$ is the trapezoidal approximation to the line integral, and *sn1* and *sn2* are the values of the scalar field at connected nodes. *LineLength* may be set to two, effectively ignoring cell size, to bring out features in small grid cells.

The Explorer visualization environment is a data flow system with an interpreted, visual programming user interface. Modules are C, C++ or Fortran functions which interface to Explorer through a set of function calls that make up an application programmer's interface (API) and the Module Builder, an X-Windows System widget based application that generates C++ source code. The generated code is compiled and linked with the API libraries and a C++ function implementing gridigration.

The Gridigrator function call uses the following C++ classes (indentation indicates derived classes):

**MinMax** -- manages extrema values
- **RegularField** -- an n dimensional field with no explicit grid
  - **Field** -- a field with a curvilinear grid
  - **Grid** -- a RegularField interpreted as a grid
- **MultiField** -- a field consisting of multiple fields. Used for the six exterior faces.

**FaceSpecIterator** -- iterate over six specifications of 3D field faces

**FieldIterator** -- iterate over all the nodes in a field
- **FieldFaceIterator** -- iterate over all the nodes on one face of a 3D field
- **FieldLineIterator** -- iterate over all the nodes in one grid line

**IntVector** -- an array of integers

Gridigrator timing data was collected using the UNIX(r) time() utility. Three data sets were examined: a blunt-fin [Hung85], a shuttle main engine LOX post [Roge86], and a shuttle orbiter in flight [Rizk85]. The whole data set was gridigrated in each case, although not all of the output is displayed. For each of the four possible deposition functions (min, max, integration, integration with intra-node length = 2) there were three time trials. The results below are averages.

RCVR timing data was collected using S. Uselton's software (Usel91) on the same hardware configuration used for gridigration. The same blunt-fin simulation data were used, but a slightly different scalar field. 511 x 511 rays were cast. 70,981 rays intersected the data set. The view was in the positive y direction looking towards the center of the bounding box. The deposition function mapped low density to blue/transparent and high density to red/opaque with a linear function between extreme values.

**Results**

Figure 4 is four gridigrations of the blunt-fin. Only three of the six generated surfaces are shown: the plate, the fin, and the grid face where the flow exits the simulation. The four images illustrate the four deposition functions. In each case the extreme values of gridigrator output are shown so that the colors may be properly interpreted. Note the two shocks visible on the flat plate in front of the fin in the bottom right image (linelength = 2). Only one is detectable in the bottom left image. The second, weaker shock only occurs near the plate -- where the grid is severely compressed. Thus, the shock becomes visible since the numerous small cells near the plate contain the feature of interest which is not washed out by the fewer, larger cells farther from the plate.

Figure 5 is a gridigration of a simulation of flow around the NASA shuttle orbiter. The images are very similar to surface pressure plots, but reflect the pressure not only at the orbiter's surface but also along the grid lines leaving the surface. In this visualization, the length of grid lines in physical space is ignored (i.e., *LineLength* = 2 in equation 1). This gives greater weight to the physics near the surface where grid spacing is very close.

Figure 6 shows gridigration combined with a polygonal isosurface to demonstrate the ease with which gridigrator volume rendering is combined with traditional graphic objects. No special software was written to achieve this combination. In fact, the gridigrator module does not draw pictures, it simply creates six scalar lattices that are rendered by other Explorer modules. Figure 7 shows the Explorer 'map' used to generate the image in figure 6.

Gridigration of the data sets examined took from one to six seconds on a Silicon Graphics Inc.

320 VGX workstation using 33 Mhz MIPS processors. Dr. Uselton's RCVR for curvilinear data sets took 280 seconds on the blunt-fin using the same hardware. Comparable time for gridigration was about four seconds. In this case, gridigration appears to be 70 times faster than RCVR on the same hardware. J. Challinger reports 28 seconds to RCVR the same blunt-fin data set using 100 processors of the BBN TC200 massively parallel computer at Lawrence Livermore National Laboratories [Chal92]. Thus, gridigration is more than six times faster than RCVR on much faster hardware.

Figure 2 shows gridigrator timing results when cell size is not ignored for the data sets in figures 4-6. Note that gridigration time increases linearly with data set size. Figure 3 is a comparison of gridigration time on the blunt-fin where the deposition function varies. Note that gridigration time is substantially longer when the distance between data nodes must be calculated (the Integrate PS case).

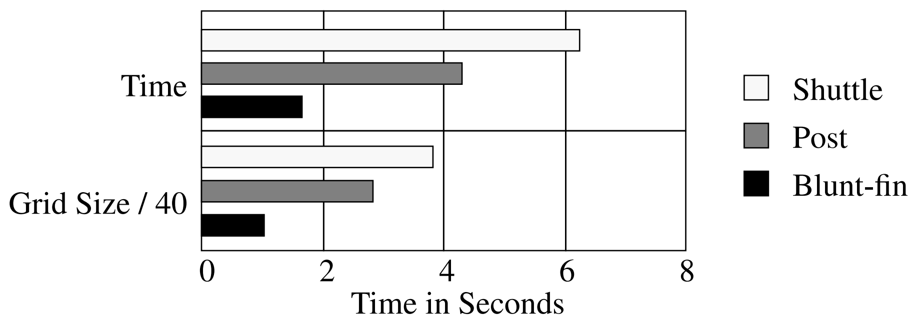


**Figure 2: Time and Grid Size**

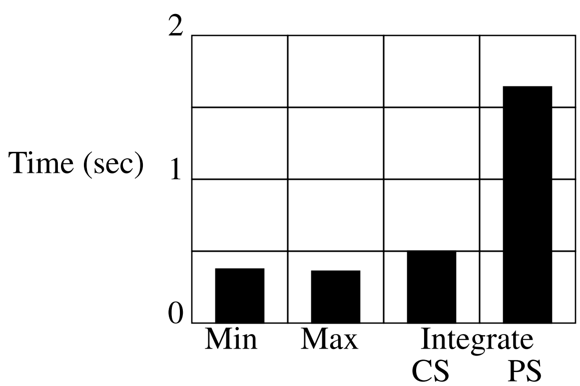


**Figure 3: Times for Different Deposition Functions**

CS stands for computation space, where *LineLength* in equation 1 is set to 2. PS stands for physical space where the actual length of intra-node grid lines is used in equation 1.

The blunt-fin is 40x32x32. Gridigration examined the pressure field. S. Uselton's RCVR software examined the density field. The post is 38 x 76 x 38. The shuttle is 80 x 63 x 45. All data is calculated but only 55 x 63 x 45 is displayed since the rest of the field is uninteresting.

**Discussion**

Gridigration is an interesting, quick and dirty variation on current volume rendering techniques. Interpretation of gridigrator images is difficult. The author will usually look at three of the six grid faces output. It is sometimes possible to determine the location of features by comparing the feature's image on each of the three faces. Figure 6 contains an annotation of one such case.

Because it is fast, gridigration may be useful for tuning transfer functions before paying the price of RCVR, at least when the relationship between deposition functions and transfer functions is clear.

The x-ray imaging analogy is potentially very useful. Software that casts rays in straight lines onto flat detectors (plates) could significantly reduce difficulties in interpretation. If such a rendering were done along each of the three axis (x, y, and z), the resulting plates could be moved along their respective normals to intersect regions of interest. It would be easier to detect such features with flat detectors and straight rays than with gridigration as in figure 6. Such software would, of course, be much slower than gridigration. However, compared with traditional RCVR fewer rays need be cast. A smaller performance improvement is achieved by casting rays only along axis rather than at arbitrary angles. If the detector is a two dimensional, structured grid, the detector may be temporarily coarsened to implement adaptive refinement to improve interactive response. Integration with other polygon/line/point based visualization is as trivial as with gridigration.

**Acknowledgments**

J. Krystynak wrote the first, limited gridigrator software as a FAST [Banc90] module and invented the name. S. Uselton suggested the integration mode where grid cells are assumed to be the same size to bring out details where the grid is concentrated. Uselton also provided RCVR workstation performance data and gave invaluable advice on this paper. D. Asimov reviewed the paper. This work was supported under NASA contract NAS2-12961.

**References**

[Banc90] G. Bancroft, F. Merritt, T. Plessel, P. Kelaita, R. McCabe, A. Globus, "FAST: A Multi-Processing Environment for Visualization of CFD," *Proc.Visualization '90*, IEEE Computer Society, San Francisco (1990).

[Chal92] J. Challinger, "Parallel Volume Rendering for Curvilinear Volumes,", submitted for publication.

[Darm91] D. Darmofal, R. Haimes, "Visual Feature Identification for 3-D Data Sets," AIAA-91-1583-CP, *Proc. AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, Hawaii, June 24-27, 1991.

[Hung85] C.H. Hung, P.G. Buning, "Simulation of Blunt-Fin-Induced Shock-Wave and Turbulent Boundary-Layer Interaction," *J. Fluid Mech.* (1985), Col. 154, pp. 163-185.

[Krue90] W. Krueger, "The Application of Transport Theory to Visualization of 3D Scalar Data Fields," *Proc. Visualization '90*, San Francisco, IEEE Computer Society Press. (1990).

[Levo88] M. Levoy, "Rendering Mixtures of Geometric and Volumetric Data," Technical Report 88-052, Computer Science Department, University of North Carolina at Chapel Hill, December, 1988.

[Rizk85] Y.M. Rizk, S. Ben-Shmuel, "Computation of the Viscous Flow Around the Shuttle Orbiter at Low Supersonic Speeds," *AIAA 23rd Aerospace Sciences Meeting*, Jan. 14-17, (1985) Reno, Nevada, AIAA-85-0168.

[Roge86] S. Rogers, D. Kwak, and U. Kaul, "A Numerical Study of Three-Dimensional Incompressible Flow Around Multiple Posts," American Institute of Aeronautics and Astronautics, paper AIAA-86-0353.

[Stro91] B. Stroustrup, *The C++ Programming Language*, second edition, Addison-Wesley Publishing Company.

[Upso89] C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, A. van Dam, "The Application Visualization System: A Computational Environment for Scientific Visualization," *IEEE Computer Graphics and Applications*, July 1989, pp. 30-41.

[Usel91] S. Uselton, "Volume Rendering for Computational Fluid Dynamics: Initial Results," Report RNR-91-026, September 1991, NAS Systems Division, Applied Research Branch, NASA Ames Research Center.