Classifying Data for Scientific Visualization via Fiber Bundles

Werner Benger

Center for Computation & Technology,
Louisiana State University
Baton Rouge, LA-70803, Louisiana, USA
**E-mail: werner@cct.lsu.edu*
sciviz.cct.lsu.edu

The lack of a standard for the exchange data for scientific visualization is a big hurdle in the interoperability among applications. The requirement to develop support for many diverse file formats demands human resources that could otherwise be spent more efficiently on real research topics. The vision of a unified data model is to provide a common denominator for at least a wide range of data types. This article presents the treatment of frequently occurring data types within the context of a data model based on the mathematics of fiber bundles, casting data in a six-level hierarchy. This scheme allows the derivation of a taxonomy of data and operations on them, as exemplified in this presentation.

Keywords: Topology, Differential Geometry, Common Data Model, Scientific Visualization, Computational Fluid Dynamics, File Formats, Interoperability

1. Introduction

The quest for a common data model[1–3] becomes increasingly important with the need of data archival and interoperability across independently developed applications.[4,5] An early approach suggested using mathematics as a common language to follow data layouts, in particular fiber bundles,[1] which led to a highly regarded implementation in the OpenDX data explorer.

The fiber-bundle based data model implemented in OpenDX[6] inspired a redesign yielding a data organization into a hierarchy of six levels.[7] This model has been used successfully e.g. in fusing of datasets from observation and simulation describing hurricanes[8] and for describing huge, complex data from computational fluid dynamics.[9] The F5 file format and library[10] is a mapping of this data model to HDF5,[11] a powerful library and file format used in high-performance computing.

A similar approach is taken by CGNS.[12] The CGNS project originated in 1994 as a joint effort between Boeing and NASA, and has since grown to include many other contributing organizations worldwide. It is now controlled by a public forum known as the CGNS Steering Committee. Same as F5, the CGNS model aims at providing a common data model and file format for data exchange among independent application. CGNS is recommended by the American Institute of Aeronautics and Astronautics (AIAA). While much of the standard and the software is applicable to computational field physics in general, its principal target of CGNS is data associated with compressible viscous flow (i.e., the Navier-Stokes equations).

In contrast, the F5 model – inspired by the requirements of general relativity – is even more general, at the price of not being equally advanced and detailed. The purpose of this article is to demonstrate the powerful expressiveness of the F5 model by describing how various common grid types are described in this model.

2. Data Taxonomy in the 6-Level Fiber Bundle Data Model

The data model used here casts data into a non-cyclic graph of six levels. Each level has a specific meaning, they are called the `Slice, Grid, Skeleton, Representation, Field` and `Fragment` level. A detailed description is given elsewhere,[7,9,13] here the terminology of this data model is employed to build a taxonomy of different data types occurring in scientific visualization.

Each of the six levels models a property of the data set as an entity. Generally, an "object" is the set of all data that is contained in the data model graph under a certain level. For instance, a Grid object refers to a collection of data sets describing a submanifold within a parameter space. Grid objects can be named arbitrarily. Depending on the existence and combinations of level beyond the Grid level, an unlimited number of Grid data types can be constructed. These properties are not necessarily mutually exclusive. A certain Grid may thus reside in more than one category in the taxonomy as presented here.

For each of the discussed case, a table will be provided with example entries. Actual data (arrays of native types such as float or integer values, or tuples of such) can only be attached to Field objects. In the following tables we use $\{x^\mu\}$ to express coordinates; $[i^k]$ for an index set, which are k integers given per element; $[i^?]$ for a varying number of integers given per element; and $p, \varrho, \vec{v}$ to express arbitrary scalar or vector field data. For Fragments, we will use $\emptyset$ to denote that a field is contiguous, and $\{\}$ to

denote that the field exists in separate parts.

Skeleton objects are classified via a pair of dimensionality and index depth, plus an optional refinement level (which is described in 2.6). Typical skeleton objects are:

Classification	Grid property
$(2,0)$	vertices within 2D manifold
$(3,0)$	vertices within 3D manifold
$(1,1)$	edges
$(2,1)$	faces, e.g. triangles
$(3,1)$	cells, e.g. tetrahedrons
$(1,2)$	a set of edges
$(2,2)$	a set of faces
$(3,2)$	a cell complex (set of cells)
$(3,3)$	a set of cell complexes
$(1,-1)$	coordinate coefficients

A collection of Field objects can be attached to a Representation object. Such reside within a Skeleton object and are identified by a chart or another Skeleton object to which they refer. This is expressed in the following way:

$\{x,y,z\}$	cartesian coordinates in 3D
$\{r,\vartheta,\varphi\}$	spherical coordinates in 3D
$\rightarrow (3,0)$	relative to vertices in 3D
$\rightarrow (2,1)$	relative to faces
$\rightarrow (1,1)\vert_{T=1.0}$	relative to vertices at time $T=1.0$

2.1. *Point Sets*

The most simple case of a Grid object is a set of points. At every time step where data are given, there exists a Grid object of an arbitrary, user-defined name (which ideally describes the relevance of the data set in a meaningful way, such as referring to the physical situation it describes). This Grid object will carry a skeleton for the vertices with a representation in cartesian coordinates, where a coordinate fields resides. Any additional fields can be defined on the vertices as well. The Skeleton will be of type $(2,0)$ if the point set is known to be constrained within a two-dimensional manifold, e.g. when describing ants running on a sphere.

Slice	T=0.0, ...	T=0.0, ...
Grid	"GalaxySimulation"	"RunningAnts"
Skeleton	$(3,0)$	$(2,0)$
Representation	$\{x,y,z\}$	$\{\vartheta,\varphi\}$,$\{x,y,z\}$
Fields	$\{x^\mu\}$	$\{x^\mu\}$
Fragments	$\emptyset$	$\emptyset$

The coordinates of a two-dimensional point set may still be three-dimensional because a two-dimensional set of points might be embedded in 3D space. In such a case, we might actually want two coordinate representations. Any true scalar fields may be shared among these representations (reference-counter pointers, symbolic links in a file system or HDF5), whereas vector and tensor quantities need to be transformed, i.e. require separate storage space.

2.2. *Lines*

A *curve* in differential geometry refers to a map $\mathbb{R} \to M$, a *line* to the image of a curve, which is a one-dimensional submanifold. Within this terminology, a curve is a parameterized line, a line is an equivalence class of curves. This definition must not be confused with the terminology in Euclidean geometry, where a line refers to a "straight curve". In consistency with the notion of e.g. "stream lines" or "integral lines", here the terminology of differential geometry is used.

2.2.1. *Line Sets*

A line is given by a point set with connectivity information defined on the vertices. Such connectivity is provided by a set of indices, which defines a single line. A set of such index sets constitutes a one-dimensional skeleton of depth one - category $(1,1)$ - which is represented relative to the vertices by a set of index arrays:

Slice	T=0.0, ...	
Grid	"Streamlines"	
Skeleton	$(3,0)$	$(1,1)$
Representation	$\{x,y,z\}$	$\to (3,0)$
Fields	$\{x^\mu\}$	$[i^?]$
Fragments	$\emptyset$	$\emptyset$

Additional Fields may be defined on the $(3,0)$ skeleton as well as on the $(1,1)$ skeleton. Fields on the $(3,0)$ skeleton provide one value per vertex,

fields on the $(1,1)$ skeleton provide one value per line. The size of the $(1,1)$ skeleton gives the number of lines. The number of vertices is flexible per line; if it is the same for all lines, we may call this line set to be *trivial*.

2.2.2. *Trajectories*

A trajectory is a line that traverses multiple time steps. In contrast to a line, its elements do not reside on a single Grid instance, but it is implicitly defined via a point set that is given on more than one time slice. An operator can be defined that maps such a time-dependent point set to a set of lines, $2.1 \mapsto 2.2.1$. This is trivial when the number of points is constant over time. If this is not the case, e.g. when points are created and destructed over time (particle collisions etc.), then there needs to be information stored which particles split up (future representation) or merge/disappear (past representation). Such a future/past representation expresses a skeleton relative to another skeleton instance of the same type, but at another time (next/previous time step). For each vertex at T_1 it provides a set of vertex indices at T_2 that tells how this specific particle has evolved.

Slice	T=0.0		T=1.0	
Grid	"Stars"		"Stars"	
Skeleton	$(3,0)$		$(3,0)$	
Representation	$\{x,y,z\}$	$\rightarrow (3,0)\|_{T=1.0}$	$\{x,y,z\}$	$\rightarrow (3,0)\|_{T=0.0}$
Fields	$\{x^\mu\}$	$[i^?]$	$\{x^\mu\}$	$[i^?]$
Fragments	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

If the number of points remains constant in time, we may assume the future and past representations to be identity maps and omit them. The mapping from trajectories to a line set is non-trivial if there are future/past representations.

2.3. *Surfaces*

The term "surface" refers to two-dimensional topological spaces, in particular two-dimensional manifolds. They are described at minimum by a set of vertices, a $(2,0)$-skeleton, and a set of faces, a $(2,1)$-skeleton represented in the vertices.

A *triangular surfaces* is constructed from triangles by a set of faces where each face has exactly three vertex indices. Similarly, *quad-based surfaces* are constructed from faces of constantly four indices, *irregular surfaces* from faces of varying number of indices.

Slice	T=0.0	
Grid	"Geometry"	
Skeleton	$(2,0)$	$(2,1)$
Representation	$\{x,y,z\}$	$\rightarrow (2,0)$
Fields	$\{x^\mu\}$	$[i^3]$
Fragments	$\emptyset$	$\emptyset$

Fields defined on the $(2,0)$ skeleton carry information per vertex, fields on the $(2,1)$ skeleton carry information per triangle, or cell type (in general).

2.4. *Regular Grids*

A Grid describing an n-dimensional manifold is called regular if each vertex has two neighbors in each dimension, i.e. 2^n neighbors all together (except boundary vertices). In 3D, each vertex has eight neighbors. Given the dimensionality, all edge, face and cell information is known already and does not need to be explicitly stored. Regular Grids are thus described by a specific key word or special data type that identifies this Grid as being regular. Since this regularity is expressed by a tensor product of the index space for each coordinate, objects which support multidimensional indexing will be denoted by attaching the symbol "$\otimes$", in contrast to arrays which only support linear indexing.

2.4.1. *Uniform, Rectilinear and Curvilinear Grids*

A Grid describing an n-dimensional manifold is *uniform* if it is regular and there exists a coordinate system where the coordinates are equidistant. The property of being "uniform" is thus coordinate-dependent; it is rather a property of a Grid's representation than of the Grid itself. It is specified by providing information about how to compute the vertex locations in the specific coordinate system, e.g. via two points or a point P_0 and a tangential vector $\vec{d}$. This will require hardly any storage space, the real data (for instance some scalar fields ϱ, p) reside in the fields given in the respective representation.

A *rectilinear* Grid is a n-dimensional regular Grid with the vertex coordinates given by the tensor product of n arrays, each array referring to one coordinate. Similar to the uniform Grid, a the property of being rectilinear is coordinate-dependent. A Grid which is uniform or rectilinear in one coordinate system, is most likely *curvilinear* in another coordinate system. In a curvilinear Grid, the vertex coordinates given explicitly as a (multidimensional) dataset.

	Uniform	Rectilinear	Curvilinear
Slice	T=0.0, ...	T=0.0, ...	T=0.0, ...
Grid	"UniformData"	"RectilinearData"	"CurvilinearData"
Skeleton	$(3,0)^{\otimes}$	$(3,0)^{\otimes}$	$(3,0)^{\otimes}$
Repr.	$\{x,y,z\}$	$\{x,y,z\}$	$\{x,y,z\}$
Fields	$\{P_0,\vec{d}\}$, $\varrho^{\otimes},p^{\otimes}$	$\{x\otimes y\otimes z\}$, $\varrho^{\otimes},p^{\otimes}$	$\{x^{\mu}\}^{\otimes}$, $\varrho^{\otimes},p^{\otimes}$
Fragments	$\emptyset$	$\emptyset$	$\emptyset$

Edges $(1,1)^{\otimes}$, faces $(2,1)^{\otimes}$ and voxels $(3,1)^{\otimes}$ may coexist on the same Grid as separate skeletons and e.g. carry cell-centered variables as they are commonly in use for computational fluid dynamic simulations.

2.4.2. *Multiblock and Multipatch*

A *block* denotes a fragment of a regular grid. For instance, if some computational simulation is carried out in parallel, each node of a computational cluster may output a block as a submanifold of the entire computational domain. A Grid is of *multiblock* type if it is regular and the fields given on it are fragmented. Each fragment carries the "regularity" property. If the Grid is uniform, then it follows that it will also be globally regular, and uniform coordinates can be given once for all blocks. A Grid is of *multipatch* type if it is a multiblock grid where the coordinates of each block are curvilinear. Such a grid is not necessarily globally regular, thus its vertex skeleton does not necessarily carry the regularity property.

	Multiblock	Multipatch
Slice	T=0.0, ...	T=0.0, ...
Grid	"Decomposed"	"Multipatched"
Skeleton	$(3,0)^{\otimes}$	$(3,0)$
Representation	$\{x,y,z\}$	$\{x,y,z\}$
Fields	$\{P_0,\vec{d}\}$	
Fragments	$\{\varrho^{\otimes},p^{\otimes}\}$	$\{\{x^{\mu}\}^{\otimes},\varrho^{\otimes},p^{\otimes}\}$

2.4.3. *Multispectral Grids*

The technique of spectral reduction is a decimation scheme that allows one to simulate large systems on uniformly-coarsened spectral grids. The multispectral method uses a hierarchy of differently-coarsened grids in Fourier space.The Grid is regular in this case, but the coordinates are given via coefficients that allow computation. We may express coordinates to be sec-

ondary information here by introducing a "negative" index depth, leading to a skeleton that allows one to store these coefficients. This "coefficient" space carries its own dimensionality, for instance it will be three-dimensional if it yields three-dimensional coordinates via fourier transformation; this is expressibly via a $(3,-1)$ skeleton carrying a data set $\tilde{f}^{\mu}$ relative to cartesian coordinates (representations in other coordinate systems may require a different set of coefficients):

Slice	T=0.0, ...	
Grid	"MultispectralGridded"	
Skeleton	$(3,-1)$	$(3,0)^{\otimes}$
Representation	$\{x,y,z\}$	$\{x,y,z\}$
Fields	$\tilde{f}^{\mu}$	$\varrho^{\otimes}, p^{\otimes}$
Fragments	$\emptyset$	$\emptyset$

2.5. *Volumetric Meshes*

Unstructured Cell Data (UCD) provide explicit data sets for each skeleton, with edges $(1,1)$, faces $(2,1)$, cells $(3,1)$ given relative to the vertices on a $(3,0)$ skeleton. The edges per face are stored in the $(2,1)$ (face) skeleton relative to the edges, i.e. in the $\rightarrow (1,1)$ representation. Similarly, the faces per edge can be stored in the $(1,1)$ edge skeleton, represented relative to the faces $\rightarrow (1,1)$.

The following table specifies the skeletons and representations for a three-dimensional mesh constructed from triangles, where information about the edges is stored as well. Each edge refers to two, each face to three vertices. Also, each face refers to three edges, and though in this case of a three-dimensional triangulation each edge may have more than one face (within a two-dimensional manifold, there would be exactly two faces per edge):

	coords	edges	faces/edge	face	edges/face
Slice	T=0.0, ...				
Grid	"UnstructuredGrid3D"				
Skeleton	$(3,0)$	$(1,1)$		$(2,1)$	
Repr.	$\{x,y,z\}$	$\rightarrow (3,0)$	$\rightarrow (2,1)$	$\rightarrow (3,0)$	$\rightarrow (1,1)$
Fields	$\{x^{\mu}\}$	$[i^2]$	$[i^?]$	$[i^3]$	$[i^3]$
Fragments	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

We end up with three Skeletons and five Representation objects in this case, not yet including cell information, such as specification of tetrahedrons. The

cell type of a tetrahedral mesh will consist of four indices, same as for a surface build from rectangles. However, while the cell type is the same, the dimensionality of their skeletons will be different, which allows one to distinguish these Grid types.

	quad surface		tetrahedral mesh	
Slice	T=0.0, ...		T=0.0, ...	
Grid	"quadmesh"		"tetramesh"	
Skeleton	$(2,0)$	$(2,1)$	$(3,0)$	$(3,1)$
Representation	$\{x,y,z\}$	$\rightarrow (2,0)$	$\{x,y,z\}$	$\rightarrow (3,0)$
Fields	$\{x^\mu\}$	$[i^4]$	$\{x^\mu\}$	$[i^4]$
Fragments	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

2.6. *Hierarchical Grids - Multiresolution Meshes*

The described Grid types are sufficient to model a particular discretization of a physical manifold. Various numerical schemes cover a physical domain by multiple discretizations with different spatial resolutions, known as multi-resolution methods or adaptive mesh refinement.[14] Basically such a hierarchy could be defined as a sequence of Grid objects. To avoid the clutter introduced by many grids, the Skeleton classification allows a third integer parameter in addition to the dimensionality and index depth. This third parameter, the "refinement level" does not yield new Grid types, but allows the formulation of refinements within a single Grid object.

2.7. *Color Spaces*

Colors are frequently modeled as elements of In the context of the Fiber Bundle data model, We may treat color spaces similar to a coordinate system. A dataset may then be represented in this color space. A set of color space elements on a regular two-dimensional grid, i.e. a curvilinear grid in color space, then becomes an "image" (or texture). A movie - a sequence of images - is easily modeled through the intrinsic time-dependency supported by the data model, whereby each frame is associated with physical time (rather than just frame number) and may be related directly to the data itself. Multi-resolution images are supported through the various refinement levels of skeletons.

A colormap (also known as transfer function, e.g. for volume rendering) corresponds to the notion of a chart. A scalar field may the be represented in such a colormap like a vector field can be represented in various coordinate systems, yielding different images of the same data set.

3. Summary

In this article, the expressiveness of the six-leveled fiber bundle data model is demonstrated via the detailed formulation of frequently used data types. The building blocks, inspired by the mathematical background of fiber bundles, allow one to model certain properties of a general dataset, leading to a taxonomy which is suitable to ease interoperability among applications and communication across different scientific domains. Portions of this work were supported by NSF/EPSCoR Award No. EPS-0701491 (CyberTools).

References

1. D. M. Butler and M. H. Pendley, *Computers in Physics* **3**, 45(sep/oct 1989).
2. R. B. Haber, B. Lucas and N. Collins, A data model for scientific visualization with provisions for regular and irregular grids, in *VIS '91: Proceedings of the 2nd conference on Visualization '91*, (IEEE Computer Society Press, Los Alamitos, CA, USA, 1991).
3. P. Moran, *Field model: An object-oriented data model for fields*, tech. rep., NASA Ames Research Center (2001).
4. S. Nativi, B. Blumenthal, T. Habermann, D. Hertzmann, R. Raskin, J. Caron, B. Domenico, Y. Ho and J. Weber, Differences among the data models used by the geographic information systems and atmospheric science communities, in *Proceedings American Meteorological Society - 20th Interactive Image Processing Systems Conference*, 2004.
5. M. T. Dougherty, M. J. Folk, E. Zadok, H. J. Bernstein, F. C. Bernstein, K. W. Eliceiri, W. Benger and C. Best, *Communications of the ACM (CACM)* **52(10)**, 42(October 2009).
6. L. A. Treinish, Data explorer data model `http://www.research.ibm.com/people/l/lloydt/dm/dx/dx_dm.htm`(March, 1997).
7. W. Benger, Visualization of general relativistic tensor fields via a fiber bundle data model, PhD thesis, FU Berlin, (Berlin, Lehmanns Media, 2004).
8. W. Benger, S. Venkataraman, A. Long, G. Allen, S. D. Beck, M. Brodowicz, J. MacLaren and E. Seidel, Visualizing katrina - merging computer simulations with observations, in *Workshop on state-of-the-art in scientific and parallel computing, Umeå, Sweden, June 18-21, 2006*, (Lecture Notes in Computer Science (LNCS), Springer Verlag, 2006).
9. W. Benger, M. Ritter, S. Acharya, S. Roy and F. Jijao, Fiberbundle-based visualization of a stir tank fluid, in *WSCG 2009, Plzen*, 2009.
10. W. Benger, F5 - fiberbundle hdf5 `http://www.fiberbundle.net/`, (2005).
11. The HDF Group, Hierarchical data format version 5 `http://www.hdfgroup.org/HDF5`, (2000-2009).
12. CGNS Steering Committee, *CFD General Notation System*. `http://www.cgns.org`.
13. W. Benger, *New Journal of Physics* **10** (2008).
14. M. J. Berger and J. Oliger, *J. Comput. Phys.* **53**, 484 (1984).