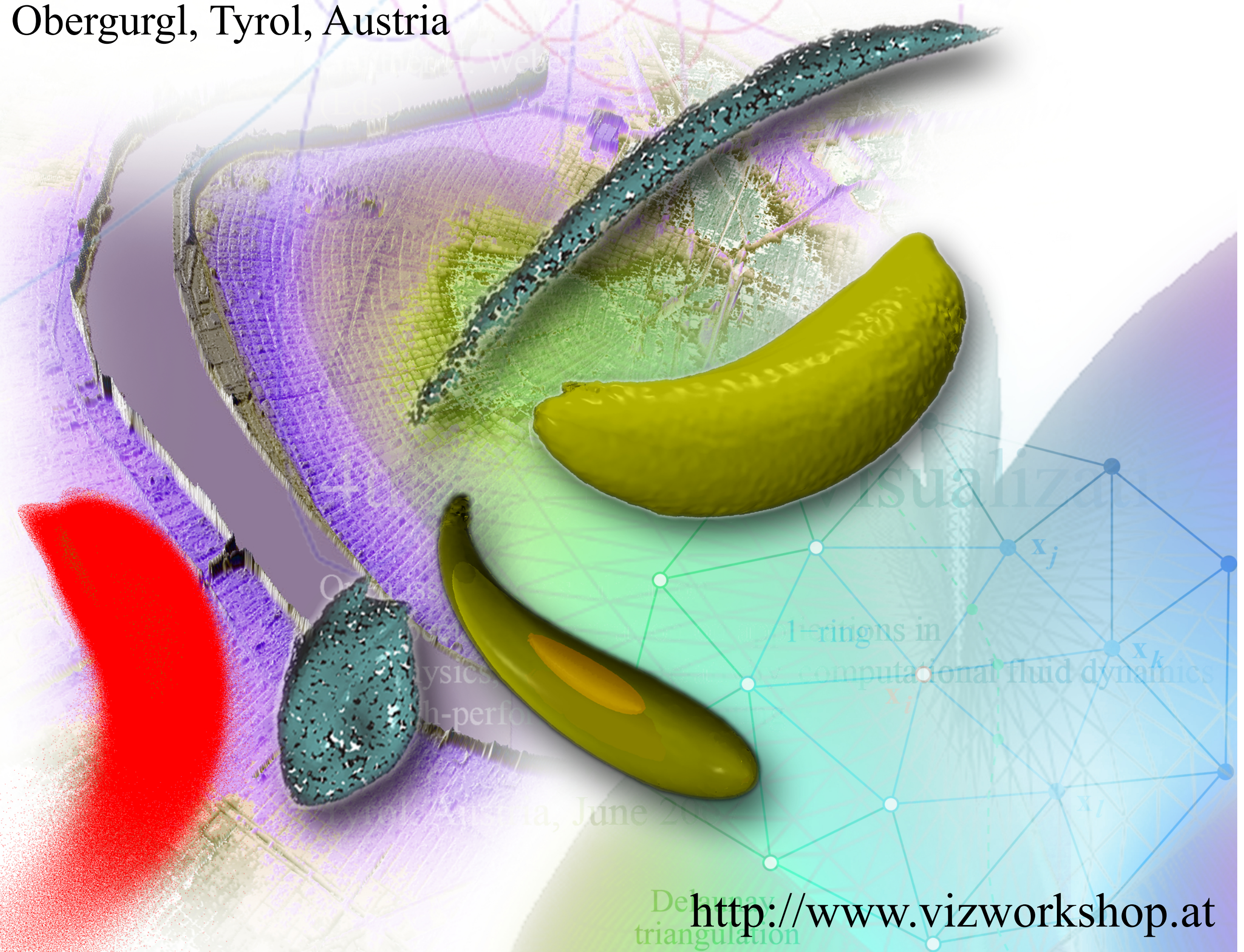
Werner Benger
René Heinzl
Wolfgang Kapferer
Wolfram Schoor
Mayank Tyagi
Shalini Venkataraman
Gunther H. Weber
(Eds.)

Proceedings of the 4th High-End Visualization Workshop

Open issues in visualization
with special concentration on applications in
astrophysics, numerical relativity, computational fluid dynamics
and high-performance computing

June 18th- 22th, 2007
Obergurgl, Tyrol, Austria



http://www.vizworkshop.at

Bibliographische Information der Deutschen Bibliothek:
Die deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet über <**http://dnb.ddb.de**> abrufbar.

Werner Benger, CCT/LSU, Baton Rouge, Louisiana, USA
René Heinzl, IuE, TU Wien, Vienna, Austria
Wolfgang Kapferer, UIBK, Innsbruck, Austria
Wolfram Schoor, IFF, Magdeburg, Germany
Mayank Tyagi, CCT/LSU, Baton Rouge, Louisiana, USA
Shalini Venkataraman, CCT/LSU, Baton Rouge, Louisiana, USA
Gunther H. Weber, LBL, California, USA
(Eds.)

Proceedings of the
4th High-End Visualization Workshop

June 18th- 22th, 2007
Obergurgl, Tyrol, Austria
Berlin, 2007, Lehmanns Media-LOB.de
ISBN 978-3-86541-216-4
Druck und Verarbeitung: Docupoint Magdeburg, www.docupoint-md.de

Cover page: composition made by Sebastian Friedrich (IFF) based on images from the contributions of:
- *Modeling of Non-Trivial Data-Structures with a Generic Scientific Simulation Environment, page 5*,
- *Direct Surface Extraction from Smoothed Particle Hydrodynamics Simulation Data, page 52*,
- *GPU Friendly Rendering of Large LIDAR Terrains, page 78*, and
- *Visualization of Polynomials Used in Series Expansions, page 139*

Preface

The High End Visualization Workshop has started out of an informal meeting among the working groups for numerical relativity at the Max-Planck Institute for Gravitational Physics (the Albert-Einstein Institute), the visualization department at the Zuse-Institute Berlin and the Institute for Astro- and Particle Physics at the University of Innsbruck. This meeting involving developers of scientific visualization on the one side and application scientists on the other side was very inspiring and successfull. It has been repeated in the following years and grown beyond its original group, then also including more institutions. With the main aspect on visualizations of numerical simulations, a special topic is featured each year, but at the same time participants are also asked to provide contributions outside of the strict focus area. This combination together with the informal atmosphere provided by the University Center in Obergurgl, Austria – at the high end of the Ötztal valley – has led to many continued collaborations.

With the 4th anniversary of this workshop it is the first time we issue printed proceedings. Scientific Visualization is an established field, but still has not reached all application scientists. We intend to discuss the reasons, from the perspective of the users as well as those from the developers. The contributions to this workshop are organized into articles relating to

- Visualization Infrastructure - general considerations of the environment where and how visualizations shall occur;
- Visualization Algorithms - specific algorithms to enable or improve the display of data, may it be quantitatively or qualitatively;
- Visualization Applications - utilization of visualizations for a specific application domain, or demands from an application domain.

We thank the Center for Computation & Technology at Louisiana State University (CCT), the University of Innsbruck (UIBK), the Fraunhofer Institute for Factory Operation and Automation (IFF), the Institute for Microelectronics at the Technical University Vienna (IuE), Lawrence Berkeley National Laboratory (LBL) and all the contributors, sponsors and supporters for their assistance to enable this event.

Werner Benger & the organizers

List of Contributions

Modeling of Non-Trivial Data-Structures with a Generic Scientific Simulation Environment

René Heinzl, Philipp Schwaha,
Carlos Giani, and Siegfried Selberherr
Institute for Microelectronics, TU Wien, Vienna, Austria
{heinzl|schwaha|selberherr}@iue.tuwien.ac.at

Abstract

Scientific computing requires efficient specification and handling of structure generation and associated modeling tasks. A generic scientific simulation environment was developed to ease these tasks by homogeneous specification of data structures in a dimensionally and topologically neutral way. The concept of fiber bundles is used to separate data structure issues from quantity management. To explain the introduced mechanisms the fiber notation of the Möbius stripe and the Hopf fibration are given with our environment.

1 Introduction

The approach taken with our generic scientific simulation environment (GSSE [Heinzl et al., 2006a]) extends concept based programming of the STL to arbitrary dimensions similar to the grid algorithms library (GrAL [Berti, 2000]). In addition, an efficient notation for data structures and mathematical concepts is obtained. The main difference to GrAL is the introduction of the concept of fiber bundles [Butler & Bryson, 1992], which separates the mechanism of application design into base space and fiber space properties. The base space is modeled by a CW-complex [Benger, 2004], whereas the fiber space is modeled by a generic data accessor mechanism, similar to the cursor and property map concept [Abrahams et al., 2003].

1.1 Related Work

In the following some work related to our approach is briefly presented.

The **Grid Algorithms Library, GrAL** [Berti, 2000] was one of the first contributions to the unification of data structures of arbitrary dimension for the field of scientific computing. A common interface for grids with a dimensionally and topologically independent way of access and traversal was designed.

The **Fiber Library** [Benger, 2004] implements several mechanisms for base and fiber space properties. The base space is modeled by a CW-complex. For the fiber space several mechanisms are offered to handle various data models with a minimum of data specification.

For the **GSSE** we have developed a consistent data structure interface based on algebraic topology and order theory. With this interface specification we can make use of several libraries, e.g., GrAL. Another important advantage is that the GSSE is mainly built for a library centric application design, which means that application design is greatly supported.

1.2 Theory of Fiber Bundles

In this section we briefly overview the concept of fiber bundles [Benger, 2004] as description for data structures of various dimensions and topological properties.

Let E, B be topological spaces and $\pi : E \to B$ a continuous map. Then (E, B, π) is called a **fiber bundle**, if there exists a space F such that the union of the inverse images of π of a neighborhood $U_b \subset B$ of each point $b \in B$ is homeomorphic to $U_b \times F$, whereby the homeomorphism ϕ has to be such that the projection pr_1 of $U_b \times F$ yields U_b again and the following diagram commutes:

$$\begin{tikzcd}
\pi^{-1}(U_b) \arrow[r, "\phi"] \arrow[d, "\pi"'] & U_b \times F \arrow[dl, "pr_1"] \\
U_b &
\end{tikzcd}$$

Figure 1: Structure of fiber bundles.

We have introduced [Heinzl et al., 2006b] a common interface for various data structures of arbitrary dimension. The advantages of our approach are similar to the cursor and property map but differ in several details as given in Table 1. The similarity is that both approaches can be implemented independently. The main difference is that the fiber bundle approach equips the fiber space with more structure, e.g., storage of more than one value corresponding to the traversal position, and preservation of neighborhoods. These features are especially useful in the area of scientific computing, where different data sets have to be managed, e.g., multiple scalar or vector values on vertices, faces, or cells.

	cursor and property map	fiber bundles
isomorphic base space	no	yes
traversal possibilities	STL iteration	cell complex
traversal base space	yes	yes
traversal fiber space	no	yes
data access	single data	topological space
fiber space slices	no	yes

Table 1: Cursor and property map compared to the GSSE approach.

As can be seen the concept of the cursor and property map can be extended to the fiber bundle approach with additional properties and mechanisms.

1.3 Homogeneous Interface for Data Structures

We briefly introduce parts of the interface specification for data structures. A full reference is given in [Heinzl et al., 2006b]. A formal concise definition of data structures can therewith be derived as presented in Table 2.

data structure	cell dimension	cell topology	complex topology
array/vector	0	simplex	global
SLL/stream	0	simplex	local(2)
graph	1	simplex	local
grid	2,3,4,..	cuboid	global
mesh	2,3,4,..	simplex	local

Table 2: Data structure classification scheme based on the dimension of cells, the cell topology, and the complex topology.

Examples of higher dimensional complexes for C++ with the GSSE notation are given next.

```
cell_type<3, simplex>        cell_s; // tetrahedron
cell_type<4, cuboid>         cell_c; // hypercube
complex_t<cell_s, local>   > base_s; //{1}
complex_t<cell_c, global>  > base_s; //{2}
```

Here {1} describes an unstructured tetrahedral mesh and {2} describes a cell complex based on hypercubes. In the following the poset structure and a possible rendering of a four-dimensional simplex are presented. The poset structure demonstrates the traversal capabilities of the GSSE data structure interface.

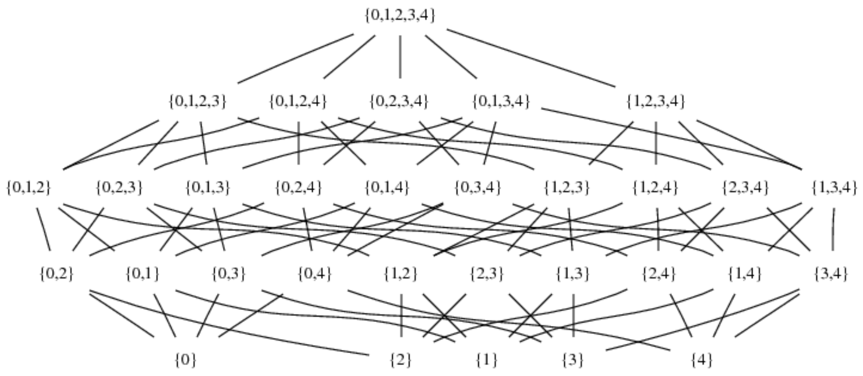


Figure 2: Poset of a four-dimensional simplex.

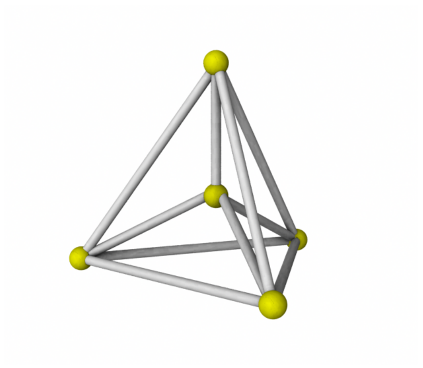


Figure 3: Rendering of a four-dimensional simplex.

The poset of the four-dimensional cube is omitted due to the large number of facets. Only one possible render image is given next:

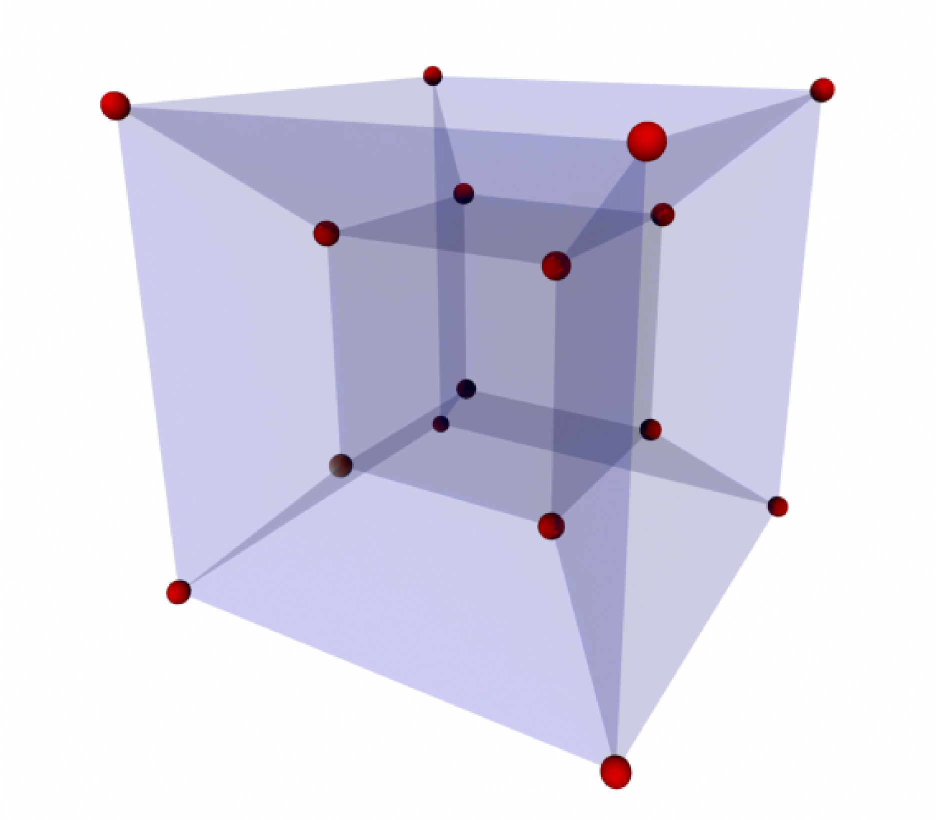


Figure 4: Rendering of a four-dimensional cube.

2 Non-Trivial Data Structures

Next to the already presented data structures which can be specified by CW-complex notation, several other non-trivial data structures exist. The embedded language of GSSE was extended to directly support the specification of several properties of data structures based on the structure of the fiber bundle approach with the corresponding base and fiber space. The topological structure of the corresponding space is given at compile-time, whereas the number of elements is given at run-time.

2.1 Möbius Stripe

The Möbius strip is a famous example of a non-trivial fiber bundle with $B = \mathbb{S}^1$ and $F = \mathbb{R}^1$ in an interval $I = [0,1]$. The total space E can only be written locally as the product of B and F. The trivial counterpart is the infinite cylinder $\mathbb{S}^1 \times \mathbb{R}^1$, where the total space can be written globally. The modeling with the GSSE is presented in the following code snippet with our modeling language embedded within C++.

```
// compile time
//
base_space<S<1> >              base_s;
fiber_space<I<1> >             fiber_s;
total_space<base_s, fiber_s>  total_s;

// run time
//
base_s    bs(100);
fiber_s   fs(0,1,f(bs));
```

The run-time part specifies the number of discretization points for the circle, whereas the fiber is specified with the interval length and the `f(bs)` which is a simple function object describing the functional dependence of the interval of the fiber space. The following picture depicts a simple OpenGL rendering.

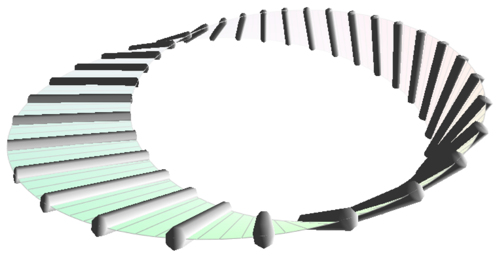


Figure 5: Visualization of the Möbius stripe based on the GSSE specification.

3 Hopf Fibration

Another example of an efficient fiber bundle notation is the Hopf fibration. The base space is modeled by S^2 (the hull of a sphere), the fiber space by S^1, and therewith the total space is modeled by a S^3 (hypersphere). The Hopf fibration can be expressed very efficiently within the GSSE.

```
// compile time
//
base_space<S<2> >              base_s;
fiber_space<S<1> >             fiber_s;
total_space<base_s, fiber_s>  total_s;

// run time
//
base_s    bs(100,100);
fiber_s   fs(100);
```

The number of discretization points is 100 in this case, which is feasible for visualization. The visualization of the Hopf fibration [Lyons, 2003] is more complex than the Möbius stripe due to the higher dimensional base space and fiber space. We use a stereographic projection where the S^3 can be seen in the three-dimensional space. The S^1 fibers are therewith projected as circles in $\mathbb{R}^3$ and depicted in Figure 6 where Figure 7 presents two possible projections.

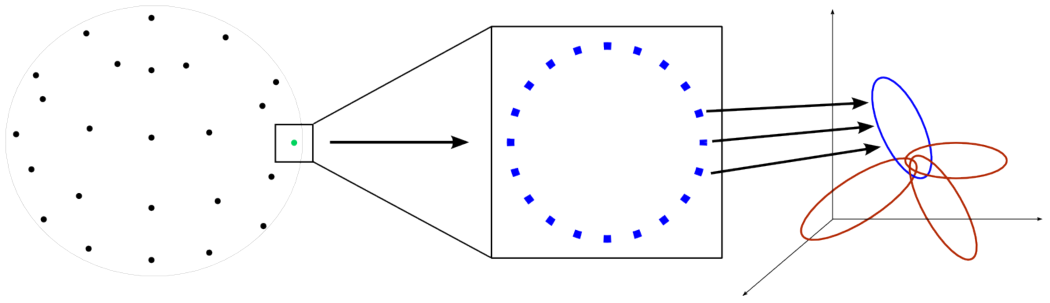


Figure 6: Projection sequence from $S^2 \rightarrow \mathbb{R}^3$ (left), $S^1 \rightarrow \mathbb{R}^4$ (middle), and finally into $\mathbb{R}^3$.

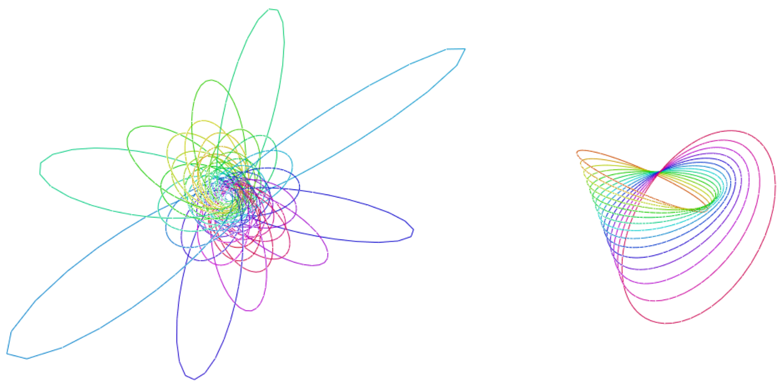


Figure 7: Left: a possbile stereographic projections of a discretized Hopf fibration. Right: visualization of a single circle of the base space.

4 Conclusions

The theory of fiber bundles separates the properties of the base space and the fiber space greatly. By formal specification of a common data structure interface based on algebraic topology a wide variety of generic environments can be used interoperably. GSSE supports the efficient notation of fiber bundles and accomplishes therewith a powerful mechanism to specify non-trivial data structures efficiently.

References

[Abrahams et al., 2003] Abrahams, D., Siek, J., & Witt, T. (2003). *New Iterator Concepts*. Technical Report N1477 03-0060, ISO/IEC JTC 1, Information Technology, Subcommittee SC 22, Programming Language C++.

[Benger, 2004] Benger, W. (2004). *Visualization of General Relativistic Tensor Fields via a Fiber Bundle Data Model*. PhD thesis, Freie Universität Berlin.

[Berti, 2000] Berti, G. (2000). *Generic Software Components for Scientific Computing*. PhD thesis, Technische Universität Cottbus.

[Butler & Bryson, 1992] Butler, D. M. & Bryson, S. (1992). Vector Bundle Classes From Powerful Tool for Scientific Visualization. *Computers in Physics*, 6, 576–584.

[Heinzl et al., 2006a] Heinzl, R., Spevak, M., Schwaha, P., & Grasser, T. (2006a). A High Performance Generic Scientific Simulation Environment. In *Proc. of the PARA Conf.* (pp.$\tilde{6}$1). Umea, Sweden.

[Heinzl et al., 2006b] Heinzl, R., Spevak, M., Schwaha, P., & Selberherr, S. (2006b). A Generic Topology Library. In *Library Centric Sofware Design, OOPSLA* (pp. 85–93). Portland, OR, USA.

[Lyons, 2003] Lyons, D. (2003). An Elementary Introduction to the Hopf Fibration. *Mathematics Magazine*, 76(2), 87–98.

Visualization Tools for Adaptive Mesh Refinement Data

Gunther H. Weber[1] and Vincent E. Beckner[1] and Hank Childs[2] and Terry J. Ligocki[1] and Mark C. Miller[2] and Brian Van Straalen[1] and E. Wes Bethel[1]

[1]Computing Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA

[2]Computing Applications and Research Department, Lawrence Livermore National Laboratory, Box 808, L-557, Livermore, CA 94551, USA

{GHWeber, VEBeckner}@lbl.gov, childs3@llnl.gov, TJLigocki@lbl.gov, miller86@llnl.gov, {BVStraalen, EWBethel}@lbl.gov

Abstract

Adaptive Mesh Refinement (AMR) is a highly effective method for simulations that span a large range of spatiotemporal scales, such as astrophysical simulations that must accomodate ranges from interstellar to sub-planetary. Most mainstream visualization tools still lack support for AMR as a first class data type and AMR code teams use custom built applications for AMR visualization. The Department of Energy's (DOE's) Science Discovery through Advanced Computing (SciDAC) Visualization and Analytics Center for Enabling Technologies (VACET) is currently working on extending VisIt, which is an open source visualization tool that accommodates AMR as a first-class data type. These efforts will bridge the gap between general-purpose visualization applications and highly specialized AMR visual analysis applications. Here, we give an overview of the state of the art in AMR visualization research and tools and describe how VisIt currently handles AMR data.

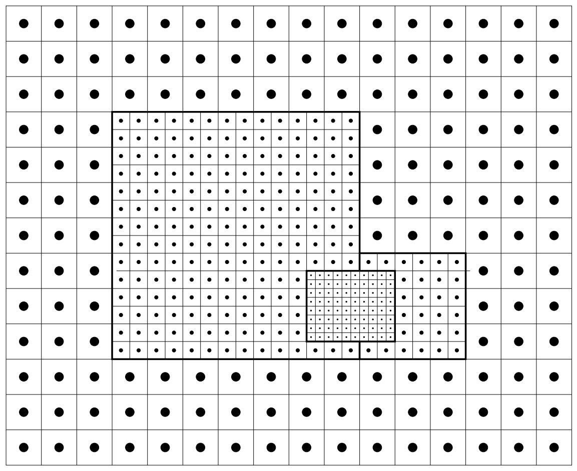


Figure 1: A simple Berger-Collela AMR hierarchy consisting of four patches organized in three hierarchy levels.

1 Introduction

Adaptive Mesh Refinement (AMR) techniques combine the compact, implicitly specified structure of regular, rectilinear with the adaptivity to changes in scale of unstructured grids. In this paper, we focus on block-structured, h-adaptive AMR techniques that represent the computational domain with a set of nested rectilinear grids or *patches* at increasing resolutions [Berger & Colella, 1989]. Figure 1 shows a simple example. Four regular patches are organized in three hierarchy levels. Grids belonging to a finer level are always completely enclosed by grids of the coarser levels.

Handling AMR data for visualization is challenging, since coarser information in regions covered by finer patches is superseded and replaced with information from these finer patches. During visualization it becomes necessary necessary to manage selection of which reslotuions are being used. Furthermore, it is difficult to avoid discontinuities at level boundaries, which, if not properly handled, lead to visible artifacts in visualizations.

While AMR data can be node centered, the majority of data sets we are currently visualizing use a cell centered format. This fact poses an additional challenge since many visualization algorithms expect data in a node centered format. Despite the growing popularity of AMR simulations, little research has been done in effective visualization of AMR data. Furthermore, there is a lack of tools that treat AMR as first-class data type. In this paper we give an overview of the current state of AMR research and describe ongoing work to extend VisIt to be one of the first mainstream visualization tools with "native" AMR support. Though other general visualization tools with AMR support exist, such as ParaView [Squillacote, 2006] and customized versions of Amira [Stalling et al., 2005], exist, we focus our discussion on VisIt due to its unique analytics capabilities.

2 Current State of AMR Visualization

2.1 Visualization of AMR Scalar Fields

Scalar quantities describe a variety of important physical characteristics such as temperature or pressure. Most simulations, including AMR simulations, include several scalar variables. Commonly used scalar data visualization techniques include slicing planes, isosurface extraction and direct volume rendering. Spreadsheets, which are somewhat related to slicing planes, provide direct access to data value and are valuable for debugging and extracting data for further analysis with a wider range of tools such as Matlab or paper and pencil.

2.1.1 AMR Visualization by Conversion to Other Types

Initial work on AMR visualization focused on converting AMR data to suitable conventional representations, which are subsequently used for visualization. [Norman et al., 1999] described a method for visualizing AMR data using is then visualized using standard unstructured grid techniques. Their method converts an AMR hierarchy into an unstructured grid composed of hexahedral cells. This resulting grid is then visualized utilizing standard AVS, IDL, and VTK algorithms. By converting AMR data to an unstructured mesh, the implicit definition of grid connectivity is lost. Overhead resulting from the required separate storage of grid structure results in poor performance and does not scale well to large AMR data sets. Furthermore, this approach prohibits using of the hierarchical nature of AMR data for efficient visualization algorithms. Recognizing these fundamental problems, Norman et al. continue by extending VTK to handle AMR grids as first-class data structure. They have yet to publish detailed descriptions of their techniques.

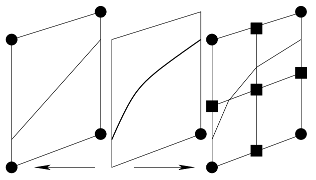


Figure 2: At the boundaries (center face in figure) between coarse (left face in figure) and fine levels (right face in figure), dangling nodes (black rectangles in the figure) occur in addition to vertices that are shared between grids (black circles in the figure). On cell faces, marching cubes approximates isosurfaces with line segments. This scheme can lead to cracks, as the actual contour (bold line on center face) is approximate by a line segment in the coarse level (left face) and a poly line in the fine level (right face).

2.1.2 Crack-free Isosurface Extraction

AMR data is particularly difficult to handle when visualizing data via isosurface extraction. This difficulty is due to the fact that AMR often uses a cell centered data format while the marching cubes algorithm [Lorensen & Cline, 1987], which is de-facto standard isosurface extraction algorithm in scientific visualization, expects values at the vertices of a grid. Furthermore, when extracting an isosurface using the marching cubes method, t-junctions will lead to visible cracks in an isosurface, even if dangling nodes have values that are consistent with the coarse level representation [Shekhar et al., 1996] (see Figure 2).

[Weber et al., 2001b, Weber et al., 2003a] developed a method that extracts crack-free isosurfaces from cell-centered AMR data by interpreting cell centers of each patch of the AMR hierarchy as the vertices of a new patch, which is the *dual grid* to the original patch. Within these dual grids, isosurfaces are extracted utilizing the standard marching cubes method. The use of dual grids leads to gaps between different levels of an AMR hierarchy. Weber et al. use a procedural scheme to fill these gaps with "stitch" cells (tetrahedra, pyramids, triangle prisms and deformed cubes) ensuring that this step produces no t-junctions. Subsequently, they extract isosurface portions within gaps between hierarchy levels utilizing the marching cubes methods by giving appropriate case tables for these new cell types.

[Fang et al., 2004] presented an alternate isosurface extraction approach for node-centered AMR data. Their main goal is preserving the original patch structure and "identity" of cells, enabling a user to determine to what particular patch cell a triangle of an isosurface belongs. Their method achieves this goal by extending refined patches until it is possible to assign values to dangling nodes that are consistent with interpolation results in the coarser level. Subsequently, they decompose coarse-level cells at the boundary to a finer level into a set of pyramids that connect the cell center with all boundary faces. For each "facet" of a subdivided face, i.e., a face at the boundary to a finer level, a separate pyramid is created, ensuring that marching cubes will not produce cracks in an extracted isosurface.

2.1.3 Volume Rendering

[Max, 1993] described sorting schemes for cells during volume rendering including one method specifically geared toward AMR data. [Ma, 1999] described and compared two approaches for rendering of structured AMR data using the PARAMESH framework. A PARAMESH hierarchy organizes grids as blocks in a quadtree (in 2-d space) or an octree (in 3-d space) structure. Inner nodes of this tree correspond to regions that need further refinement while leaf nodes specify a grid whose resolution is given by the current hierarchy level. Ma described two approaches for volume rendering of AMR data. One method resamples a hierarchy on an uniform grid at the finest resolution. The resulting grid is evenly subdivided and each part rendered in parallel on a separate processor. Partial images are combined using binary-swap composition.

A second method preserves the AMR structure. Individual blocks (leaves of the octree) are distributed among the processors in a round-robin fashion to achieve static load balancing. Since a block structure can lead to many small ray-segments, Ma buffers these segments into larger messages to decrease communication overhead. Individual blocks are rendered using ray-casting. Two sampling schemes are used: A simple approach using a fixed, constant sample distance and an adaptive approach that decreases sample distance in finer resolution blocks.

[Weber et al., 2001a] described an interactive, hardware accelerated volume rendering approach to generate previews of AMR data and a higher-quality software approach based on cell projection. Both approaches use data duplicated in coarser hierarchy levels as a less accurate approximation for the data in finer levels. The hardware-based approach uses a k-d-tree-like structure to partition an AMR hierarchy into blocks of homogeneous resolution and renders these blocks in back-to-front order. Based on view-dependent criteria (e.g., the number of pixels covered by a voxel) and a measured rendering time for the current frame, the traversal depth into the individual patches of the AMR hierarchy is chosen to achieve interactive rendering rates.

[Weber et al., 2001a] also described a software cell-projection-based approach to render AMR data sets in higher quality. While rendering a level of an AMR hierarchy, additional information is stored for each pixel that makes it possible to "replace" the contribution of those parts of the domain that are refined by another hierarchy level with a more accurate representation, supporting progressive rendering of AMR data sets. In later work, [Weber et al., 2001a] used the dual mesh and stitch cells introduced for isosurface extraction [Weber et al., 2003a] to define a C^0 continuous interpolation scheme and utilized this interpolation method in their progressive cell-projection rendering approach.

[Kreylos et al., 2002] described a framework for remote, interactive rendering of AMR data. The framework consists of a "lightweight" viewer and a renderer running on one or several remote machines. The method of Kreylos et al. "homogenizes" an AMR hierarchy, i.e. partitions it in blocks of constant resolution using a k-d tree. Resulting blocks of constant resolution are distributed among processors and rendered using either a texture-based hardware-accelerated approach or a software-based cell-projection renderer. Two distribution strategies are implemented: One strategy attempts to distribute cost evenly among processors, the other variant tries to minimize data duplication. [Weber et al., 2003b] built on this work and compared various AMR partitioning strategies for parallel volume rendering of AMR data.

[Kähler & Hege, 2002] introduced a method that partitions Berger-Colella AMR data into homogeneous resolution regions and visualizes it using texture-based hardware-accelerated volume rendering. Their partitioning scheme uses a heuristic that is based on assumptions concerning the placement of refining grid to minimize the number of constant-resolution blocks. Generally, this approach generates fewer blocks than the approach described by [Weber et al., 2001a] and the approach developed by [Kreylos et al., 2002]. Subdivision into a smaller number of blocks is beneficial when data is rendered on a single machine.

In later work, [Kähler et al., 2002] used a set of existing tools to render results of a simulation of a forming star using the framework developed by [Bryan, 1999]. They define camera paths within a CAVE environment using the *Virtual Director* virtual reality interface. Subsequently, they render animations of the AMR simulation utilizing their previously developed hardware-accelerated volume rendering approach [Kähler & Hege, 2002]. To enhance depth perception, rendered images are augmented with a background that is obtained by rendering a particle simulation of the formation of the early universe. Recently, [Kähler et al., 2006] implemented a GPU-based ray-casting approach for AMR data, which improves rendering quality considerably compared to slicing-based approaches and supports a more complex light model with wavelength dependent absorption.

By specifying a transfer function, and a range of isovalues, [Park et al., 2002] produced volume-rendered images of AMR data based on hierarchical splatting, see [Laur & Hanrahan, 1991]. Their method converts an AMR hierarchy to a k-d-tree structure consisting of blocks of constant resolution. Each node of this k-d tree is augmented with an octree. Octree and k-d-tree nodes contain a 32-bit field, where each bit represents a continuous range of isovalues. Using the k-d tree and the octree, regions containing values within the specified range are identified and rendered back-to-front using hierarchical splatting.

2.2 Visualization of Time-varying AMR Scalar Field Data

[Chen et al., 2003] introduced the *feature tree* that describes, how connected components of an isosurface change as one successively adds AMR refinement levels. Nodes in a tree represent connected components of an isosurface, with each level in the tree corresponding to a level in the AMR hierarchy. The resulting tree describes whether a connected component splits or merges when considering a finer hierarchy level. Using the resulting feature tree, it becomes possible to describe how individual isosurface components change over the course of time in a simulation.

[Kaehler et al., 2005] described a framework for visualization of time-varying AMR data. Their method addresses the problem that most AMR simulations update finer AMR patches more frequently than coarse patches. Considering two subsequent time steps, their interpolation scheme first ensures that both time-steps have the same refinement configuration, i.e., that each cell that is refined in one time step is also refined in the other time step. Values for cells that are not refined in the current time step but the other are obtained by interpolation. Subsequently, they define an interpolation scheme to compute intermediate values in regions that are covered by coarser level and thus, updated less frequently. In addition to this interpolation scheme, their framework automatically handles remote data access and computes interpolated values on the machine, which also runs the simulation.

3 Custom AMR Visualization Tools

3.1 ChomboVis

ChomboVis [Ligocki et al., 2003] is an AMR visualization tool built to visualize AMR data produced by Chombo [Chombo, 2007]—an AMR library distributed by the Applied Numerical Algorithms Group (ANAG). Data is stored in files using HDF5. ChomboVis reads these files and uses VTK along with custom Python and C++ code for visualization. A variety of methods are provided for visualizing the data sets including color mapping, grid display, slicing, isosurfaces/contours and streamlines. Users can browse the data directly via spreadsheets by selecting grids graphically or via indices. ChomboVis has both a graphical user interface and a Python application programming interface (API) to facilitate interactive use and scripting.

ChomboVis has rudimentary support for vector field visualization. A GUI interface allows a user to specify which scalar data fields should be combined to form a d-dimensional vector in $\mathbb{R}^d$, as well as specify a *rake* (a line segment in $\mathbb{R}^d$ with seed points space equidistantly). These seed points are integrated either forward or backward (using negative vectors) using backward Euler integration with a user-specified step length for a user-specified number of steps. After each step a check is made to determine if the current position is still inside the current AMR box. If it is, integration is continues, otherwise, the AMR hierarchy is searched, from finest to coarsest, to determine a box in which integration can be continued. These are *streamlines*, not to be confused with streaklines or pathlines.

3.2 Amrvis

Amrvis is a visualization and data analysis tool for examining data files generated by the Center for Computational Sciences and Engineering (CCSE) using their AMR codes. A user can view color planar images of the data, grids, numerical values in a chosen format, subregions, animations, volumetric renderings, contour plots, line graphs and 2D vector fields using arrow glyphs. Amrvis works with 2D and 3D data, requires no specialized graphics hardware, and can run in parallel on both SMP and distributed memory parallel machines. The user can interactively choose the displayed variable (density, pressure, etc.) and AMR level, scale the images, set arbitrary data ranges, and set viewing planes. Amrvis also has batch capabilities for extracting subregions, planes, lines, and points from data files and for preprocessing volume renderings. A separate tool, Amrmovie, can be used to view, animate, and output volume renderings. This tool can animate volume data, which has been preprocessed with Amrvis, at arbitrary orientations and through time. Another separate tool, Amrderive, can be used to access and manipulate AMR data. Examples include deriving a new variable such as vorticity or log of density, calculating integrals, and finding average values across multiple files.

4 Visualization of AMR Data with VisIt

VisIt [Childs & Miller, 2006] is a richly featured visualization and analysis tool for large data sets. It employs a client-server model where the server is parallelized. The nature of parallelization is data parallel; the input data set is partitioned among VisIt's processors. In the case of AMR data, each patch is treated as an atomic unit and assigned to one of the processors on VisIt's parallelized server. For example, patches "level zero, patch zero" and "level one, patch two" may be assigned to the server's processor zero, while patches "level zero, patch one," "level one, patch zero," and "level one, patch one" may be assigned to processor one.

Visualization and analysis of massive scale data sets is an important use case for VisIt. As such, it employs many optimizations to enable the processing of this data. For example, VisIt is able to use spatial extents meta-data to reduce the amount of data that is processed. When a slice of a three-dimensional data set is being rendered, VisIt is able to to limit the patches processed to those that actually intersect the slice. Although this functionality may sound straight forward, it is difficult to implement in a richly featured, module based framework. More information about the contract methodology that enables these optimizations can be found in [Childs et al., 2005].

VisIt's handling of AMR data is made possible by marking coarse cells that are refined at a lower level as "ghost." This marking is done by adding an array to each patch that designates the status of each cell (ghost or non-ghost). Most algorithms can ignore the ghost markings; they operate identically on ghost and non-ghost cells. One advantage of using the ghost cells is that it allows structured grids to retain their native form. That is, removing the ghost cells before applying visualization algorithms would create a grid that was no longer structured. The resulting grid would often be unstructured and that unstructured grid could have a memory footprint that is an order of magnitude larger. Another advantage of using ghost cells is that they allow for proper interpolations to take place, which would not be possible if refined cells were removed before applying visualization algorithms. After all algorithms have been applied, a module walks the data set and removes all cells or geometry resulting from a ghost cell.

VisIt employs the standard Marching Cubes algorithm to contour data. Most AMR data is cell-centered, requiring interpolation to the nodes. For hanging nodes at the boundary of patches at different refinement levels, this interpolation is done incorrectly in VisIt and cracked isosurfaces can result. However, VisIt does *not* produce cracked isosurfaces when abutting patches are at the same refinement level. In this case, VisIt can create a layer of ghost cells around each patch that contains the values of neighboring cells from the other patches. These ghost cells allow for correct interpolation to take place, meaning that a consistent contouring takes place from patch to patch and no cracks are created in this case.

VisIt's employs a data parallel volume rendering scheme that is able to resolve the types of complex sorting issues that arise in unstructured meshes [Childs et al., 2006]. AMR meshes present a special type of load balancing challenge for VisIt's volume rendering algorithm, however. The running time of the algorithm is dependent on the amount of data and the amount of samples. For AMR meshes, the patches at the coarser refinement levels occupy a larger spatial footprint, and, as such, often cover a much larger portion of the picture and contribute more samples. Hence, sampling the patches at coarser refinement levels typically takes much longer than the sampling for patches at finer refinement levels. VisIt attempts to counteract this problem by minimizing the amount of patches at the coarse refinement levels on any given processor. In terms of additional AMR handling, the volume rendering algorithm ignores all samples from cells that are marked ghost. So if a sample point is contained by many patches at different refinement levels, only the value at the finest level will be accepted, since all other levels will have the corresponding cell marked as ghost.

A trend of increasing importance is where visualization and analysis capabilities are coupled in one production quality application. Here, analysis means computing statistical moments of subsets of AMR hierarchies, distribution functions, computed/derived quantities, temporal analysis, etc. VisIt provides a rich set of analysis capabilities (such as integrating density over volume to obtain mass, calculating volumes, surface areas, and moments of inertia), all of which execute in parallel.

5 Future Work

We are currently working on extending VisIt's visualization capabilities for AMR data. To this end, we had extensive meetings with members of the LBNL Applied Numerical Algorithms Group (ANAG) and the LBNL Center for Computational Sciences and Engineering (CCSE). Some feature requests, such as requests for support of picking abilities and spreadsheet support (duplicating functionality present in Amrvis and ChomboVis) indicate that visualization is still used frequently for debugging. On the other hand there is a growing need for for data analysis and visualization capabilities to interpret the results of production AMR simulations. For example, we are currently working on adding line integral convolution-based vector field visualization capabilities to VisIt. Some extensions, such as the spreadsheet interface, are already part of the recent VisIt 1.6 release, even though this functionality is still under development.

6 Acknowledgments

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET). We thank the members of the LBNL Visualization Group, the LBNL ANAG, the LBNL CCSE, and the VisIt development team.

References

[Berger & Colella, 1989] Berger, M. & Colella, P. (1989). Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82, 64–84.

[Bryan, 1999] Bryan, G. L. (1999). Fluids in the universe: Adaptive mesh refinement in cosmology. *Computing in Science and Engineering*, 1(2).

[Chen et al., 2003] Chen, J., Silver, D., & Jiang, L. (2003). The feature tree: Visualizing feature tracking in distributed amr datasets. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003* (pp. 103–110).: IEEE Computer Society.

[Childs et al., 2005] Childs, H., Brugger, E. S., Bonnell, K. S., Meredith, J. S., Miller, M., Whitlock, B. J., & Max, N. (2005). A contract-based system for large data visualization. In *IEEE Visualization 2005* (pp. 190–198).

[Childs et al., 2006] Childs, H., Duchaineau, M. A., & Ma, K.-L. (2006). A scalable, hybrid scheme for volume rendering massive data sets. In *Eurographics Symposium on Parallel Graphics and Visualization* (pp. 153–162).

[Childs & Miller, 2006] Childs, H. & Miller, M. (2006). Beyond meat grinders: An analysis framework addressing the scale and complexity of large data sets. In *SpringSim High Performance Computing Symposium (HPC 2006)* (pp. 181–186).

[Chombo, 2007] Chombo (2000–2007). `http://seesar.lbl.gov/ANAG/chombo/`.

[Fang et al., 2004] Fang, D. C., Weber, H., G., Childs, H., Brugger, E., Hamann, B., & Joy, K. (2004). Extracting geometrically continuous isosurfaces from adaptive mesh refinement data. In *Proceedings of 2004 Hawaii International Conference on Computer Sciences (DVD-ROM conference proceedings)* (pp. 216–224). ISSN 1545-6722.

[Kaehler et al., 2005] Kaehler, R., Prohaska, S., Hutanu, A., & Hege, H.-C. (2005). Visualization of time-dependent remote adaptive mesh refinement data. In *IEEE Visualization 2005* (pp. 175–182).: IEEE Computer Society.

[Kähler et al., 2002] Kähler, R., Cox, D., Patterson, R., Levy, S., Hege, H.-C., & Abel, T. (2002). Rendering the first star in the universe – a case study. In *IEEE Visualization 2002* (pp. 537–540).: IEEE Computer Society.

[Kähler & Hege, 2002] Kähler, R. & Hege, H.-C. (2002). Texture-based volume rendering of adaptive mesh refinement data. *The Visual Computer*, 18(8), 481–492. Zuse Institut Technical Report ZR-01-30.

[Kähler et al., 2006] Kähler, R., Wise, J., Abel, T., & Hege, H.-C. (2006). GPU-assisted raycasting for cosmological adaptive mesh refinement simulations. In *Proceedings of Volume Graphics*: Eurographics Association.

[Kreylos et al., 2002] Kreylos, O., Weber, G. H., Bethel, E. W., Shalf, J. M., Hamann, B., & Joy, K. I. (2002). *Remote Interactive Direct Volume Rendering of AMR Data*. Technical Report LBNL 49954, Lawrence Berkeley National Laboratory.

[Laur & Hanrahan, 1991] Laur, D. & Hanrahan, P. (1991). Hierachical splatting: A progressive refinement algorithm for volume rendering. *Computer Grahpics (Proceedings of ACM SIGGRAPH 91)*, 25(4), 285–288.

[Ligocki et al., 2003] Ligocki, T. J., Straalen, B. V., Shalf, J. M., Weber, G. H., & Hamann, B. (2003). A framework for visualizing hierarchical computations. In *Hierarchical and Geometrical Methods in Scientific Visualization* (pp. 197–204). Springer Verlag.

[Lorensen & Cline, 1987] Lorensen, W. E. & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4), 163–169.

[Ma, 1999] Ma, K.-L. (1999). Parallel rendering of 3D AMR data on the SGI/Cray T3E. In *Proceedings of Frontiers '99 the Seventh Symposium on the Frontiers of Massively Parallel Computation* (pp. 138–145).: IEEE Computer Society.

[Max, 1993] Max, N. L. (1993). Sorting for polyhedron compositing. In *Focus on Scientific Visualization* (pp. 259–268). Springer-Verlag.

[Norman et al., 1999] Norman, M. L., Shalf, J. M., Levy, S., & Daues, G. (1999). Diving deep: Data management and visualization strategies for adaptive mesh refinement simulations. *Computing in Science and Engineering*, 1(4), 36–47.

[Park et al., 2002] Park, S., Bajaj, C., & Siddavanahalli, V. (2002). Case study: Interactive rendering of adaptive mesh refinement data. In *IEEE Visualization 2002* (pp. 521–524).: IEEE Computer Society.

[Shekhar et al., 1996] Shekhar, R., Fayyad, E., Yagel, R., & Cornhill, J. F. (1996). Octree-based decimation of marching cubes surface. In *IEEE Visualization '96* (pp. 335–342, 499).: IEEE Computer Society.

[Squillacote, 2006] Squillacote, A. H. (2006). *The ParaView Guide*. Kitware.

[Stalling et al., 2005] Stalling, D., Westerhoff, M., & Hege, H.-C. (2005). Amira: A highly interactive system for visual data analysis. In *The Visualization Handbook* (pp. 749–767). Elsevier.

[Weber et al., 2001a] Weber, G. H., Hagen, H., Hamann, B., Joy, K. I., Ligocki, T. J., Ma, K.-L., & Shalf, J. M. (2001a). Visualization of adaptive mesh refinement data. In *Proceedings of the SPIE (Visual Data Exploration and Analysis VIII)*, volume 4302 (pp. 121–132).

[Weber et al., 2001b] Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J. M., Hagen, H., Hamann, B., & Joy, K. I. (2001b). Extraction of crack-free isosurfaces from adaptive mesh refinement data. In *Proceedings of the Joint EUROGRAPHICS and IEEE TCVG Symposium on Visualization, Ascona, Switzerland, May 28–31, 2001* (pp. 25–34, 335).: Springer Verlag.

[Weber et al., 2003a] Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J. M., Hagen, H., Hamann, B., & Joy, K. I. (2003a). Extraction of crack-free isosurfaces from adaptive mesh refinement data. In *Hierarchical and Geometrical Methods in Scientific Visualization* (pp. 19–40). Springer Verlag.

[Weber et al., 2003b] Weber, G. H., Öhler, M., Kreylos, O., Shalf, J. M., Bethel, E. W., Hamann, B., & Scheuermann, G. (2003b). Parallel cell projection rendering of adaptive mesh refinement data. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003* (pp. 51–60).: IEEE Computer Society.

The Concepts of VISH

Werner Benger[1] and Georg Ritter[2] and
René Heinzl[3]

[1]Center for Computation and Technology, Louisiana State University, USA

[2]Institute for Astro- and Particle Physics, University of Innsbruck

[3]Institute for Microelectronics, TU Wien, Vienna, Austria

werner@cct.lsu.edu, georg.ritter@uibk.ac.at,
heinzl@iue.tuwien.ac.at

Abstract

VISH is a novel application interface aiming at the separation of algorithm implementation from the software environment they run in. It provides different layers of abstraction to shield algorithms from application-specific details. On the coarsest level, these are generic objects with parameters, on the finest level, this is a concrete model for scientific data covering a wide range of data types. Special attention is given to algorithms for scientific visualization. The objective is to have algorithms only implemented once and share them, together with the data, among applications, even in binary form. This paper presents the concepts and current implementation status in C++.

1 Introduction

1.1 Motivation

Although Scientific Visualization is an established field, the gap between algorithm development and accessibility of new visualization techniques to a wider audience, especially application scientists as end-users, is still a frequent hurdle. New techniques are either implemented as minimal standalone versions using libraries such as GLUT or as a plugin to a larger, sometimes proprietary, software environment. Both methods serve well for the development process, but complicate the wider deployment of the algorithms because they are bound to their runtime environment. Therefore they do not necessarily integrate well into the daily work flow of an application scientists due to limitations of that specific environment for his own purpose.

We envision a software environment that allows to develop and implement visualization algorithms with

- independence from a specific application; ideally, such that even precompiled algorithms can be shared among applications and users;
- minimal overhead on dependencies and external components;
- easy deployment to end users, i.e. support for application-specific file formats and user-friendly GUI without impacting the visualization algorithm itself;
- accessibility to all levels of interfacing hardware;
- high reusability of algorithms, in particular also including I/O layers (support for diverse file formats, remote data access etc.).

This may be achieved via some visualization microkernel that may serve as an "operating system for visualization algorithms". A minimal abstract API shall serve as a framework for development and allow sharing of plugins throughout applications, which provide implementations of the same interfaces.

This is the vision of VISH as a "visualization shell" ("shell" in the sense of a "structural work" or "skin"). It is not an another application by itself, but an implementation of the infrastructure necessary for scientific visualization and therefore a collection of interfaces in the form of libraries. The implementation of such interfaces (such as an input method for an integer) is left to a specific application environment. Algorithms will just see the VISH API (such as to formulate the request for an integer value) and remain application independent.

VISH encompasses different levels of abstraction: the VISH kernel (section 2) provides only interface functionality for general purpose objects with parameters. A data model for scientific visualization is provided by the FiberLib2 (section 3) and complements the VISH kernel. While the VISH kernel basically abstracts event handling and data flow, the FiberLib2 functionality also allows to handle and share data sets among applications. Both components can be used independently, but are designed to integrate well with another, thus forming the FiberLib2-VISH (or shortly "FISH") environment (see Fig. 1 for a depiction of the relationships of the VISH components).

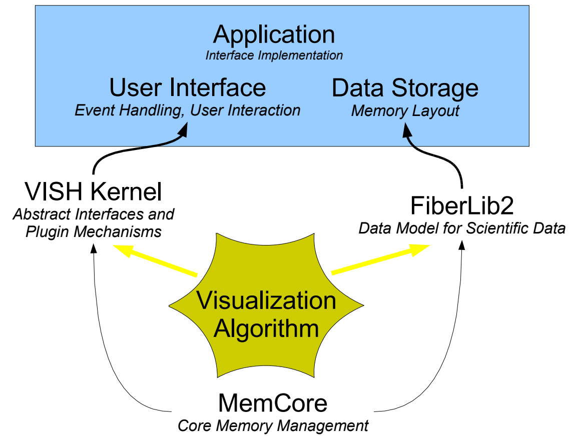


Figure 1: Based on the memory management facilities of MemCore, the VISH and FiberLib2 libraries provide abstraction interfaces for object operations and formulating scientific data, respectively. A newly implemented visualization algorithm would only communicate with these layers, whereas the concrete application behind builds on them or implements the interfaces.

1.2 Previous Work

The IBM Data Explorer [Treinish, 1997] has a long history as a tool for scientific visualization. Its design concepts (in particular its data model based on fiber bundles) were so well founded, that it is still actively used even though its implementation concepts are outdated and active development has ceased. Both Amira [Stalling et al., 2005] and Ensight [CEI, 2007] are widely used visualization environments, but are proprietary, whereas the open source model offers many benefits to both academia and industry, and to both researchers and end users [Johnson et al., 2006]. The visualization toolkit [Kitware, 2005] is a well known open source collection of algorithms, but is far from being a microkernel. SciRun [SCI, 2007] is a recently released open source solution, but a complex application by itself and quite intrusive for visualization algorithms. GLUT[1] is a convenience library based on OpenGL frequently used to demonstrate proof-of-concept, but is not aiming at providing end user friendly applications.

2 The VISH Kernel

The VISH kernel is the API that plugins and application code see. All interactions from application code and with algorithms are through this API. This kernel itself consist of a collection of libraries which are contained in a common folder called "`ocean`" (the ocean is what is required to let fish swim). The main library is the "`plankton`" library (plankton are the smallest animals in the ocean and are essential to nourish fishes). Dependencies of the `plankton` library to external libraries are kept to a minimum, the only requirement is the "`memcore`" library that provides reference pointer and similar functionality in a generic way. Another component of the VISH kernel is the GLvish library, which adds objects with OpenGL rendering capabilities to the VISH kernel. The `vscript` library implements one possible scripting interface to the VISH objects; other scripting languages such as tcl or python are possible as well. However, these libraries will not be discussed here.

2.1 MemCore Functionality

The MemCore library implements means for automatic dynamic memory management. Similar to e.g. the Boost smart pointers [Colvin, 1994], it supports the concept of weak and strong pointers (implementation details are given on p.81 – p.83 in [Benger, 2004]). In addition, these pointers allow implicit up- and downcasting within the same class hierarchy, thereby enabling very compact code when querying objects at runtime such as in:

```
struct A {};
struct B : A {};

void f(const RefPtr<A>&a)
{
      if (RefPtr<B> b = a ) { /* it's a B object*/ }
}
```

[1]http://www.opengl.org/documentation/specs/glut/spec3/spec3.html

In the MemCore library, any strong pointer is automatically a weak pointer as well. As a consequence, if the referenced object is destroyed explicitly, then all strong pointers become invalid. Thus, reference pointers may as well point to automatic, static or dynamically allocated objects that are deleted explicitly.

Another feature of the MemCore library are *typemaps*, that allow to associate objects with C++ type information. The resulting objects can then be indexed using `typeid()`:

```
std::map<type_info, string>  TypeName;
TypeName[ typeid(int) ] = "Integer";
```

In practice, the `type_info` type is not directly suitable as a key to STL maps and some intermediate class is employed. Note that only some ordering relationship among type information instances is necessary. This is provided natively by any C++ implementation and guarantees a bijective association in a platform-independent way.

Typemaps are useful to enable an dynamic interface mechanism to objects, similar to the interface concept in Java as an alternative to multiple inheritance. The MemCore library allows to add and remove interfaces to a base class "Intercube" (inspired by the notion that a cube has many faces) at runtime. It is basically a type map that associates a type domain to some interface object. Dynamic interfaces are very useful to allow plugins to add properties to existing objects that are managed in a central library.

Note that in the following section we will use native pointers in the code examples for illustration, though in the actual implementation only strong and weak pointers are employed.

2.2 Object Management in VISH

The VISH `plankton` library provides a common pool of "managed objects" (class `VManagedObject`). They are contained in a global associative container that allows to address each object via some type ID (the managed object's *domain*) and a user-defined unique text. Schematically, the pool of managed object is of a signature such as (using the STL standard map):

```
map<type_info, map<string, VManagedObject*> >  ObjectPool;
```

An arbitrary object may now be retrieved via some type information, and an arbitrary name:

```
VManagedObject*MyObject = ObjectPool[ typeid(Domain) ][ "Name"];
```

An example for managed objects are Creators. Their constructor inserts them automatically into the ObjectPool using a “Creator” type domain. Within this domain they may be accessed via a a unique textual identifier. They may well be implemented as static objects within a dynamic library, such that this Creator object is visible in the ObjectPool once the library is loaded. Upon unloading the dynamic library, when the Creator’s destructor is called, it will automatically become invisible in the ObjectMap due to the use of MemCore’s reference pointer scheme. The base class of these Creator object comes with a virtual function that allows to create a certain category of objects, which are introduced in the next section.

2.3 Objects, Parameters and Inputs

The basic instance in VISH are abstract objects that may perform some operation based on some input parameters and may serve as input themselves; these are called `VObjects`. VISH provides means to expose these input pa-

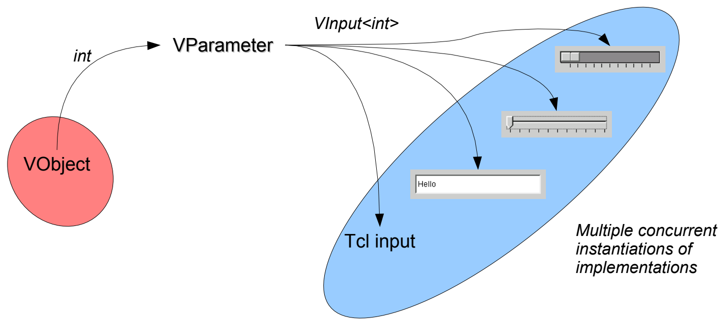


Figure 2: `VObjects` request `VParameters` via intrinsic C++ types, which are implemented via one or many `VInputs`.

rameters, equip them with appropriate input objects, and to connect or share them with other `VObjects`. The relationship among objects, their parameters, and inputs of them defines the core functionality of VISH (see Fig. 2 for illustration):

- A `VObject` is an abstract class that implements some functionality controlled by input parameters. Its results are provided as output parameters.
- A `VParameter` provides means to retrieve numerical values. Multiple `VObjects` may refer to the same `VParameter` such that they can be easily coupled (for instance, different visualization algorithms operating on the same slice of the same volume, but different data fields).
- A `VInput` implements an actual numerical value and the means to modify it. A typical example is a slider within a graphical user interface. Alternative representations of the same value are desirable, such that a problem-specific representation can be chosen (e.g. a dial widget instead of a slider). Since such a representation may well depend on the context, also multiple representations of the same value are allowed (all referring to the same `VParameter`).

Considering these requirements, we may illustrate the central VISH classes schematically via native C pointers (note that the integer data type is only used for exemplification, the actual code uses weak and strong pointers on on abstract type-independent base classes):

```
struct VObject     // some VISH object doing something
{
     int **red;    // a parameter to the VObject

     void update() // the object's operation
     {
         printf("My red value is %d\n", **red);
     }
};
```

I.e., the numerical value of the object is not contained in the object itself, but only retrieved indirectly. The `VObject` needs to dereference the pointer twice to get the actual value. This indirection allows multiple objects to share the same value:

```
VObject A, B;
   A.red = B.red;
```

The *double* indirection allows to change the implementation of a certain value for all instances simultaneously:

```
int a, b;
int *value;
VObject A, B;

   B.red = &value;
   A.red = B.red;
   value = &a; // drive A and B objects via value a
   value = &b; // drive A and B objects via value b
```

VISH generalizes this basic idea through abstract objects. They are template instantiations over arbitrary C++ types and are derived from abstract base classes. Conceptually, the `ints` in the above example correspond to the `VInput` objects, and the first indirection, the `int*` to the `VParameters`:

- Both input and output parameters of `VObjects` are bound to an intrinsic C++ type.
- A specific `VObject` may
  - request `VParameters`, each such request is specified via a C++ type info plus an associated text. Such *input parameters* may be shared among `VObjects`.
  - provide `VParameters`, which are then bound to this certain object. It is the duty of the respectively owning `VObject` to update those output`VParameters` with the correct numerical values. Such *output parameters* may well be used as input parameters of other `VObjects`. This relationship among `VObjects` thus forms a graph among `VObjects`, called the *data flow* graph since it corresponds to the flow of data among `VObjects`.
- One `VParameter` may be represented by one or many `VInputs`. If one `VInput` is modified, then its corresponding `VParameter` will be notified about this change and forward this notification to all other associated `VInputs`. Thus different representations have the means to update themselves to display the changed value. The relationships

among `VParameter` and `VInput` forms a graph, which is called the *control flow* graph since its values controls (parameterizes) the behavior of the `VObjects`.

The control flow graph is executed synchronously to the user interaction. In contrast, the data flow graph is only traversed when data are requested by some *data sink* (a `VObject` that resides at the end of the data flow graph, i.e. it has only inputs, but no outputs), i.e. asynchronously to user interaction. For instance, within a graphical display, such a data sink may be implemented by some OpenGL viewer that requests new data to be updated at 30 frames per second, or it may be some `VObject` which saves data to a file.

In the data flow model, VISH implements the push model (the data sink determines traversal of the data flow graph), like VTK [Kitware, 2005], not the pull model (the data source drives traversal through the data flow graph), like Amira [Stalling et al., 2005]. Only those data that are supposed to be displayed (in a visualization context) are to be loaded on-demand from the disk, for instance one time slice out of an evolution of a dynamic data set.

3 The FiberBundle Data Model

The fiber bundle data model is a generic scheme to cover a wide range of scientific data types through a specific data structure that is inspired by the mathematics of fiber bundles. It draws upon concepts of differential geometry and topology. The original ideas have been laid out by [Butler & Pendley, 1989] and later refined by [Haber et al., 1991]. While many modern visualization environments do not implement a data model at all and rather implement the various data types on an ad-hoc basis (with more or less random overlap of properties), the IBM Data Explorer (now OpenDX) [Treinish, 1997] has successfully proved the benefits of the fiber bundle data model. The model developed in [Benger, 2004] has extended the concepts found there in order to further systematize and reuse the concept of a fiber bundle.

The `FiberLib2` is a new implementation of the fiber bundle data model from [Benger, 2004]. Fundamental to it are the six hierarchy levels `Bundle, Slice, Grid, Topology, Representation, Field`. Given one hierarchy level, the next one is accessed via some identifier that is specifically appropriate for this level:

hierarchy object	identifier type	identifier semantic
`Bundle`	floating point number	time value
`Slice`	string	grid name
`Grid`	integer set	topological properties
`Topology`	pointer	relationship map
`Representation`	string	field name
`Field`	multidimensional index	array index

Only two of these identifier types are strings, and of arbitrary value. The semantics of the grid and the field names are left to the application code and the user. All other identifiers do have specific meanings in the fiber bundle data model and are used to determine the specific properties of a data set. In order to get from one hierarchy level to the next one, the "`[]`" and "`()`" operators are used for modified indexing (the return value is guaranteed to exist) and unmodified indexing (the return value will be zero if no subhierarchy entry exists for the given index):

```
Bundle MyBundle;
Slice&S = MyBundle[ 1.0 ];
RefPtr<Slice> MySlice = MyBundle( 1.0 );
```

In this example, the operator "`[]`" will create an entry for the time $t = 1.0$ in the `MyBundle` object, if it does not exist yet. It provides a reference to the Slice object that contains all data for $t = 1.0$. The operator "`()`" will query the `MyBundle` object whether data exist for $t = 1.0$, and if not, return an invalid pointer.

3.1 Topological Properties

The `Topology` level of the fiber bundle hierarchy describes a certain topological property. This can be the vertices, the cells, the edges etc. . It is loosely connected to the skeletons of a cw-complex, but in this context also used to specify different mesh refinement levels and agglomerations of certain elements. Details are given in [Benger, 2004], whereas here the only property of relevance is that all data fields that are stored within a `Topology` level have the same number of elements. I.e., they share their index space (a data space in HDF5 terminology). Moreover, each `Topology` object within a `Grid` object is uniquely identified via a set of integers, which are the *dimension* (e.g., the dimension of a k-cell), *index depth* (how many dereferences are required to access coordinate information in the underlying manifold) and *refinement level* (a multidimensional index, in general).

3.2 Relationship Maps

Numerical values within a `Topology` level are grouped into `Representation` objects, which hold all information that is *relative* to a certain “representer”. Such a representer may be a coordinate object that refers to some cartesian (or polar) chart, or it may well be another `Topology` object, either within the same `Grid` object or even within another one. Given a `Topology` object called `Vertices` and a chart object `CartesianChart3D`, we may retrieve the representation of the Vertices in cartesian chart using the operator syntax:

```
Topology Vertices;
Chart    CartesianChart3D;

Representation& CartesianVertices = Vertices[ CartesianChart3D ];
```

Given a `Topology` object describing the triangles of a Grid, we may retrieve triangle information in cartesian coordinates in a similar way (e.g. surface normal vectors of the triangles), but as well retrieve the information on how the triangles relate to the vertices:

```
Topology Vertices, Triangles;
Representation & TrianglesAsVertices = Triangles[ Vertices ];
```

The inversion, the representation of the vertices via triangles, e.g. for determining which triangles share the same vertex, is easily accessed by the inverse operation:

```
Representation & VerticesAsTriangles = Vertices[ Triangles ];
```

Each `Representation` is a collection of `Field`s, which are basically multidimensional arrays that are accessed via some textual identifier. The value of this textual identifier is left to the application, with the mere exception of the “`Positions`” entry. This specific field describes the locality of the elements of one index space within the domain of the representer (e.g., within a chart, or as set of indices).

4 Example Application

The following example demonstrates how to set up a VISH object that defines a time-dependent uniform vector field. It involves infrastructure components from VISH , Memcore, and FiberLib2.

```
class   Vectorfield : public VObject // Implement VISH object
{
   Bundle B;
   RefPtr<VParameter> TimeParameter;  // refer to some parameter
public:

  Vectorfield()
  {
      // request floating point parameter
      TimeParameter = addParameter("time", 0.0);
  }

  void update()   // virtual VISH function
  {
  double time = TimeParameter->getValue() // request parameter value

       Slice &S = B[ time ];        // operating on FiberLib2 from here
       Grid  &G = S[ "unigrid" ];
       RefPtr<Skeleton> Vertices = G.makeVertices(3);
       RefPtr<Chart> myCartesian = G.makeChart( typeid(Fiber::CartesianChart3D) );
       Representation&R = (*Vertices)[ myCartesian ];
       RefPtr<Field> Coords  = R[ "Positions" ];
       MultiIndex<3> Dims = MIndex(31,43,53); // define size of the grid

       typedef MemArray<3, tvector3> VectorArray_t;
       RefPtr<Field> Vectors = R[ "vectors" ];
       RefPtr<VectorArray_t> VectorData = new VectorArray_t(Dims);
          Vectors->setData( VectorData, MemCache() );

      ProcArray_t*PCrds = new ProcArray_t();   // define uniform coordinates
         Coords->setData( PCrds, MemCache() ); // (details not shown)

      MultiArray<3,tvector3>     Vec = *VectorData;
  /* Vec is a multidimensional array of vectors and
     can be modified now, e.g. at index 4,4,4 with some
     time-dependent value (in practice, all elements need to be set) */
     Vec[ MultiIndex(4,4,4) ] = tvector3(1.0, 2.0, time);
  }
};
```

5 Results

VISH is still under an experimental development state, but the evolution of the core infrastructure has widely settled down already. Using this infrastructure, a simple visualization algorithm like rendering vector arrows of slice from a time-dependent uniform vector field can be realized in less then 300 lines of source code, including low-level OpenGL calls and parameter steering. A reference implementation employing QT of a user interface as a plugin to the VISH kernel has been developed. Alternatively there exists a preliminary interface for the Amira visualization software, such that plugins can be shared in binary form among these two environments.

6 Conclusion

Practical experience shows that newly developed visualization techniques are not easily and quickly deployed by end-users. There exist complex – and frequently proprietary – applications that are problematic to adapt to a custom problem on one side, and there exist separate stand-alone versions of highly specialized algorithms that cannot be used in general context on the other side. The concepts of VISH intend to close this gap.

As part of the ongoing evolution, the VISH kernel has proved to suit well the needs of an abstraction layer, since the requirements for further kernel modifications have decreased as more functionality has been added to the kernel's periphery. Complemented with the fiber bundle component VISH is able to cover a wide range of scientific data types as well, even in its early phase. Future efforts therefore will now focus on plugin components rather than on the kernel itself.

It will be part of future investigation also to test whether the VISH abstraction is sufficient to encapsulate even contradictory concepts such as the push and pull model within the same application. The envisioned "worst-case" scenario here is to have the entire VISH kernel appear as a single object within another application without exposing its internal structure (i.e., the collection of VISH objects and their relation to parameters).

References

[Benger, 2004] Benger, W. (2004). *Visualization of General Relativistic Tensor Fields via a Fiber Bundle Data Model*. PhD thesis, FU Berlin.

[Butler & Pendley, 1989] Butler, D. M. & Pendley, M. H. (1989). A visualization model based on the mathematics of fiber bundles. *Computers in Physics*, 3(5), 45–51.

[CEI, 2007] CEI, C. E. I. (last visited 4/30/2007). EnSight - FEA and CFD post-processing visualization. Available from: `http://www.ensight.com/ensight.html`.

[Colvin, 1994] Colvin, G. (1994). Exception safe smart pointers. Available from: `http://std.dkuug.dk/jtc1/sc22/wg21/docs/papers/1994/N0555.pdf`.

[Haber et al., 1991] Haber, R. B., Lucas, B., & Collins, N. (1991). A data model for scientific visualization with provisions for regular and irregular grids. In *VIS '91: Proceedings of the 2nd conference on Visualization '91* (pp. 298–305). Los Alamitos, CA, USA: IEEE Computer Society Press.

[Johnson et al., 2006] Johnson, C. R., Moorehead, R., Munzner, T., Pfister, H., Rheingans, P., & Yoo, T. S., Eds. (2006). *NIH-NSF Visualization Research Challenges Report*. Los Alamitos, CA, USA: IEEE Press, 1st edition. Available from: `http://tab.computer.org/vgtc/vrc/NIH-NSF-VRC-Report-Final.pdf`.

[Kitware, 2005] Kitware (2005). Visualization toolkit. Last visited 4/25/2007. Available from: `http://www.kitware.org/`.

[SCI, 2007] SCI (last visited 4/30/2007). SCIRun: A Scientific Computing Problem Solving Environment, Scientific Computing and Imaging Institute (SCI). Available from: `http://software.sci.utah.edu/scirun.html`.

[Stalling et al., 2005] Stalling, D., Westerhoff, M., & Hege, H.-C. (2005). Amira - an object oriented system for visual data analysis. In C. R. Johnson & C. D. Hansen (Eds.), *Visualization Handbook*: Academic Press. Available from: `http://www.amiravis.com/`.

[Treinish, 1997] Treinish, L. A. (1997). Data explorer data model. `http://www.research.ibm.com/people/l/lloydt/dm/dx/dx_dm.htm`.

HD Collaborative Framework for Distributed Distance Learning

Luděk Matyska[1,2,3] and Petr Holub[1,2] and Eva Hladká[1,3]

[1]CESNET z. s. p. o., Prague, Czech Rep.

[2]Institute of Computer Science, Masaryk University, Brno, Czech Rep.

[3]Institute of Computer Science, Masaryk University, Brno, Czech Rep.

ludek@ics.muni.cz, hopet@ics.muni.cz, eva@fi.muni.cz

Abstract

A high fidelity collaborative environment based on a multi-point uncompressed audio and HD video capturing, transmission, and presentation systems has been deployed to support *Introduction to High Performance Computing* class given by prof. Sterling at LSU. The lecture has been transmitted live to several other universities in the USA and in the Czech Republic. The data distribution used a dedicated high throughput (10GE) networks partly managed by the HARC software. The system worked very well most of the time, with audio being the most problematic part to setup correctly.

1 Introduction

Teaching and learning are activities that could benefit from the availability of high fidelity collaborative platforms, as they are in essence a collaborative undertaking. This way, larger numbers of geographically dispersed students could get access to top scientist, without extreme demands on the scientist—to travel to each place and have a lecture separately—and on university budgets—to be able to pay the top scientists and also cover their travel expenses.

We have implemented a multi-point uncompressed HD over IP transmission system complemented by other supporting technologies. As a pilot experiment, this high fidelity collaborative environment has been used by prof.

Thomas Sterling from the Louisiana State University, USA, to present his *Introduction to High-Performance Computing* class. The institutions participating in the class were mostly from Louisiana and neighboring US states—University of Arkansas (UARK), Louisiana Technical University (LATECH), and recently also MCNC and North Carolina State University (NCSU)—and one truly remote—Masaryk University (MU) in the Czech Republic.

2 Interactive Collaboration Technologies

The collaborative system used is based on our implementation of a multipoint uncompressed audio and HD video capturing, transmission, and presentation. The audio part has been implemented by RAT [Hardman et al., 1995] at 16 b quantization and up to 48 kHz sampling rate in stereo. Thus data rates up to 1.6 Mbps were used. The audio is distributed in full N:N way, such that all participants can hear and talk to one another.

The uncompressed HD video was sent as HD-SDI over IP transmission based on the UltraGrid software [Holub et al., 2006], featuring full 1080i SMPTE 274M/292M video with effective resolution of 1920×1080, 60 interlaced fields per second, 4:2:2 color space sampling, and 10 b per color plane. The total data rate after packetizing the data using 8,500 B Jumbo Ethernet frames is about 1.5 Gbps including aggregate IP/UDP/RTP packet overhead (44 B per packet in total). As the video needs much higher throughput than the audio, a simpler distribution model is used: (1) 1:N from the lecturer to all participating sites and (2) N:1 from all sites to the lecturer only (the sites hear but do not see each other).

The high demanding setup with data rates of over 1 Gbps has been complemented with a backup solution based on Access Grid and webcasting in QuickTime format for those unable to participate at full rate such as LATECH.

2.1 Network and Data Distribution Setup

Because of multi-Gigabit aggregate media data flows, the network has been implemented using experimental 10 Gigabit Ethernet (10GE) infrastructure with both statically and dynamically allocated λ-services. The network topology resembles star with center in StarLight (SL) in Chicago. LSU and LATECH use the 10GE backbone Louisiana Optical Network Initiative (LONI), which has an uplink to SL via a dedicated Enlightened wave running 10GE on National Lambda Rail (NLR). The Enlightened wave is being dynamically provisioned for the class and for the testing windows using a preliminary version of HARC [Battestilli et al., 2006], the software stack developed by the Enlightened project. UARK is connected via the OneNet network and a wave on NLR running 10GE; again, the circuit is automatically setup before the class begins and is shut down after it ends. MCNC and NCSU are connected to SL in the same way using NLR waves. MU is connected using a dedicated permanent circuit Prague–SL leased by CESNET and the Brno–Prague link is implemented using CzechLight experimental infrastructure using leased dark fiber lit by CESNET equipment. The whole infrastructure is a switched L2 network and is transparent on the IP level. Measured network latencies are shown in Table 1 (data for LATECH are not provided as LATECH does not have a 10GE link at the last mile).

From	*to StarLight*	*to LSU*
LSU	30.6 ms	–
Masaryk University	115.4 ms	145.7 ms
MCNC	23.5 ms	53.8 ms
University of Arkansas	19.3 ms	49.6 ms

Table 1: Network round-trip latencies (RTTs) in the 10 Gigabit infrastructure

The data flows are distributed with our user-empowered software UDP packet reflectors [Hladká et al., 2004, Holub et al., 2005]. The reflectors run on dual-Opteron computers equipped with 10GE NICs. The schematics of the 1:N video distribution is shown in Figure 1. Note that there are actually two sites participating at LSU as not everybody can fit in a single room—thus the same technology is used for distributing the data locally on the campus between two buildings. There are also two streams being sent to MU—one of them is used for live video feed, while the other is used for full-quality recording (see Section 3).

2.2 Site Setup

Each site is equipped with one sender and one or more receivers for uncompressed HD video over IP transport system. The machines are essentially

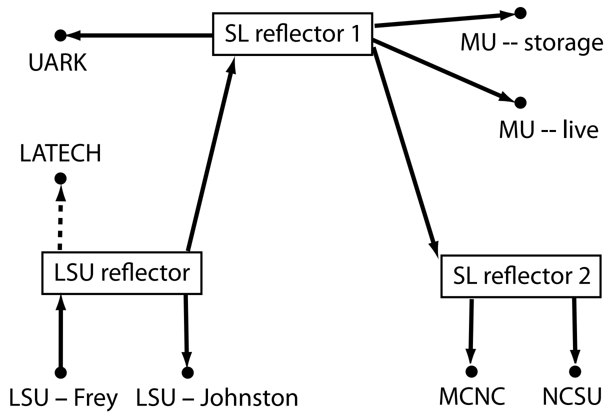


Figure 1: Overlay data distribution network for the 1.5 Gbps uncompressed HD video streams.

the same as used for the reference UltraGrid 1080i [Holub et al., 2006] implementation, i.e., dual-Opteron computers with 10GE NICs. The sender computer is equipped with a DVS Centaurus HD-SDI capture card, while the receiver uses a NVidia graphics card to render the video on the attached LCD screen or projector—we used 24" to 30" LCD screens and plasmas or at MU, the Projection Design Cineo3+ 1080i projectors. The video part runs flawlessly with very good user perception.

Though less challenging from the networking and processing perspective, the audio part creates many more problems namely on the capture side. The worst problems are caused by bad audio installations (ranging from wiring problems to echo canceling and gain control problems and over-processing of the sound by various components before it gets to the computer—all this resulting in noises and distorted sound) used within the lecturing rooms primarily at LSU and wireless microphone interferences. Problems have also been encountered when using some on-board integrated sound cards, as their obviously half duplex behavior resulted in clicks and sound distortions. After appropriate corrective actions were undertaken, the resulting sound quality was very high and pleasing to hear.

3 Data Storage

The lectures were given at 3pm Central American Time, which is 7 hours behind from the Central European Time[1] used at MU. That meant lectures given from 3:30PM onwards at LSU started at 10:30pm - too late for regular lectures. An additional complication arose due to the difference in the beginning of a semester—LSU starts one month before the MU Spring semester. For the actual semester, we decided to overcome both problems by recording live lectures at full quality *at the receiving site* and displaying them later at a more appropriate time.

The data storage—requiring at least 190 MBps read/write throughput—has been implemented using a very fast local disk array comprising 12 disks in RAID 0. The write performance of the array was 385 MBps and read performance was 414 MBps. The media streams (1 video and N audio streams) from the network are captured and stored as raw data on the disk array, with each 1.5 hour lecture requiring about 1 TB of stored data. Subsequently, they are indexed so that seeking capability is available. The data is replayed later from the same disk array.

4 Conclusions

The original collaborative system described in this paper had been implemented in 2005 and redesigned by the beginning of year 2007. It has been used since then for supporting the HPC class. Due to running on highly experimental networking infrastructure, some sites had to resort to using backup solutions at times, but the infrastructure worked flawlessly most of the time. Another experience gathered from this class is the importance of high quality audio installation and the difficulty of its implementation at participating sites. The low sound quality is much less tolerated when other means of communication are of high quality compared to common point to point communication tools.

[1]By coincidence, both time zones use the same acronym—CET.

5 Acknowledgments

This project has been kindly supported by the research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201). We would also like to thank other people working along with us on this project, namely Tomáš Rebok, Miloš Liška, Lukáš Hejtmánek from Laboratory of Advanced Networking Technologies at Masaryk University, and our partners at CCT at LSU and other partnering sites.

References

[Battestilli et al., 2006] Battestilli, L., Hutanu, A., Karmous-Edwards, G., Katz, D. S., MacLaren, J., Mambretti, J., Moore, J. H., Park, S.-J., Perros, H. G., Sundar, K. S., Tanwir, S., Thorpe, S. R., & Xin, Y. (2006). EnLIGHTened computing: An architecture for co-scheduling and co-allocating network, compute, and other grid resources for high-end applications. Available from: `http://enlightenedcomputing.org/`.

[Hardman et al., 1995] Hardman, V., Sasse, A., Handley, M., & Watson, A. (1995). Reliable audio for use over the internet. In *Proceedings of INET'95* Honolulu, Hawaii.

[Hladká et al., 2004] Hladká, E., Holub, P., & Denemark, J. (2004). An active network architecture: Distributed computer or transport medium. In *Proceedings of 3rd International Conference on Networking (ICN'04)* (pp. 338–343). Gosier, Guadeloupe.

[Holub et al., 2005] Holub, P., Hladká, E., & Matyska, L. (2005). Scalability and robustness of virtual multicast for synchronous multimedia distribution. In *Proceedings of 4th International Conference on Networking (ICN'05)*, volume 3421 of *Lecture Notes in Computer Science* (pp. 876–883). La Réunion: Springer-Verlag Heidelberg.

[Holub et al., 2006] Holub, P., Matyska, L., Liška, M., Hejtmánek, L., Denemark, J., Rebok, T., Hutanu, A., Paruchuri, R., Radil, J., & Hladká, E. (2006). High-definition multimedia for multiparty low-latency interactive communication. *Future Generation Computer Systems*, 22(8), 856–861.

Is Visualization only Data Representation?

Wolfgang Kapferer
Astro- und Teilchenhysik, Universität Innsbruck
`wolfgang.e.kapferer@uibk.ac.at`

Abstract

A short presentation on the different needs for investigating state of the art simulations will be given. In addition some needs (or wishes) for a visualization package are listed.

1 Introduction

Besides the well established branches in astrophysics, observation and theory, a new branch arose in the last decades, namely computational astrophysics. With the help of numerical methods to solve highly non-linear physical systems, different models about the origin and evolution of large scale structures in the universe, like clusters of galaxies or the cosmic web, were feasible on supercomputers. The comparison of multiwavelength observations and the results of the simulation revealed a robust set of cosmological model parameters (ΛCDM). The numerical simulations give nowadays several tens of quantities of the simulated system with very high resolution for many timesteps. Therefore the requirements on visualization packages are shifting, from the purely visualization of the different quantities of the computational domain by well known techniques (slices, height-fields, isosurfaces, projection-views or volumetric rendering) to complex techniques to find the 'unexpected' results, which lead to a deeper comprehension of the simulated and/or observed system.

2 Are typical techniques for visualization of numerical data enough?

In fig. 1 some typical example for 3D N-body/hydrodynamic data are shown. Typical visualization packages offer volumetric rendering for scalar fields, stream ribbons or even fancy illuminated stream lines for velocity fields or special geometries for high dimensional data. All of these techniques are able to show the evolution of quantities involved in a simulation, such as density, temperature, velocity or tensor fields. They are necessary to assist by the development of algorithms and the extracting of the final results.

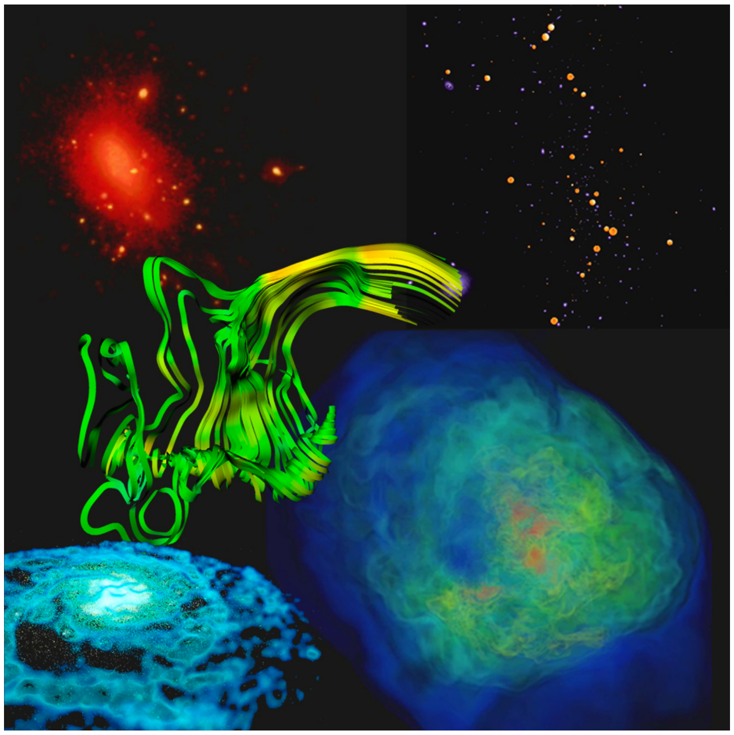


Figure 1: Typical visualization techniques for 3D numerical simulation data. Volumetric rendering, stream ribbons, more complex geometries like ellipsoids are typical techniques to investigate numerical simulations.

Are standard packages like AMIRA, PARAVIEW, MAYAVI, just to mention some, really as useful as commonly believed?
In observations/experiments data reduction is very important. In principle

it is the art to remove the useless data from the useful data. In complex experiments or multiwavelength observations the data reduction becomes more and more complex and crucial. If the amount of complexity is too high, there are no real conclusions drawable. In fig. **??** a small example is given. A 3D quantity of a simulated galaxy cluster, i.e. the iron distribution in a hot plasma between the galaxies, is reduced in an averaged radial profile of the same quantity. Only this representation of the numerical data is direct comparable to X-ray galaxy cluster observations. The postprocessing of the original 3D numerical data is the only way to compare simulation and observation.

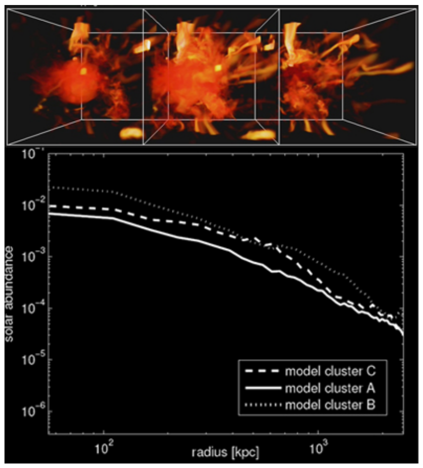


Figure 2: The results from simulations are most often simple relations, or sometimes even only one number. A useful visualization package should assist the user by reducing complex data like in the top of the image to simple relations like 2D radial binned quantities.

Some packages, like AMIRA, provide very basic statistical methods to investigate data, but postprocessing in other numerical tools, like MATLAB or IDL is almost always needed. A useful tool would be a automated relation finder. A helper, that would dig deep into the data and would search for all correlations in an automated way. A wish list for a visualization packages for state of the art simulations could look like this:

- advanced visualization techniques for scalar, vector or higher dimensional fields,

- capable of processing complex topologies and adaptive or fixed refinement techniques
- handling of time series and a powerful numerical postprocessing interface
- processing many different quantities at the same time and testing the numerical data in an automated way for dependencies and relations with advanced fitting routines

To display many (e.g. ten) quantities per voxel seems an open issue for present visualization techniques. As simulations get more advanced, it will get more and more important to address this open question.

3 Conclusions

There is no doubt that advanced visualization techniques are necessary to develop and investigate state of the art simulations. Much efforts are made by visualization software developer to handle large datasets, complex topologies and computational domain refinements. The increasing complexity of numerical simulations shift the focus on pure data representation to advanced data reduction in addition. Researchers need tools which help to represent the complexity of a simulation in a visual way and find new relations and 'unexpected' results in the simulation as well.

4 Acknowledgments

The author wants to the thank the members of the HYDRO-SKI Team (`http://astro.uibk.ac.at/astroneu/hydroskiteam/`)

Direct Surface Extraction from Smoothed Particle Hydrodynamics Simulation Data

Paul Rosenthal[1] and Stephan Rosswog[2] and Lars Linsen[1]

[1]Computational Science and Computer Science,
School of Engineering and Science,
Jacobs University Bremen, Germany

[2]Astrophysics,
School of Engineering and Science,
Jacobs University Bremen, Germany

`{p.rosenthal,s.rosswog,l.linsen}@iu-bremen.de`

Abstract

Smoothed particle hydrodynamics is a completely mesh-free method to simulate fluid flow. Rather than representing the physical variables on a fixed grid, the fluid is represented by freely moving interpolation centers ("particles"). Apart from their position and velocity these particles carry information about the physical quantities of the considered fluid, such as temperature, composition, chemical potentials etc. Being completely Lagrangian and following the motion of the flow, these particles represent an unstructured data set at each point in time, i.e. the particles do not exhibit a spatial arrangement nor a fixed connectivity. To visualize the simulated particle data at a certain point in time, we propose a method that extracts surfaces segmenting the domain of the particles with respect to some scalar field.

For scalar volume data, isosurface extraction is a standard visualization method and has been subject to research for decades. We propose a method that directly extracts surfaces from smoothed particle hydrodynamics simulation data without 3D mesh generation or reconstruction over a structured grid. It is based on spatial domain partitioning using a kd-tree and an indexing scheme for efficient neighbor search.

A geometry extraction step computes points on the surface by linearly interpolating between neighbored pairs of sample points. Its output is a point cloud representation of the surface. The final rendering step uses point-based rendering techniques to visualize the point cloud. A level-set approach can be applied for smoother segmentation results when extracting the geometry of the zero level set.

1 Introduction

Surface extraction is the most commonly used visualization technique for scalar volume data beside direct volume rendering. Common ways to deal with unstructured point-based volume data are to resample the data using scattered data interpolation techniques [Franke & Nielson, 1991], or to compute a grid that connects the unstructured data points [Co & Joy, 2005]. With a recently presented new approach [Rosenthal & Linsen, 2006], it is now possible to extract surfaces directly from unstructured point-based volume data without any resampling or mesh generation.

In smoothed particle hydrodynamics each particle is moving according to the flow equations. Therefore, at each point in time, the simulation data are represented on an unstructured point-based data set. Standard visualization techniques include rendering of the color-coded sample points, e.g. by the commercial software AVS or IDL, extraction of the point set's bounding surface [Vesterlund, 2004], or direct volume rendering, e.g. by a free software package developed by Daniel Price from the University of Exeter, UK. We propose a method that directly extracts a surface that segments the domain of the particles with respect to a given isovalue. For noisy and sparse sampled data sets we introduce a level-set method smoothing the data set to get a better segmentation result.

The isosurface-extraction pipeline is described in Section 3. In Section 4 an introduction to the used level-set method is given. Finally, Section 5 provides a case study applying our methods to smoothed particle hydrodynamics.

2 Smoothed Particle Hydrodynamics

Invented in the astrophysical context [Lucy, 1977, Gingold & Monaghan, 1977], smoothed particle hydrodynamics (SPH) has by now found its way in many different modeling areas, see [Monaghan, 2005] for a recent comprehensive review. SPH is a completely Lagrangian, mesh-free method to solve the equations of fluid dynamics and therefore particularly well suited for simulations in which large deformations occur.

The fluid continuum is represented by moving interpolation centers (”particles”), each of which possesses an interaction range, the so-called smoothing length. The local density is calculated via a kernel weighted sum over neighboring particles within its interaction range, the calculation of pressure gradients involves sums over kernel gradients rather than finite differences. To arrive at the discretized fluid equations nothing more than a Lagrangian, a summation prescription for the density, and the first law of thermodynamics are needed. Therefore, even in their discretized form the equations conserve all physically conserved quantities *by construction.* The particles are advanced in time according to the equation which expresses momentum conservation.

3 Isosurface Extraction

In every point in time, the result of a smoothed particle hydrodynamics simulation is an unstructured point-based volume data set. More precisely, it is a set of trivariate scalar fields $f : \mathbb{R}^3 \to \mathbb{R}$, whose values are given for a large, finite set of sample points $(\mathbf{x}_i, f(\mathbf{x}_i))$, whose positions are unstructured, i.e. they are not arranged in a structured way, nor are any connectivity or neighborhood informations known for the sample point locations.

To visualize such a scalar field, our intention is to extract an isosurface $\Gamma_{\text{iso}} = \{\mathbf{x} \in \mathbb{R}^3 \ : \ f(\mathbf{x}) = v_{\text{iso}}\}$ with respect to a real isovalue v_{iso} out of the range of f. This isosurface extraction is performed in two main steps. First, a set of isopoints $\mathbf{p}_k \in \mathbb{R}^3$ on the isosurface is computed, i.e. $f(\mathbf{p}_k) = v_{\text{iso}}$. In a second step, some kind of neighborhood information for the isopoints is generated. This is subsequently used for rendering the isosurface.

Our idea for the computation of isopoints from the smoothed particle hydrodynamics simulation data is based on linear interpolation between pairs of samples with close positions $\mathbf{x}_i$ and $\mathbf{x}_j$. The inspiration for this approach is given by isopoint computation using the marching tetrahedra algorithm [Treece et al., 1999] after partitioning the domain via Delaunay tetrahedrization. In this case, the Delaunay triangulation connects natural neighbors defined by the Voronoi diagram.

We want to adopt this isopoint computation method for our purposes while avoiding the expensive computation of the Delaunay tetrahedrization. Thus, we need an approximation of the natural neighbors for each sample position $\mathbf{x}_i$, which will be obtained by using a spatial space decomposition. Simply replacing the natural neighbors by the nearest neighbors would fail in our case of smoothed particle hydrodynamics simulation data due to varying density distributions of the sample point locations.

The kd-tree data structure is known to be a data structure with well-balanced trade-off between flexibility and efficiency [Bentley, 1975] and, what is essentially in our application, with robustness against varying density distribution of sample point positions. To perform a fast exploration of the kd-tree, we introduce an indexing scheme that, beside saving storage space, allows us to determine neighbors using bitwise operations on the index. We determine a small number of potential candidates for our neighbors and reject some of them using an angle and maximum distance criterion.

We compute isopoints respectively by linearly interpolating between two neighboring sample points on different sides of the isosurface. In a final step, we use a recently presented point-based rendering technique [Linsen et al., 2007] using splats to render the isosurface in point cloud representation.

3.1 Data Storage

The n SPH-simulation data points are stored in a three-dimensional kd-tree. Besides the Cartesian coordinates of the points also the function values from the SPH-simulation have to be stored. The tree is stored as a vector of points. Thereon the kd-tree is build recursively splitting the sample point sets of each cell at the median, while cycling through the coordinates. The recursion stops if every cell contains exactly one sample point.

The height of the tree is $\lceil \log_2(n+1) \rceil$. In worst-case ($n = 2^j$), $j \in \mathbb{N}_+$, the number of nodes in the kd-tree is $2n - 1$. The nodes of the kd-tree are stored in the vector in breadth-first order, i.e. the root is in position 1 and the children of the node in position j are in positions $2j$ and $2j + 1$.

To have fast access to the nodes of the kd-tree a binary indexing scheme is introduced. It allows fast navigation through the tree by using only binary operations on the binary representation of the nodes positions. Moreover, qualitative propositions about the locations of cells can be made. Thus many informations are implicitly saved in the indexing scheme and will be used for speeding up runtime.

3.2 Neighborhood Computation

The search for all neighbors will be done for all sample points that lie inside a cell of the kd-tree. The neighborhood of a sample point $\mathbf{x}$ contains all cells or splitting planes that have at least one point in common with the cell of $\mathbf{x}$. A two-dimensional illustration of the neighborhood is shown in Figure 1.

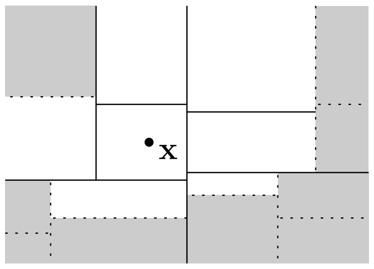


Figure 1: Neighborhood of the sample point $\mathbf{x}$. The bright areas denote the neighboring cells and the solid lines represent the neighboring splitting planes.

As mentioned in Section 3.1 the way of storing the nodes of the kd-tree in the vector constitutes that the position of every node and leaf in the tree is clearly determined by the binary representation of its index. Thus, cells and splitting planes can be identified with their index in the vector.

The computation of the neighborhood for a cell is divided in two main steps. First we compute direct neighbors, i.e. cells and planes that resulted from the last three steps of the tree building process. These neighbors can be directly determined by the index of the observed sample point $\mathbf{x}$.

In the second step we want to get neighbors for the three other faces of the cell of $\mathbf{x}$. The maximum number of these indirect neighbors is in contrast to the direct neighbors not constant. More precisely it is $O(\sqrt[3]{n^2})$ in our case of a three-dimensional kd-tree. But even in the worst case only a small number of cells reach this number of indirect neighbors. Averaged, a cell has at most nine indirect neighbors for each face.

To compute the indirect neighbors, the three splitting planes covering the remaining faces of the cell of **x** are directly obtained from the cells index. Subsequently for every found splitting plane a binary search on the opposite side of **x** is done and delivers the remaining indirect neighbors.

All in all, the neighborhood computation uses with small exceptions only informations provided by the indexing scheme. Thus, the search for all neighbors uses mostly binary operations on the cells indices and has consequentially a performance that is similar to nearest neighbor search in kd-trees.

3.3 Angle and Maximum Distance Criterion

For our intention, i.e. interpolating isopoints between neighboring sample points, it is very important having neighbors not lying behind other neighbors. One problem occurring in this case is shown in a two-dimensional example in Figure 2.

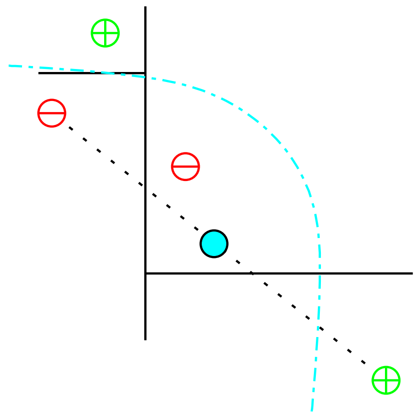


Figure 2: Example of a bad isopoint interpolation caused by the small angle between the two negative sample points and the positive sample point in the lower right cell. The desired isocontour is drawn in light blue. Note that the interpolated isopoint is farther away from the isoline than each sample point.

To avoid such situations we establish an angle criterion extending the angle criterion method Linsen and Prautzsch [Linsen & Prautzsch, 2001] used for point-based surface representations to volume data. It limits the maximum angle between neighbors to one sample point. If the criterion is violated by a pair of neighbors, the farther neighbor is omitted from the neighborhood.

A second criterion limits the maximum distance between neighbors. This criterion arises from the fact, that errors of linear interpolation can grow enormously with increasing distance. If a neighbor's distance to the observed point exceeds the distance threshold it will also be omitted from the neighborhood.

4 Level-Set Segmentation

Due to varying particle number densities of the SPH data the direct isosurface extraction step can produce rough isosurfaces in sparse areas. To generate smoother segmentations, we propose the application of a level-set segmentation method to the data prior to isosurface extraction.

The basic idea of level-set methods goes back to Osher and Sethian [Osher & Sethian, 1988], who first described the evolution of a closed hypersurface. In general a so called level-set function φ is first initialized on the sample points. Subsequently this level-set function is successively adapted to the data set f. For this process a variety of approaches exist. A detailed overview of this field of research is given by Osher and Fedkiw [Osher & Fedkiw, 2003].

For our purposes, i.e. the smooth segmentation of SPH data, we used the level-set process equation

$$\frac{\mathrm{d}\varphi}{\mathrm{d}t} = \left(a\left(f - f_{\mathrm{iso}} - \varphi\right) + b\kappa_{\varphi}\right)|\nabla\varphi| \ ,$$

which models hyperbolic normal advection, weighted with factor $a > 0$, and mean curvature flow, weighted with factor $b > 0$. This means that the level-set function moves towards the data set with the factor a and the mean curvature of the resulting isosurface is smoothed with the factor b.

This level-set process is done using a forward Euler time integration until the level-set function does not move any more. The needed derivatives are approximated using a least-squares approach. Subsequently the zero level set is extracted using direct isosurface extraction, as described in Section 3.

5 Case Study: White Dwarf Black Hole Encounters

Our Galaxy, the Milky Way, is surrounded by a halo of more than 150 globular clusters, spherical agglomerations of typically 100.000 stars. The very large stellar number densities in their centers are believed to produce via stellar collisions black holes between several hundred and several thousand solar masses, so-called "intermediate mass black holes" (intermediate between stellar mass black holes that result from a supernova explosion and the supermassive black holes that are observed in the centers of galaxies).

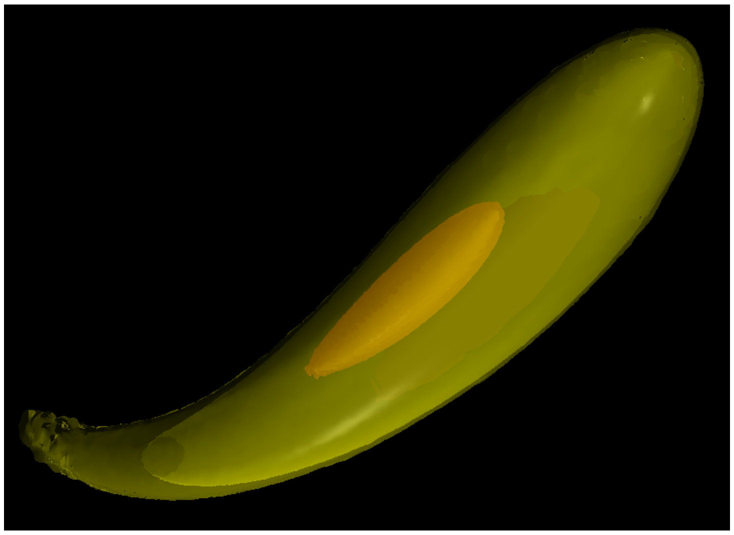


Figure 3: Rendering of two surfaces segmenting the white dwarf data set with respect to the density. The transparent yellow surface represents the isosurface to the isovalue $2 \cdot 10^4 \mathrm{g/cm}^3$, whereas the red surface is the isosurface to the isovalue $2 \cdot 10^5 \mathrm{g/cm}^3$.

Being old, globular clusters also contain large numbers of the remnants of solar-type stars, so-called white dwarfs. We simulated the fly-by of white dwarfs close to such an intermediate mass black hole [Rosswog and Ramirez-Ruiz 2007, to be submitted]. In these simulations the white dwarf was modeled with up to 7.000.000 SPH particles. During the fly-by the white dwarfs become very strongly compressed and the resulting temperature increase triggers the thermonuclear ignition of the white dwarf material. Thus for each

SPH particle the evolution of the nuclear abundances and the resulting energy release is calculated via a nuclear reaction network [Hix et al., 1998]. Since during the compression the required nuclear reaction time steps are smaller than the hydrodynamic time steps by several orders of magnitude, the hydrodynamics and the nuclear reactions are calculated by two different time-integration schemes in an operator splitting fashion.

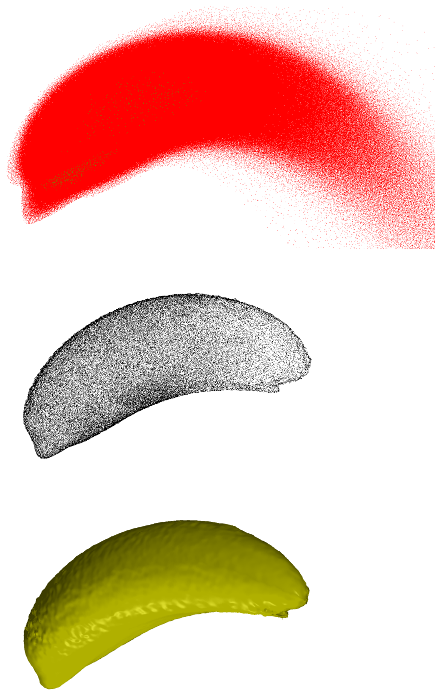


Figure 4: Illustration of the surface extraction pipeline on a later point in time of the white dwarf simulation. In the topmost picture, the seven million SPH data points are shown. The middle picture shows the extracted isopoints. The extracted isosurface is rendered in the lowermost picture.

Such a simulation usually produces of the order one thousand data dumps, where each dump contains more than one Gbyte of physical data represented on the SPH particles. To understand and analyze such simulations it is of uttermost importance to be able to visualize the physical variables in a 3D rendering like the one shown in Figure 3. As the outcome of such simulations is a priori unknown, there are no "standard tasks" that can be routinely performed on the data sets. Therefore, the key to an efficient scientific production process is a very rapid and flexible 3D visualization tool that allows to interactively explore the large data sets.

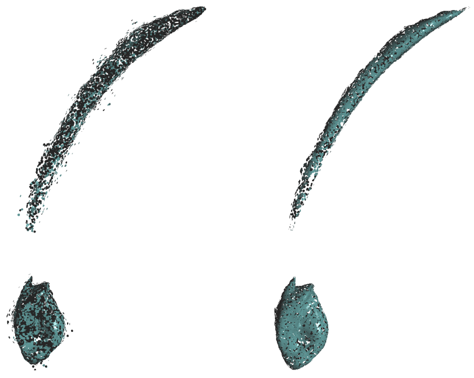


Figure 5: Comparison between direct isosurface extraction on the left side and level-set segmentation on the right side for a white dwarf simulation with 500.000 particles. To illustrate the significant advantage of the level-set approach, we show a rendering of the surface points with very small splats.

For every data set with seven million particles, the presented method takes ten seconds to build the kd-tree and another eight seconds to generate the neighborhood informations. Subsequently every isosurface-point-cloud extraction lasts eight seconds, i.e. no recalculation of the kd-tree or neighborhood is needed. The extracted 3D point cloud can be explored interactively. If a relevant isosurface is found, it can be rendered in at most one minute. An illustration of the surface-extraction pipeline is shown in Figure 4.

A comparison between isosurface extraction and the level-set approach on a sparse sampled 500.000 particles white dwarf data set is shown in Figure 5.

6 Conclusion

We presented a visualization method for smoothed particle hydrodynamics simulation data. To visualize a scalar field associated to the particles, we directly extract a surface segmenting the domain of the particles with respect to a given isovalue without any 3D mesh generation or reconstruction over a structured grid. For this purpose we use a spatial domain partitioning using a kd-tree and an indexing scheme for efficient neighbor search. Between appropriate neighbors, surface points are computed using linear interpolation with respect to the isovalue. To improve the smoothness of the extracted surface a level-set method is applied to the data prior to surface extraction. The resulting point cloud is visualized using point-based rendering techniques.

The presented application pipeline was tested on a case study dealing with the simulation of a white dwarf passing a black hole. After a short preprocessing the extracted isopoint clouds can be explored interactively, whereas the isovalue or the physical quantity can be changed in nearly interactive time. This fast and interactive data visualization approach allows for an efficient extraction of the physical information contained in the simulation data and substantially speeds up the otherwise cumbersome analysis process.

References

[Bentley, 1975] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9), 509–517.

[Co & Joy, 2005] Co, C. S. & Joy, K. I. (2005). Isosurface Generation for Large-Scale Scattered Data Visualization. In G. Greiner, J. Hornegger, H. Niemann, & M. Stamminger (Eds.), *Proceedings of Vision, Modeling, and Visualization 2005* (pp. 233–240).: Akademische Verlagsgesellschaft Aka GmbH.

[Franke & Nielson, 1991] Franke, R. & Nielson, G. M. (1991). *Geometric Modeling: Methods and Applications*, chapter Scattered Data Interpolation: A Tutorial and Survey, (pp. 131–160). Springer Verlag, New York.

[Gingold & Monaghan, 1977] Gingold, R. A. & Monaghan, J. J. (1977). Smoothed particle hydrodynamics – theory and application to non-spherical stars. *Royal Astronomical Society, Monthly Notices*, 181, 375–389.

[Hix et al., 1998] Hix, W. R., Khokhlov, A. M., Wheeler, J. C., & Thielemann, F.-K. (1998). The Quasi-Equilibrium-reduced alpha -Network. *Astrophysical Journal*, 503, 332–+.

[Linsen et al., 2007] Linsen, L., Müller, K., & Rosenthal, P. (2007). Splat-based ray tracing of point clouds. *Journal of WSCG*, 15(1–3).

[Linsen & Prautzsch, 2001] Linsen, L. & Prautzsch, H. (2001). Global versus local triangulations. In J. Roberts (Ed.), *Proceedings of Eurographics 2001, Short Presentations.*

[Lucy, 1977] Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82, 1013–1024.

[Monaghan, 2005] Monaghan, J. J. (2005). Smoothed particle hydrodynamics. *Reports of Progress in Physics*, 68, 1703–1759.

[Osher & Fedkiw, 2003] Osher, S. & Fedkiw, R. (2003). *Level set methods and dynamic implicit surfaces.* Springer.

[Osher & Sethian, 1988] Osher, S. & Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formualtions. *Journal of Computational Physics*, (79), 12–49.

[Rosenthal & Linsen, 2006] Rosenthal, P. & Linsen, L. (2006). Direct isosurface extraction from scattered volume data. In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization* (pp. 99–106).

[Treece et al., 1999] Treece, G. M., Prager, R. W., & Gee, A. H. (1999). Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4), 583–598.

[Vesterlund, 2004] Vesterlund, M. (2004). Simulation and rendering of a viscous fluid using smoothed particle hydrodynamics. Master's thesis, Umea University, Sweden.

Visualization of the Gödel Spacetime

Frank Grave[1] and Thomas Müller[1]
and Günter Wunner[2] and Thomas Ertl[1]
and Michael Buser[3] and Wolfgang Schleich[3]

[1]Visualization and Interactive Systems Group, University of Stuttgart, Germany

[2]1. Institut für Theoretische Physik, University of Stuttgart, Germany

[3]Institute of Quantum Physics, Ulm University, Germany

`frank.grave@vis.uni-stuttgart.de`,
`thomas.mueller@vis.uni-stuttgart.de`
`michael.buser@uni-ulm.de`

Abstract

The Gödel metric - an exact solution to Einstein's Field Equations - describes a rotating, homogeneous but non-isotropic spacetime. There exists a "Gödel Horizon" beyond which an observer cannot see and where time travel might be possible.

With four-dimensional raytracing we are able to adopt an egocentric point of view. Our focus lies on the visual appearance of the Gödel Universe itself as actually seen by an observer located within this universe.

In this paper we will discuss the appearance of static objects to avoid the effects of motion and time travel. These cases will be discussed in future work. We will show and discuss what an observer surrounded by a cylindrical coordinate grid would actually see if the rotational parameter of the spacetime is varied. Then the coordinate grid will be replaced by a grid of equidistant points. The distance between these points will be determined by light pulses starting from the observer's position. By this means, we create a grid with a real physical distance concept. This will illustrate several features of the Gödel Universe and hopefully give some insight into this non-trivial spacetime.

1 Introduction

On the occasion of Einstein's 70th birthday Gödel released the first model of a universe in which time travel is possible ([Gödel, 1949]). This exact solution of Einstein's Field Equations is no realistic description of our universe as it describes a rotating universe without Hubble Expansion. Nevertheless, it can be seen as an illustration of a hypothetical universe. For every observer there exists a Gödel radius (or Gödel Horizon), beyond which the lightcones penetrate the plane of constant coordinate time ([Hawking & Ellis, 1973]). This associates with Gödel's assertion of time travel in this universe.

With Gödel's Universe discussion of time travel to the past within general relativity began. By now, we know of many metrics in which time travel is possible. Even in the Kerr-Metric of a rotating black hole we find this possibility ([Visser, 1996]). [Frolov & Novikov, 1990] come to the conclusion that ,,the interaction beween a wormhole with its surrounding matter and the external gravitational field almost inevidably transfroms it into a time machine".

We are especially interested in answering the question what a real physical observer located within a certain spacetime would actually *see*. Our method to answer that question is special- and general-relativistic raytracing.

2 Previous Work

The Gödel solution of Einstein's Field Equation is very easy, because it exhibits a high symmetry which is expressed in not less than five Killing-Vectors ([Stephani, 1991]). This allows many calculations, like the solution of the geodesics equation to be accomplished analytically. We can find discussions of the geodesics in [Kundt, 1956], [Chandrasekhar & Wright, 1961] and [Novello et al., 1983]. More recently, the Sagnac-Effect of Gödel's Universe was discussed in [A. Delgado, 2002] and [Kajari et al., 2004a].

[Kajari et al., 2004b] found an elegant description of Gödel's Universe using cylindrical coordinates $x^\mu = (t, r, \phi, z)$. Thus, the Gödel metric in

Kajari's formulation reads:

$$\begin{aligned} \mathrm{d}s^2 &= -c^2\,\mathrm{d}t^2 + \frac{1}{1+\left(\frac{r}{2a}\right)^2}\,\mathrm{d}r^2 + r^2\left(1-\left(\frac{r}{2a}\right)^2\right)\mathrm{d}\phi^2 \\ &\quad + \mathrm{d}z^2 - r^2\frac{2c}{\sqrt{2}a}\,\mathrm{d}t\,\mathrm{d}\phi \end{aligned} \tag{1}$$

where $r = 2a$ is the Gödel radius. An observer located at the origin of this coordinate system cannot see beyond this radius. Due to the homogenity of this metric, this statement is valid for every observer and the Gödel radius around him. All images have been calculated in geometrical units $c = G = 1$.

At the end of the eighties the necessary computing power existed to visualize relativity by means of raytracing. The first relativistic simulation was a special relativistic flight through the Brandenburg Gate in 1991 ([Ruder & Ruder, 1991]). [Zahn, 1991] was the first to offer an approach to realize observers (cameras) on arbitrary woldlines. [Weiskopf, 2001] realized several approaches to visualize many areas of the theory of relativity. He extended the *Ray Visualisation System* (RayViS, [Gröne, 1996]) to four dimensions in order to solve the geodesic equation of light in curved four-dimensional spacetimes.

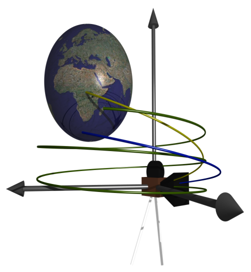


Figure 1: Raytracing in curved spacetimes. A camera traces null geodesics (photons) back in time. The image taken by the camera depends on where the geodesics intersect objects in the scenery.

In RayViS, the objects defining a scenery have to be implemented as coordinate objects making their appearance dependent on which set of coordinates are used. But an observer should always see a scene in the same way, no matter in which coordinate system his environment was described.

[Müller, 2006] solved this problem in designing a new raytracer especially adjusted to the requirements of special and general relativity. In his raytracer GeoViS we have the possibility to describe objects as well as the observer (camera) with respect to a local reference frame. Therefore, all objects are unambiguously described with pseudo-cartesian coordinates. Recent works on the visualization of black holes or wormholes can be found in [Grave, 2004], [Fechtig, 2004] and [Zatloukal, 2005].

3 Theoretical Foundation

If we take a closer look at equation (1), we find that for $2a \to \infty$

$$g_{\mu\nu} \to \operatorname{diag}\left(-c^2, 1, r^2, 1\right), \tag{2}$$

which is the Minkowski metric in cylindrical coordinates. We can use this fact to build a scenery in flat spacetime and then study the visual effects of Gödel's Universe by decreasing its Gödel radius.

Furthermore, the metric is flat on the symmetry axis z. But in cylindrical coordinates there is a coordinate singularity at $r = 0$ where ϕ is undefined. So if we want to place our observer on this axis we have to transform the metric to cartesian coordinates $x^\mu = (t, x, y, z)$ where $x = r\cos(\phi), y = r\sin(\phi)$. The result is

$$\begin{aligned} \mathrm{d}s^2 &= -c^2\,\mathrm{d}t^2 + \frac{-y^4 - x^2y^2 + 16a^2}{4a^2(4a^2 + x^2 + y^2)}\,\mathrm{d}x^2 + \frac{-x^4 + x^2y^2 + 16a^2}{4a^2(4a^2 + x^2 + y^2)}\,\mathrm{d}y^2 \\ &\quad + \mathrm{d}z^2 + \frac{2cy}{\sqrt{2}a}\,\mathrm{d}x\,\mathrm{d}t - \frac{2cx}{\sqrt{2}a}\,\mathrm{d}y\,\mathrm{d}t \end{aligned} \tag{3}$$

Because we want to express all objects in a local reference frame or so called tetrad, we need the relation between a four-vector relative to a tetrad at a given event and its coordinate representation ([Nakahara, 1990]). [Fechtig, 2004] found a universal correlation for an arbitrary stationary spacetime. With this correlation, we obtain the transformation rule

$$V^\mu = V^{\hat{\alpha}} e_{\hat{\alpha}}{}^{\mu} \tag{4}$$

of a four-vector between a tetrad and cylindrical coordinates. Then, we transform this vector into cartesian coordinates. We get

$$
\begin{aligned}
V^t &= V^{\hat{t}}A - V^{\hat{y}}BD, && \text{(5a)}\\
V^x &= -V^{\hat{t}}\Omega A y + V^{\hat{x}}\frac{x}{\sqrt{(x^2+y^2)g_{rr}}} + V^{\hat{y}}CDy, && \text{(5b)}\\
V^y &= V^{\hat{t}}\Omega A x + V^{\hat{x}}\frac{y}{\sqrt{(x^2+y^2)g_{rr}}} - V^{\hat{y}}CDx, && \text{(5c)}\\
V^z &= V^{\hat{z}}, && \text{(5d)}
\end{aligned}
$$

where

$$
\begin{aligned}
A &= -(g_{tt} + \Omega g_{t\phi} + \Omega^2 g_{\phi\phi}), && \text{(6a)}\\
B &= g_{t\phi} + \Omega g_{\phi\phi}, && \text{(6b)}\\
C &= -(g_{tt} + \Omega g_{t\phi}), && \text{(6c)}\\
D &= (g_{tt}g_{\phi\phi} - g_{t\phi}^2)(g_{tt} + 2\Omega g_{t\phi} + \Omega^2 g_{\phi\phi}), && \text{(6d)}
\end{aligned}
$$

where Ω is the angular velocity of an object in the scenery and $g_{\mu\nu}$ are the metric components of (1). V^μ denotes a component of a vector in the coordinate direction $\{\partial_\mu\}$, $V^{\hat{\alpha}}$ is a vector component relative to a tetrad $\{e_{\hat{\alpha}}\}$ and ${e_{\hat{\alpha}}}^\mu$ is the transformation matrix between both representations. We can easily obtain the transformation matrix ${e_{\hat{\alpha}}}^\mu$ by inserting (5) into the transformation rule (4).

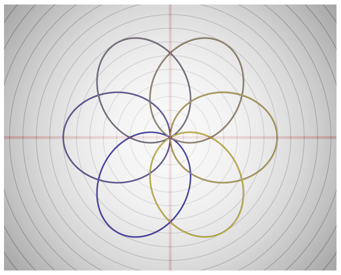


Figure 2: Here, six null geodesics starting at the origin of the xy-plane in a Gödel Universe with $r = 8$ are shown. The light rays evolve counter-clockwise into the future. When using raytracing, geodesics have to be traced into the past. In that case, they follow a clockwise rotation. After touching the Gödel radius, they all refocus in both cases at the origin at the same time.

In [Kajari et al., 2004b] we find a parameterisation for geodesics in the xy-plane.

4 Implementation

4.1 Regular coordinate grid

4.1.1 Observer at $z = 10$

At first, we want to discuss what an observer sees if he views along the z-axis. He is located at $z = 10$ and observes a polar coordinate grid in the xy-plane. The result for four values of the Gödel radius $r = 2a$ is shown in figure 3. The top row shows what the observer actually sees and the images in the bottom row each depict four spatial projections of geodesics. These geodesics correspond to the corner pixels of the image above.

The first row represents the observer's impression for a nearly flat spacetime ($2a = 4000$). The coordinate grid is virtually undistorted and the geodesics in the lower subpicture are staight lines. As the Gödel radius is decreased in the following row, we can see that the geodesics follow a *clockwise* rotating path similar to figure 1. But these geodesics are still in relationship with the corners of the image above, i.e. with the corners of the oberver's field of view. Hence, the percepted coordinate grid is rotated *counter-clockwise*. We also know that Gödel's Universe in eq. (3) is flat along the z-axis. Therefore, this rotational effect is smaller in the centre of the observers field of view. This explains why the radial axes are not percepted as straight lines.

The third row shows another feature of this metric. Besides the increasing torsion of the coordinate grid it also appears magnified. Magnification of an object always occurs if geodesics corresponding to the observers field of view originate from a very small area of this object. We know (compare figure 1) that geodesics in the xy-plane only reach the Gödel radius before they refocus. So the magnification has two reasons. On the one hand, geodesics do not reach arbitrarily big radii. On the other hand, they are driven back to $r = 0$. In the last row, this effect becomes very extensive. If we take a closer look at the last explanatory subpicture of figure 3, we see that the geodesics intersect slightly above the xy-plane. If the coordinate grid had been placed there, it had appeared infinitely magnified from the observers point of view. In fact, the displayed grid also appears point inverted because of the intersection of the geodesics above the plane. Additionally, the surface brightness appears reduced like in special relativitsic abberation. The total luminosity of one fixed area of the grid stays the same, but it is seen in a larger field of view making this area appear darker.

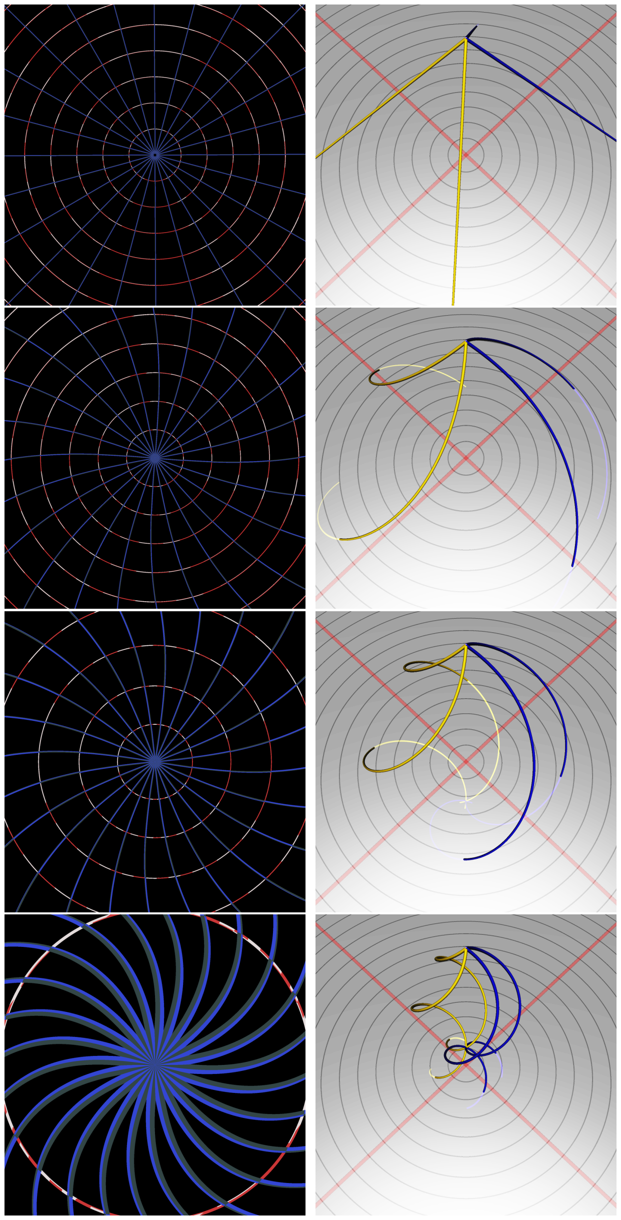


Figure 3: We are looking at a polar coord grid from $z = 10$ and a field of view of $60° \times 60°$. The left column shows how the observer perceives the coordinate grid, the right column shows explanatory geodesics. Starting from the top, the Gödel radius is $r = 4000, 20, 10$ and 4.

4.1.2 Observer at $z = 1$

Now we place the observer at $z = 1$ and let him take a look into the positive x direction. Furthermore, an earth texture mapped onto a sphere with a radius of $r = 5$ is placed into the scenery. We will study again four cases of the Gödel radius. The first value is rather large in order to obtain a nearly flat spacetime. The following three values of the Gödel radius are about the radius of the earth sphere. In figure 4 the result of this simulation can be seen.

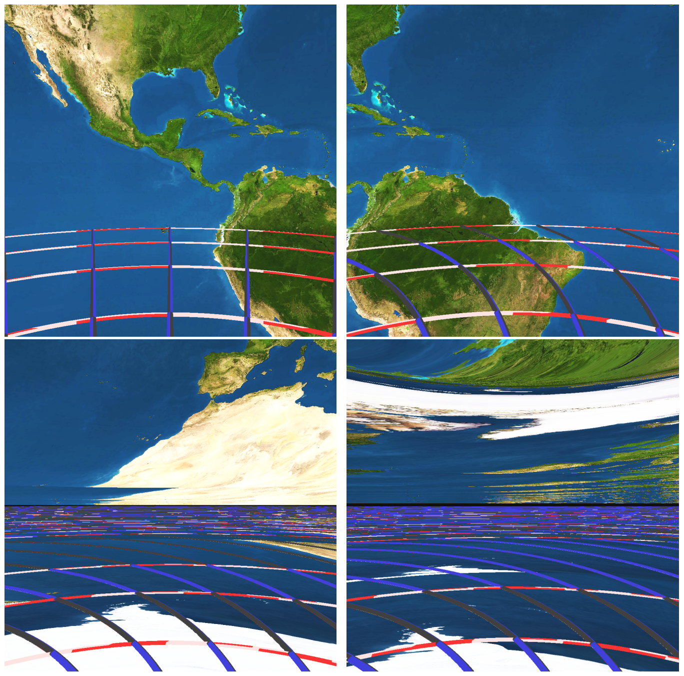


Figure 4: The perceptions of the observer at $z = 1$. From left to right starting at the top the Gödel radius is $2a = 4000, 10, 4.8$ and 4, the earth radius is $r = 5$.

As expected, the first picture in figure 4 shows the scene like in flat spacetime. Then, after decreasing the Gödel radius to $r = 10$, the scene appears to be rotated to the left. But, after decreasing the Gödel radius *below* the radius of the earth sphere, our observer gets some unexpected impressions of this scenery. Earth's surface for example appears multiple times. We have to take a look at the explanatory geodesics in figure 5 to understand that.

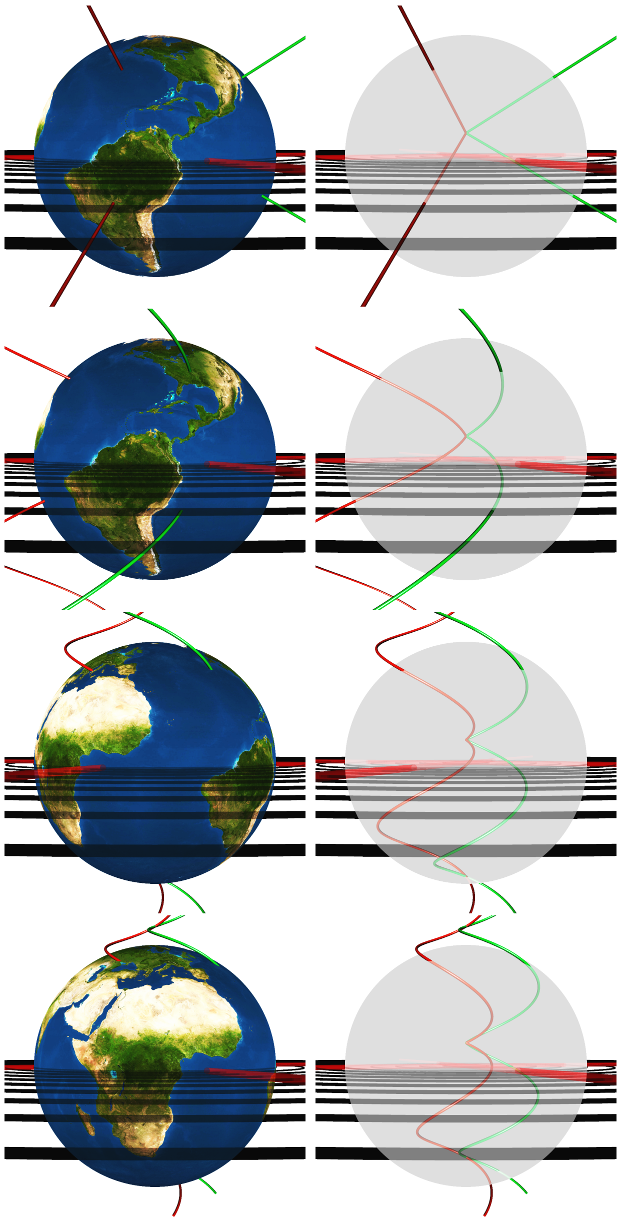


Figure 5: Plot of several explanatory geodesics for the images in figure 4. The parameters are still $2a = 4000, 10, 4.8, 4$ and $r = 5$. Nota bene: the camera observing the earth changes its position in the last two rows.

In the last two rows of figure 5 the rotation of the Gödel universe is so strong that the relevant geodesics which reside in the xy-plane *never* intersect the sphere. This explains the horizontal black line in the last subpictures of figure 4. Even geodesics reaching the observer just above or below his horizon (marked by the black line) don't originate from earths equator. The absorbed photons originate mostly from earth's poles (most of last picture is white) from where they spiralled their way to the observer.

To understand these repetitions in the observed earth sphere for small Gödel radii, we can take a look at figures 6 and 7. In figure 6 we study a bundle of photons which arrive the observer under an angular range of 60°. These photons create the picture along the vertical middle axis of the observer's field of view. In figure 6 we see that the geodesics intersect the earth sphere in some distance to the equator. As a consequence, this area around the equator can not be seen by the observer.

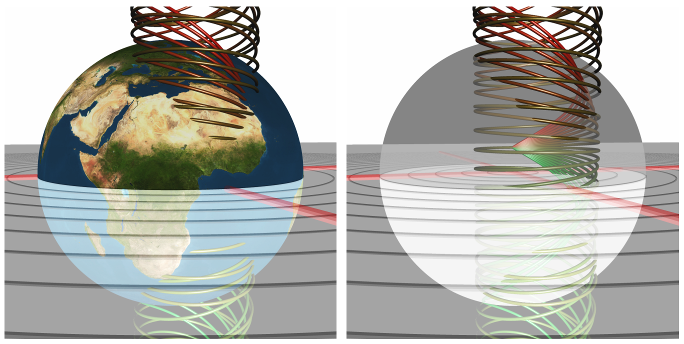


Figure 6: Geodesics in a Gödel Universe with $2a = 4.8$. These geodesics correspond to a vertical field of view of 60°

If we narrow the observers field of view to about 6° (see figure 7), we can clearly see that even these geodesics intersect the earth sphere in the same region as in figure 6. This explains why the observer sees the visible region of this earth multiple times. If we made his viewing field even smaller we would basically get the same picture. Therefore, the visible region can be seen infinitely often. We can clearly see that due to these infinite repetitions or infinite magnifications (see fig. 3) of certain visible areas, wavefronts of photons intersect themselves. Therefore, caustics can be found in the Gödel Universe.

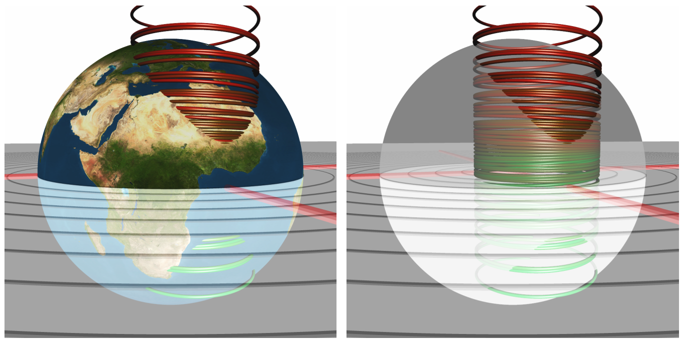


Figure 7: Geodesics in a Gödel Universe with $2a = 4.8$. These geodesics correspond to a vertical field of view of only about 6°

4.2 Equidistant coordinate grid

In this section we want to construct a pseudo-polar grid in the xy-plane with equidistant points. The simplest way to accomplish that is to use light pulses starting at the observer's position at $(x, y, z) = (0, 0, 0)$.

Along a null geodesic we can associate the length of a piece of the curve itself with the elapsed coordinate time because $\mathrm{d}s^2 = 0$. And for this metric in particular the elapsed coordinate time is the same as the elapsed proper time for an observer resting in $(x, y, z) = (0, 0, 0)$. So we can simply define lengths by dividing null geodesics which start at the origin by dividing them in pieces of constant proper time intervals as measured by this observer.

The main question is if we should regard light pulses *emitted* at the observers position or if we use light pulses *absorbed* by the observer's eye instead. Emitting photons means tracing their path into the future, absorbing them corresponds to a negative time direction when tracing them. We decide to follow the idea of backtracing the photons in order to create a grid which appears as undistorted as possible. In taking a closer look at figure 1 the association between backtracing photons and an undistorted grid can be understood by the following idea. If we placed a number of objects on the green coloured and backtraced geodesic, then all of these objects could be seen straight ahead from the cameras point of view.

In figure 8 we can see the result of the corresponding simulation. The parameters used for the Gödel radius are the same as in figure 4. The yellow-grey spheres mark equidistant points along a geodesic hinted with the blue-grey lines. The radius of all spheres is 0.1 if measured with respect to the local frame at the center of each of them. The observer himself is *not* positioned at the origin of the coordinate system but at $z = 0.5$ to have some overview on the grid. We neglect the error made by this translation.

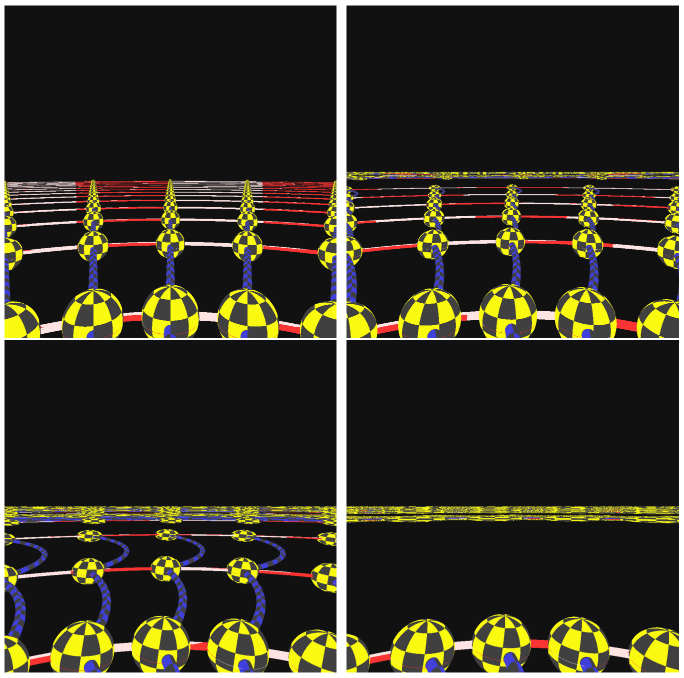


Figure 8: The perceptions of the observer at $z = 0.5$ when he is looking at an equidistant polar coordinate grid. From left to right starting at the top the Gödel radius is $2a = 4000, 10, 4.8$ and 4. The yellow-grey spheres are local objects which define the grid.

The radial axes (blue-grey) appear bent because these axes are still implemented as coordinate objects. As the Gödel radius decreases, the maximum radial extent of the grid decreases as well. Near the horizon the observer sees again multiple repetitions of the grid as explained in the previous section.

5 Conclusions

In this work we have discussed several visual effects of the Gödel Universe. Beneath the visualization of rather obvious effects like the disappearance of objects located beyond the Gödel radius, unexpected properties like infinite repetition of some perceived areas could be explained. In future work interesting topics like the visual appearance of slow or fast moving objects or observers will be discussed. In this paper we only investigated geometrical effects of the Gödel Universe, so redshift and intensity effects are not included yet. We plan to vizualize these effects as well, including a detailled view on caustics, lightcone structures and wavefronts of photons.

The most interesting question of all concerning the Gödel Universe is how a e.g. blinking particle appears when it travels along a worldline leading it to travel into the past. Therefore the particle has to be accelerated beyond the Gödel radius itself.

6 Acknowledgments

We want to thank the Deutsche Forschungsgemeinschaft DFG for funding this project.

References

[A. Delgado, 2002] A. Delgado, W. P. S. u. G. S. (2002). *NJP*, 4(37.1).

[Chandrasekhar & Wright, 1961] Chandrasekhar, S. & Wright, J. P. (1961). *Proc. Nat. Acad. Sci*, 47(341).

[Fechtig, 2004] Fechtig, O. (2004). Physikalische Aspekte und Visualisierung von stationären Wurmlöchern. Master's thesis, Universität Stuttgart.

[Frolov & Novikov, 1990] Frolov, V. P. & Novikov, I. D. (1990). Physical effects in wormholes and time machines. *Phys. Rev. D*, 42(4), 1057–1065.

[Gödel, 1949] Gödel, K. (1949). An Example of a New Type of Cosmological Solutions of Einstein's Field Equations of Gravitation. *Reviews of Modern Physics*, 21(3), 447–450.

[Grave, 2004] Grave, F. (2004). Visualisierung zum Gravitationskollaps und Wellenfronten in der Allgemeinen Relativitätstheorie. Master's thesis, Universität Stuttgart.

[Gröne, 1996] Gröne, A. (1996). *Entwurf eines objektorientierten Visualisierungssystems auf der Basis von Raystracing*. PhD thesis, Eberhard-Karls-Universität Tübingen.

[Hawking & Ellis, 1973] Hawking, S. W. & Ellis, G. F. R. (1973). *The Large Scale Structure of Space-Time*. Cambridge University Press.

[Kajari et al., 2004a] Kajari, E., Walser, R., Schleich, W. P., & Delgado, A. (2004a). *Gen. Rel. Grav.*, 36(2289).

[Kajari et al., 2004b] Kajari, E., Walser, R., Schleich, W. P., & Delgado, A. (2004b). Sagnac Effect of Gödel's Universe. *Gen. Rel. Grav.*, 36(10), 2289–2316.

[Kundt, 1956] Kundt, W. (1956). *Zeitschrift für Physik*, 145(611).

[Müller, 2006] Müller, T. (2006). *Visualisierung in der Allgemeinen Relativitätstheorie*. PhD thesis, Eberhard-Karls-Universität Tübingen.

[Nakahara, 1990] Nakahara, M. (1990). *Geometry, Topology and Physics*. Institute of Physics Publishing.

[Novello et al., 1983] Novello, M., ao Soares, I. D., & Tiomno, J. (1983). *Phys. Rev. D*, 27(779).

[Ruder & Ruder, 1991] Ruder, H. & Ruder, M. (1991). *Die Spezielle Relativitätstheorie*. Springer.

[Stephani, 1991] Stephani, H. (1991). *Allgemeine Relativitätstheorie*. Deutscher Verlag der Wissenschaften, 4 edition.

[Visser, 1996] Visser, M. (1996). *Lorentzian Wormholes - From Einstein to Hawking*. Springer-Verlag New York, Inc.

[Weiskopf, 2001] Weiskopf, D. (2001). *Visualization of Four-Dimensional Spacetimes*. PhD thesis, Eberhard-Karls-Universität Tübingen.

[Zahn, 1991] Zahn, C. (1991). Vierdimensionales Raytracing in einer gekrümmten Raumzeit. Master's thesis, Universität Stuttgart.

[Zatloukal, 2005] Zatloukal, M. (2005). Visualisierung in der Kerr-Raumzeit. Master's thesis, Eberhard-Karls-Universität.

GPU Friendly Rendering of Large LIDAR Terrains

Shalini Venkataraman and Werner Benger and Amanda Long

Center for Computation and Technology, Louisiana State University, USA

`shalini@cct.lsu.edu,werner@cct.lsu.edu, along@cct.lsu.edu`

Abstract

The extremely high-resolution terrain data from LIDAR (LIght Detection And Ranging) mapping results in hundreds of millions of triangles using brute-force triangulation methods. This is beyond the capabilities of interactive rendering. Alternative approaches such as downsampled subsets do not provide sufficient image quality. In order to still be able to depict the full quality of the dataset, continuous Level-of-Detail (LOD) techniques are necessary that dynamically simplify the mesh at run-time depending on the view point. Traditionally, terrain rendering techniques attempt to create a "perfect set" of triangles by performing extensive LOD computations on the host processor. Although such techniques generate considerably fewer triangles on the CPU, they underutilize the GPU, thus not achieving optimal performance.

We look at optimized implementations of such techniques on commodity graphics hardware. In particular, we draw upon existing algorithms like Geometrical MipMapping which is analogous to texture mip-mapping but extended to height-fields. Our implementation uses OpenGL extensions such Vertex Buffer Objects (VBO) for faster drawing and efficient management of geometry. Automatic texture coordinate generation functionality in OpenGL will also be used to apply colormaps as well as drape additional aerial photography on the 3D representation. We will conclude with a discussion of future work such as support for out of core datasets.

1 Motivation

Visualization of geoscientific data faces challenges of size, integration and representation. As processing, memory, and bandwidth capabilities continue to increase, so does the resolution of scanned landscapes and display technology. Today, satellite range scans comprised of billions of samples are commonplace, posing a formidable challenge to existing rendering, data management and display techniques. For example, the height-field representing the earth's topography at 1m resolution would require 2 Petabytes! However, modern high-end graphics cards have maximum 2GB on-board memory.

In this paper, we describe our efforts in visualizing large terrains from LIDAR [1] mapping implemented as part of our Hurricane Katrina Visualization project[Venkataraman et al., 2006]. Katrina was one of the most powerful hurricanes ever to hit the US, and certainly the most devastating. The resulting wind-driven surge impacted millions by flooding major parts of New Orleans, a city that lies mostly below sea-level. For many coastal modelers, the interaction of the surge with topology is crucial to understanding regions more prone to devastation. This information is critical in the construction of levees and in emergency response management such as planning evacuation routes.

Figure 1a) shows our input data - the terrain of New Orleans containing 11,000x7,000 points. This height field is stored as a grayscale image where the color of each pixel represents a 8-bit height value. A colormap is applied to delineate the areas at, below and above sea-level. Figure 1c) shows a brute-force triangulation of this grid resulting in about 254 million triangles that took 30s to render on a high-end system. Clearly, this motivates us to employ some Level Of Detailing (LOD) technique to cull the data dynamically.

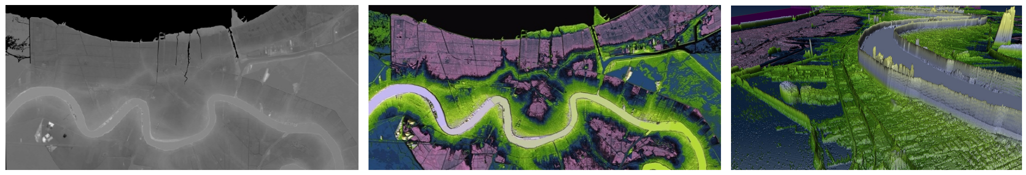


Figure 1: Different views of the LIDAR terrain: a) Grayscale data b) Colormapped to show sea-level(blue), above(green) and below(magenta) c) Perspective rendering showing the levees along the Mississippi.

[1]LIght Detection And Ranging `http://www.lidarmapping.com`

2 Previous Work

LOD techniques attempt to generate a finite number of representations for the underlying mesh, each at a different level of detail. The appropriate level is selected at run-time, depending on viewer position and orientation to show more details in foreground closer to the camera than in the distance. These algorithms can be broadly divided into two categories - *adaptive triangulation* (CPU-bound) and *block-based* (GPU-bound).

Prior to the widespread adoption of GPUs, the class of adaptive triangulation algorithms focused on minimizing the number of triangles drawn at the expense of extra CPU processing (Figure 2a). The Real-time Optimally Adapting Meshes (ROAM) [Duchaineau et al., 1997] algorithm is a classic example. This method attempts to create an optimal set of triangles for a terrain using the CPU to perform the extensive calculations at run-time and spew the vertices one at a time to the graphics for rendering. Successors to ROAM [Lindstrom & Pascucci, 2002] use more efficient triangle fans to manage geometry. However, as these are still computed at run-time, it is impossible to retain any geometry in video memory necessitating costly data transfers on the PCI bus.

Todays graphics hardware is capable of processing and rendering a large amount of triangles. Taking advantage of this feature, the class of block-based algorithms such as Geomipmapping[de Boer, 2000] cache the geometry of the terrain on a per-block basis for all levels in graphics memory. At run-time, CPU involvement is limited to merely pointing the GPU to the memory address of the blocks to be rendered (Figure 2b). While this pushes more triangles to the rendering pipeline, it is faster on modern graphics because of the low run-time CPU overhead and low utilization of the PCI bus.

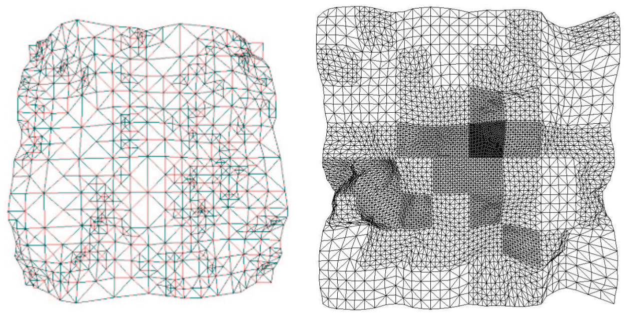


Figure 2: LOD methods a) Adaptive triangulation b) Block-based

3 Our approach

Our implementation is inspired by the Geometry MipMapping (Geomipmapping) algorithm. First, the terrain is divided into patches. Then, multiple resolutions of the patches are generated and the actual errors in screen space caused by these rough models are pre-computed. At run-time, after visibility culling, the level to render each patch is computed according to the viewpoint and a user-specified screen space error. Further optimizations are done by caching the patches in vertex buffers on the graphics card for fast access, using triangle strips and texture-based colormaps for efficient rendering and low memory overhead. Our API enumerated below is based on OpenGL and currently invoked within the Amira [Stalling et al., 2005] framework.

- *Init* : Handles all the preprocessing and generally invoked from the user-interface when a new data is attached to the terrain rendering module.
- *Render* : Called every time the scene needs to be rendered. View-frustum culling, LOD selection and actual drawing is done here.
- *Update**: Used to update different parameters such as screen error threshold, colormap, transformations, drawing style etc.

4 Data Layout

To make processing easier, we have resampled our terrain to be 8193x8193 of the form (2^n+1) X (2^n+1) resulting in 2^{n+1} quadrilaterals. Each quad has 2 triangles defined by its 4 corner vertices.

4.1 Creating the Quadtree

We use a quadtree data structure to spatially organize the terrain. The quadtree is generated at terrain load time and subsequently traversed for the update and draw functions. In a quadtree, each node has either zero or four children; nodes with zero children are leaf nodes also referred to as *Patches* in our nomenclature. All the nodes in the quadtree store information such as the bounding box and the patches additionally contain the actual geometry (vertices) with associated attributes such as normals, colors and indices. The advantage of organizing vertices in a patch is that we can issue one draw-primitive call for the entire block. This limits the calls made to the graphics subsystem to the number of patches that will be drawn. There is no restriction in the number of grid points within a patch and it is a matter of experimentation to find the optimal patch size for a given mesh. It should be noted however that very large patch sizes imply less LOD'ing (more rendering) while very small patchsizes can lead to over-LOD'ing (more processing). Figure 3 shows the quadtree structure and how it maps to the terrain. The patch size together with the terrain size decide the eventual depth of the quadtree.

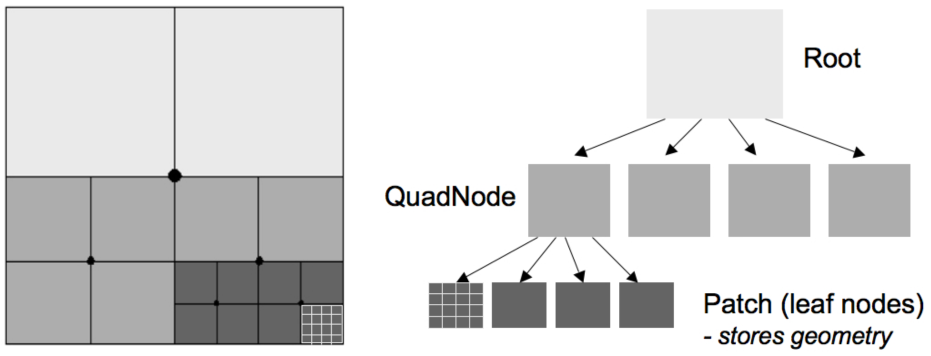


Figure 3: An example terrain and its associated quadtree. Patch size is 4.

4.2 Efficient Geometry Management

4.2.1 Vertex and Index Buffer Objects

Moving geometric data to the graphics hardware is one of the perennial difficulties in achieving good performance on modern accelerators. Existing features such as display lists and vertex arrays offer rudimentary memory management and involve expensive data transfers over the PCI bus. A recent OpenGL extension called vertex buffer object (VBO) [NVidia, 2003] is a powerful feature to store and access data in high-performance graphics memory. VBO's can contain vertex attributes, such as coordinates, normals, per vertex-colors as well as indices to access the vertices. This mechanism allows multiple chunks of memory (buffers) that will be available through identifiers. As with any display list or texture, we can bind such a buffer so that it becomes active. An interesting consequence of using these targets is the ability to switch between various index buffers while keeping the same vertex array buffer, enabling us to implement LOD. Figure 4 shows a simple example, with one mesh and 2 index tables to define 2 mipmap levels.

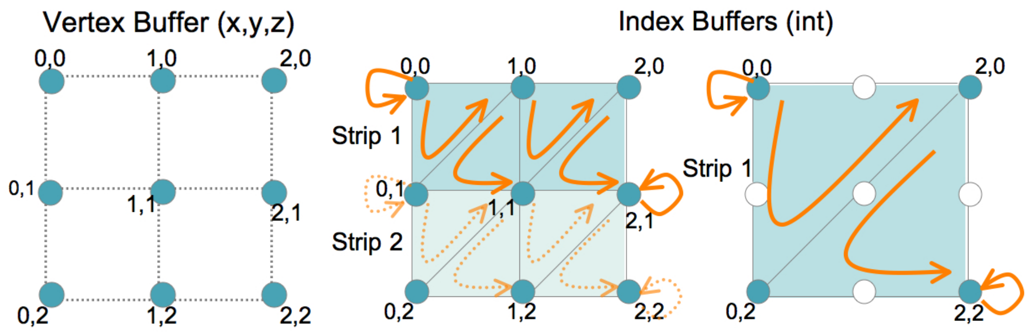


Figure 4: 3x3 Patch - a) Vertices; Indices for b) Level 0 and c) Level 1

4.2.2 Triangle Strip Generation

The mesh in each patch is converted to triangle strips in order to speed up rendering while simultaneously saving storage space. The vertex data (x,y,z) is stored for each patch (at full resolution) in a vertex buffer object and downloaded to the graphics card at load time. Taking this a step further, we also use indexed triangle strips that reuse the vertices between the mipmap levels, allowing the graphics card to perform efficient caching. The indices for each mipmap are stored in separate index buffer arrays and downloaded to the card at loadtime. By default, we have 5 mipmap levels. Unlike the vertex data, just one set of index buffers suffices for the whole terrain since the patch size is uniform throughout. Figures 4b) and c) show the triangle-strip generation process. Within each patch, the triangle strips are lined up side by side. To join the strips, we use redundant vertices and degenerate triangles[Evans et al., 1996]. This is achieved by specifying two extra vertices to transition from one row to the next. At the end of a row, the last vertex is used for a second time, and at the beginning of the next row, the first vertex is also specified twice. For a nxn patch, we require 2n + 2 indices to form each row. Since there are n-1 rows, the maximum number of indices for each level is (2n + 2)(n-1) = $2n^2$ - 2 . For the 3x3 patch in Figure 4, the level 0 mipmap has a total of 16 indices.

5 Data Culling

5.1 Visibility Culling

Once the data is laid out in the quadtree structure on the host and all the geometry and indices are cached to the graphics, a preliminary data reduction method like view-frustum culling [Assarsson & Möller, 2000] is used to discard patches not visible on-screen. This is done at run-time by traversing the quadtree and intersecting the bounding box of each quadnode with the 6 planes of the viewing frustum. If there is no intersection, we can safely discard the node and its children as not visible. When the bounding box is at least partially inside the view-frustum, this process is recursively repeated for the children. The visible leaf nodes are then marked for LOD'ing. Depending on the view direction, the rendering load is reduced by a factor of 2 to 3. However, the distant patches are still rendered at full-resolution.

5.2 LOD Selection

We are now left with the details of selecting the appropriate LOD for each visible patch. While this entire computation can be done dynamically, we do some pre-processing to limit the CPU involvement to just a few multiplies and comparisons at run-time.

The *world-space error* for a mipmap level is a measure of the *maximum* difference in height-values between the mipmap and its coarser approximation over the entire patch. The approximated height-value in a coarser level mipmap is bilinearly interpolated from the previous level mipmap. For a static terrain, this computation can be done at load-time. The *screen-space error* is the projection of the world-space error onto the screen and can be calculated using the standard projection equations in computer graphics.

$$screen\_space\_error = \frac{world\_space\_error}{distance\_to\_viewer} * \frac{viewport\_height}{2 * tan\frac{vertical\_field\_of\_view}{2}} \quad (1)$$

The consequence is that if the different levels of mipmaps do not cause visible change on screen we rather stick to a lower-level mipmap and save on triangles. A user-specified *screen-space error threshold* encodes the deviation that can be tolerated from the "ideal" image in terms of screen pixels. A threshold of 0 indicates no deviation whatsoever from the original terrain and forces the mesh to be rendered in brute force, useful for offline rendering such as movies and high-quality images. As the threshold increases, the switch to a coarser mipmap level occurs at a closer distance to the viewer. For a given screen error threshold, we can re-arrange equation 1 to calculate the minimum distance from the camera at which each mipmap level is used. At run-time, the CPU simply calculates the distance_to_viewer for each patch and compares that value with the precalculated distances to choose the appropriate mipmap level. The square of the distances are stored to avoid expensive square root operations. The minimum distance values for each level will of course, need to be recomputed when any of the camera-projection parameters such as viewport or field of view change.

The pre-processing steps outlined above are quite expensive and can take a few minutes when loading large terrains. Ideally, the user should be able to load an arbitrarily sized terrain and begin traversing it immediately. The results of the preprocessing can be saved with complete hierarchical information for subsequent faster loading.

6 Rendering and Interaction

After the LOD selection, the patches are depth-sorted to exploit the early-Z test, if available, and to reduce overdraw. At this point, rendering the terrain is as simple as a single call (glDrawElements) with a pointer to the GPU memory where the triangle strips are stored for each patch. If a colormap is attached, it is stored as a 1D texture and applied to the terrain using the GL automatic texture generation to map z-values to colors and transparency. This is computationally and memory efficient in comparison to applying the colors on a per-vertex basis. As the texture mapping occurs during the rasterization phase, the colors are applied to the filtered data values, thus preserving high-frequencies in the colormap, a feature useful for drawing isolines.

Apart from standard object transformations such as scaling, translation and rotation, other interaction functions include manipulating the screen space error threshold, updating the colormap, saving the configuration for faster loading in future, changing drawing styles such as wireframe, shaded, textured as well as some benchmarking and debugging options to identify bottlenecks.

7 Results

For our tests, we used a high-end Workstation with 2 Dual Opteron 2.6GHz processors, 16GB RAM and a NVidia Quadro 5500 card with 1GB memory.

Graphics Memory Footprint - Each vertex point in the terrain requires 3 floats or 12 Bytes for x,y and z position. Since, the vertices are cached at the highest resolution, this requires 8K*8K*12B = 768MB. The indices and the colormap occupy less than 1MB.

Graphics Performance - We use the perspective shown in Figure 5 as a baseline for our benchmarks. A brute-force method that renders the full terrain at highest resolution ran at 3 frames per second (fps). Adding view-frustum culling increased this to 10fps and further LOD'ing resulted in 15fps.

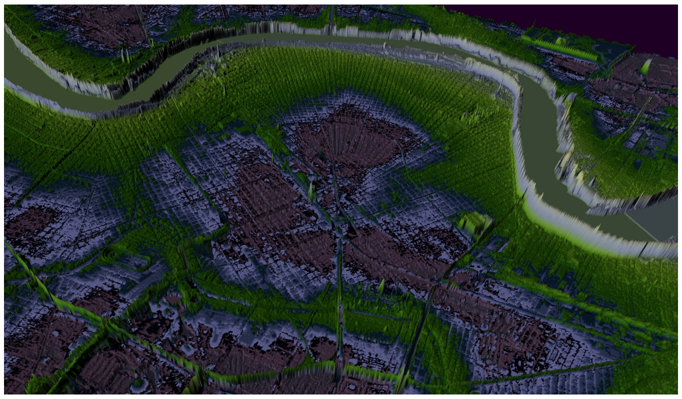


Figure 5: LOD Results with screen space error of 1 pixel and patch size 128

8 Conclusions and Future Work

We have described a method for faster rendering of terrain data using data reduction methods such as LOD and view frustum culling. Optimizations such as vertex buffering, indexed triangle strips and texture-based colormaps make effective use of the graphics hardware, freeing the CPU and system bus for other tasks. In future, we would like to experiment with storing the height-map within a vertex texture on-board and generating the vertices on the fly using a shader. Many applications will require visualization of terrain-data that will not fit into memory in its entirety. Some form of on-demand loading, prefetching or even compression methods will be required.

9 Acknowledgments

We would like to thank our colleagues for reviewing the paper, Dewitt Braud at LSU Coastal Studies and Shreekanth Balasubramaniam for help with data preparation and Wei He for the colormap selection.

References

[Assarsson & Möller, 2000] Assarsson, U. & Möller, T. (2000). Optimized view frustum culling algorithms for bounding boxes. *Journal of Graphics Tools: JGT*, 5(1), 9–22.

[de Boer, 2000] de Boer, W. (2000). Fast terrain rendering using geometrical mipmapping. `citeseer.ist.psu.edu/deboer00fast.html`.

[Duchaineau et al., 1997] Duchaineau, M. A., Wolinsky, M., Sigeti, D. E., Miller, M. C., Aldrich, C., & Mineev-Weinstein, M. B. (1997). ROAMing terrain: real-time optimally adapting meshes. In *IEEE Visualization* (pp. 81–88).

[Evans et al., 1996] Evans, F., Skiena, S. S., & Varshney, A. (1996). Optimizing triangle strips for fast rendering. In R. Yagel & G. M. Nielson (Eds.), *IEEE Visualization '96* (pp. 319–326).

[Lindstrom & Pascucci, 2002] Lindstrom, P. & Pascucci, V. (2002). Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3), 239–254.

[NVidia, 2003] NVidia (2003). White paper - using vertex buffer objects. `http://developer.nvidia.com/object/using_VBOs.html`.

[Stalling et al., 2005] Stalling, D., Westerhoff, M., & Hege, H.-C. (2005). Amira - an object oriented system for visual data analysis. In C. R. Johnson & C. D. Hansen (Eds.), *Visualization Handbook*: Academic Press.

[Venkataraman et al., 2006] Venkataraman, S., Benger, W., Long, A., Jeong, B., & Renambot, L. (2006). Visualizing hurricane katrina: large data management, rendering and display challenges. In *ACM GRAPHITE '06* (pp. 209–212). New York, NY, USA: ACM Press.

A Data Transfer Benchmark

Andrei Hutanu
[1]Center for Computation and Technology, Louisiana State University, USA
ahutanu@cct.lsu.edu

Abstract

This works analyzes a number of parameters that characterize data transfer operations in order to see how they influence data throughput. The motivation is given by an application that visualizes remote data using high-speed optical networks but the results can be applied for a wide range of data-intensive applications. The goal of this work is to gain an understanding of how network-related parameters that can be controlled by the user such as the transport protocol, number of data servers, number of threads need to be set in order to minimize data access time.

1 Introduction

HPC simulations and scientific experiments generate datasets with sizes that exceed the storage capability of the workstations used to visualize and analyze this data.

Also, the speed of local disks limits the interactiveness of a visualization application. Raw HDD read speeds are currently in the range of 20-150 Mbytes/second with typical speeds of around 35 Mbytes/sec for laptops and 70 Mbytes/sec for workstations. For comparison, laptops now typically have Gigabit Ethernet connections with transport speeds of around 950 Mbits/sec and 10 Gb Ethernet optical network adapters easily achieve speeds of 5-6 Gbits/sec and new adapters can go near the theoretical maximum of 10Gbps.

These high-speed networks open new perspectives on the data access problem and one idea is to utilize remote machines to store the data in their main memory and visualize the data from there. The idea of utilizing network

RAM for application speed-up is derived from the concept of virtual memory and its roots can be traced back to the 90's [Comer & Griffioen, 1990]. With this approach, the bottleneck is the data transfer speed and since the network can physically support data rates of up to 10Gbps, the problem is now making the application achieve such data rates.

In this context, this work tries to take a systematic look at the performance of various transport protocols applied for network RAM. Additionally, it compares the performance of single threaded versus multi-threaded communication and looks at the overhead of implementing transport protocols in user space (as opposed to have them implement in the kernel). It also analyzes the relation between the number of data servers and the data transfer performance.

2 Application

The work described here is based on existing work in remote data access across wide-area optical networks. Initial motivation of this project was given by the need of visualizing datasets generated on remote supercomputers by scientific simulations. These datasets can have sizes in excess of 100 Gbytes (for example a numerical relativity simulation working on a grid size of 400x400x400 running 500 timesteps will generate 120 Gbytes of data for a single variable - 4 bytes/data) and it is not desirable to transfer the entire data file locally for visualization. As a user is only interested in certain parts of the dataset at any given time, one alternative to transferring the entire file is to transfer only those sections of interest to the local machine. When the data is received, the visualization is updated and the user can move to another section of interest, thus interactively exploring the dataset [Prohaska et al., 2004]. Another possible way of reducing the amount of data transferred to the local machine is to perform a progressive visualization where low resolution versions of the data are transferred first and then higher resolution data (see Fig. 1). The visualization client needs to be able to update the visualization as the data arrives.

The implementation of the remote data access library uses a dual-channel approach, where the data selection request is transmitted over a SOAP channel and the data response is transmitted over a separate, binary data channel[Prohaska & Hutanu, 2005] (This is illustrated in Fig. 2). This allows the application to switch between different transfer mechanisms for the

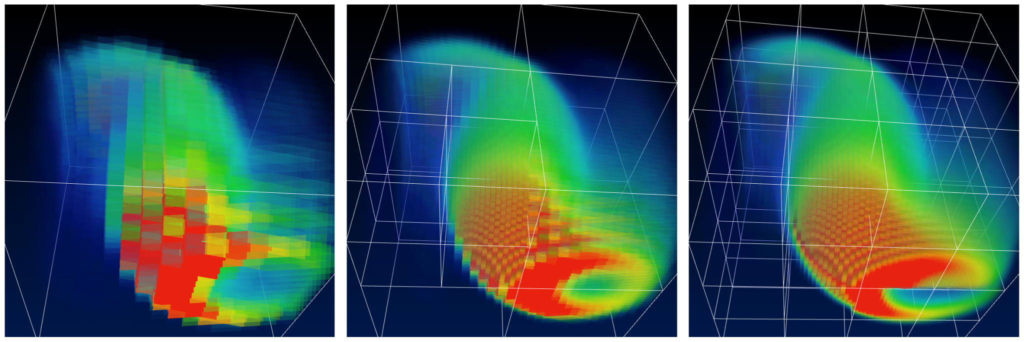


Figure 1: Progressive visualization

important binary data channel, a feature that is used here to compare the performance of various transport protocols.

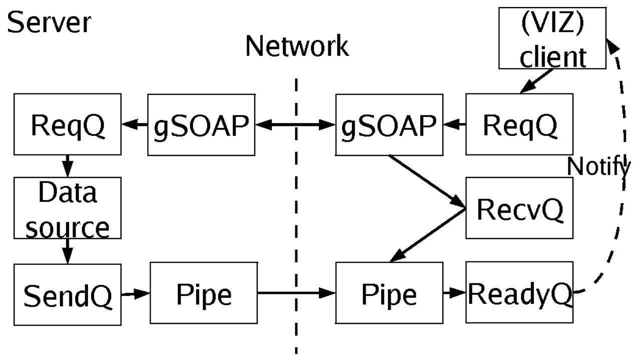


Figure 2: Implementation of client-server data access

With the advent of optical network connections, the network bottleneck has disappeared and network utilization can be improved by using remote main memory to store the data of interest. In addition to this, multiple remote resources (data servers) can be utilized in parallel to improve data transfer performance. This approach is illustrated in Fig. 3 - left.

The data object of interest is distributed across all data servers that are active. The work size of each data server should be proportional to the speed that can be delivered by that server. In this benchmark, the data object is equally distributed across all the nodes that are active as data servers as the transfer speed of all nodes to the client node should be equal. This is illustrated in Fig. 3 - right, with a mention that in the current benchmark

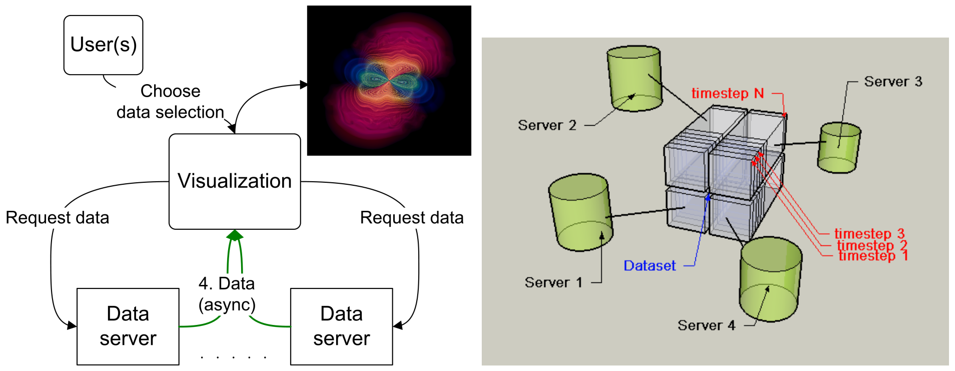


Figure 3: Distributed data server (left), Data distribution (right - illustration by Farid Harhad)

a large dataset containing a single timestep is utilized. The applications of this method are not limited to visualization but can be extended to any application requiring remote data access.

3 Benchmark

For this benchmark, a nine node cluster was utilized. Each node has two Dual Core AMD Opteron 64 bit processors and 8 GB of shared memory available to the processing cores. The cluster has two network interconnects, a gigabit Ethernet and a Myrinet interconnect. The network performance between two nodes measured with *iperf*[1] is 941 Mbps (TCP) & 957 Mbps (UDP) for Ethernet and 1.98 Gbps (TCP) & 1.91 Gbps (UDP) for Myrinet.

In the benchmark, one of the compute nodes was selected as a receiving client and a variable number (from one to six) of nodes were utilized as data servers. A dataset of a size varying from 16 Mbytes to 1 Gbyte was equally distributed between all data servers. For each configuration the average of 12 measurements is shown.

Much work has been done in the field of transport protocols, and most of the research on transport protocols has produced modifications of the TCP protocol. TCP and its variants are used on top of unreliable connectionless protocol such as IP and as most important features provide reliability and ordering for data transport. Some of the existing TCP variants are:

[1]iperf:http://dast.nlanr.net/Projects/Iperf/

HSTCP [Floyd, 2003], Scalable TCP [Kelly, 2003], BIC [Xu et al., 2004], H-TCP [Leith & Shorten, 2004], TCP Westwood and Fast TCP.

All of these TCP variants are available as patches for the Linux kernel and since kernel 2.6.7 they are an integrating part of it and can be enabled or disabled on the fly. However, switching TCP variants in the kernel on a shared resource is not desirable.

Fortunately, the UDT library provides a framework for user-space implementations of congestion-control algorithms [Gu & Grossman, 2005]. Congestion control, or the mechanism that responds to packet loss in the network (which usually happens when the data rate exceeds the network capacity) has the most significant impact on data transfer rates. UDT provides reference implementations for congestion control algorithms of some of the protocols described above including the following TCP variants: HighSpeed, Scalable, BiC, Westwood, Vegas [Brakmo et al., 1994], and Fast plus the native UDT (UDP-based Data Transfer Protocol). Everything else required by a reliable transport protocol (buffering, retransmission, loss lists) is implemented in the UDT library and is common for all supported congestion control algorithms. This flexibility is the reason why the UDT library is utilized here to compare the performance of the various protocols. This benchmark also compares the performance of the UDT library implementation with the kernel implementation of TCP (available through the sockets API).

Another interesting issue is the performance of a multi-threaded approach, where for each data server, a separate thread is started on the receiving end so the number of threads receiving data is equal to the number of connections. This is compared with an implementation that uses a single thread to retrieve data across all connections (using a variant of the select system call to determine which connection has data available).

The numbers quoted reflect the end-to-end data transfer performance achieved by the remote data access library described above. As such, the results are expected to show a slightly lower performance than the pure network data transport performance due to library overheads.

4 Results

A subset of transport protocol, network, message size, threading and number of data servers combinations was selected and the results are presented in the following.

Table 1 shows the results of the benchmarks that were run over Ethernet, using a 1 Gigabyte dataset, and a multi-threaded implementation for the following congestion control algorithms : UDT, TCP, BIC TCP, Scalable TCP, High-speed TCP, TCP Westwood, TCP Vegas and Fast TCP. All these protocols are implemented in the UDT library. Some of the measurements have failed (possibly because of the implementations of the congestion control algorithms in the UDT library) and some measurements were not performed for all combinations of data servers but a clear indication of the performance should given by the given measurements. The results show that standard TCP (also known as TCP Reno), BIC TCP, Scalable TCP and TCP Vegas perform best with a slight advantage for BIC TCP and Scalable TCP.

#	UDT	TCP	BIC	S-TCP	HS-TCP	TCP W	TCP V	F TCP
1	461	533	540	530	527	536	509	537
2	339	626	622	618	N.W	N.W.	610	464
3	425	-	-	-	-	-	-	-
4	520	-	-	-	-	-	-	-
5	630	**727**	**734**	**762**	N.W	511	744	349
6	620	**749**	**787**	**782**	N.W	655	756	335

Table 1: Throughput (Mbits/s) using various congestion control algorithms in the UDT library over Ethernet, 1Gbyte data messages, multi-threaded. First column shows number of data sources, first row shows the transport protocol. S-TCP=Scalable TCP, HS-TCP=High-speed TCP,TCP W=TCP Westwood, TCP V=TCP Vegas, F TCP=Fast TCP, N.W=measurement failed, – = Not measured.

Figure 4 shows the results of the measurements that used the Myrinet network. These measurements were carried out for three protocols : TCP - kernel implementation, single threaded and the UDT library implementation of TCP both single and multi-threaded. An interesting result captured here is the behavior of TCP when the number of data servers is increased. Contrary to all other results that show (with one exception) that increasing the number of data servers increases the throughput, here we see that the maximum throughput is reached for three data servers and then it steadily decreases. One conclusion is that the throughput does not increase indefinitely and at the point where the maximum is reached it starts to decrease as more data servers are added.

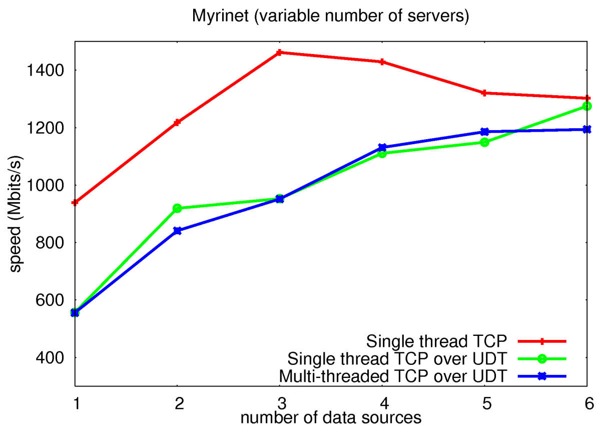


Figure 4: Throughput (Mbits/s) over Myrinet, 1Gbyte data messages

The final set of measurements used variable dataset sizes (256 Mbytes, 64 Mbytes and 16 Mbytes) to see how the data size may affect the transport performance. The results show that while decreasing with smaller dataset sizes, varying the message size does not affect the throughput performance significantly for either one of the Ethernet (716-748 Mbps for six servers) or Myrinet (1322-1344 Mbps for six data servers) networks. This indicates that the overhead of the remote data access for the networks considered here is minimal.

Figure 5 shows a comparison between Myrinet and Ethernet and the advantage is clearly with the former. Figure 6 shows a partial comparison of multi-threaded implementations versus single-threaded. In this case there is no clear advantage for any of the implementations. Figure7 shows the overhead of a UDP-based implementation. It is clear that kernel-based implementations are more efficient, however the most important disadvantage of kernel-based implementations is that they cannot be switched by the user.

5 Conclusions and future work

The performance of the remote data access library is up to 30 percent lower than the pure network transport performance and is due to the overhead

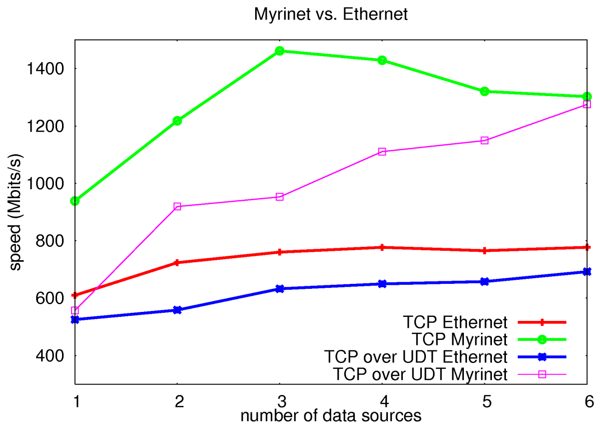


Figure 5: Myrinet vs Ethernet

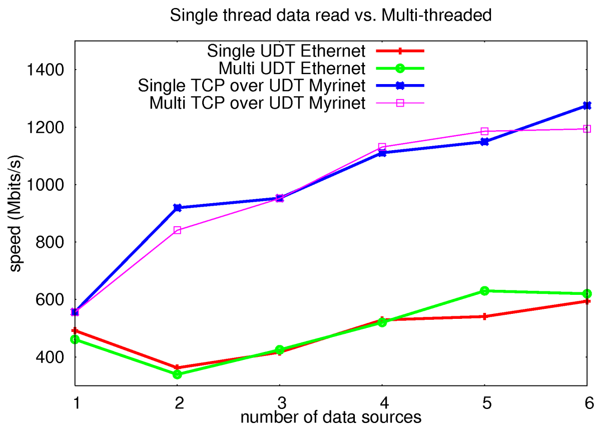


Figure 6: Single threaded vs. Multi-threaded

of data request distribution and threading in the library. This overhead is relatively high and needs to be reduced by optimizing the library code.

The results show that in terms of congestion control algorithms BIC TCP and Scalable TCP perform best on a low-latency network as that of our cluster. It would be interesting to see if the results are consistent with kernel implementations of these protocols. Further investigation is necessary for determining the usefulness of a multi-threaded implementation. Increasing the number of data servers can increase the performance with up to 50% but care must be taken in order not to exceed the optimal number of data

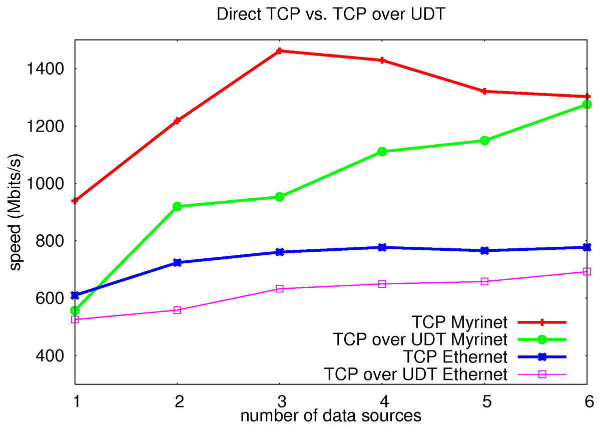


Figure 7: Overhead of UDP-based TCP implementation

sources. From an application perspective this means that certain data rates cannot be exceeded when using a single client. This means that in order to further increase data rates distributed visualization algorithms need to be applied.

The overhead of a UDP-based implementation is significant and tends to decrease as the speed is increased. This is an indication that if computationally intensive visualization algorithms are executed on the client, the data transfer rate could be decreased even further. Varying the data sizes does not affect the performance significantly. This is mostly because the overheads for intra-cluster communication are minimal. This implies that combining multiple remote operations into larger ones for purposes of optimization is not necessary.

The library used in the benchmark was part of the Amira code and is currently a stand-alone library. In future it should be (re)integrated in visualization applications.

References

[Brakmo et al., 1994] Brakmo, L. S., W.O'Malley, S., & Peterson, L. L. (1994). Tcp vegas: New techniques for congestion detection and avoidance. In *Proceedings of the SIGCOMM '94 Symposium* (pp. 24–35).

[Comer & Griffioen, 1990] Comer, D. & Griffioen, J. (1990). A new design for distributed systems: The remote memory model. In U. Association (Ed.), *Proceedings of the USENIX Summer 1990 Technical Conference* (pp. 127–136).

[Floyd, 2003] Floyd, S. (2003). Highspeed tcp for large congestion windows. ftp://ftp.rfc-editor.org/in-notes/rfc3649.txt.

[Gu & Grossman, 2005] Gu, Y. & Grossman, R. (2005). Supporting configurable congestion control in data transport services. In *Supercomputing 2005*.

[Kelly, 2003] Kelly, T. (2003). Scalable tcp: improving performance in highspeed wide area networks. *SIGCOMM Comput. Commun. Rev.*, 33(2), 83–91.

[Leith & Shorten, 2004] Leith, D. & Shorten, R. (2004). H-tcp: Tcp for high-speed and long-distance networks. In *Workshop on Protocols for Fast Long-Distance Networks*.

[Prohaska & Hutanu, 2005] Prohaska, S. & Hutanu, A. (2005). Remote data access for interactive visualization. In *13th Annual Mardi Gras Conference: Frontiers of Grid Applications and Technologies*.

[Prohaska et al., 2004] Prohaska, S., Hutanu, A., Kahler, R., & Hege, H.-C. (2004). Interactive exploration of large remote micro-ct scans. In *VIS '04: Proceedings of the conference on Visualization '04* (pp. 345–352). Washington, DC, USA: IEEE Computer Society.

[Xu et al., 2004] Xu, L., Harfoush, K., & Rhee, I. (2004). Binary increase congestion control for fast long distance networks. In *INFOCOM 2004*, volume 4 (pp. 2514–2524).

Views on Fusion

Alexander Kendl
Platform for Computer Science & Applied Computing,
University of Innsbruck, Austria
alexander.kendl@uibk.ac.at

Abstract

An overview on visualisation needs and strategies in fusion plasma physics and engineering for fusion energy research with tokamak and stellarator magnetic confinement configurations is presented.

1 Introduction

Currently two major new experiments for magnetical confinement of plasmas for fusion energy research are being built in Europe: the world's largest tokamak ITER in Cadarache, France, and the most advanced stellarator Wendelstein 7-X in Greifswald, Germany. In these experiments both the plasma with its confining magnetic field and also the coils and technical support structures feature complex toroidal shapes. Stellarators have a particulary intricate three-dimensional (helically twisted) geometry.

Visualisation needs arise on various levels of the experiment: At the design and engineering phase for the 3D coils and their periphery, structural force and stress simulations and virtual tests of the complicated assembly process have to be analysed. In the experimental phase, measured 3D time-dependent plasma parameters have to be tomographically reconstructed from various (non-invasive) diagnostics.

Theoretical plasma physicists work in 3D field-aligned curvilinear coordinates for simulation and visualisation of plasma stability, turbulence and transport; hereon will also be the main focus of this paper, as it covers recent work of the presenting author. And not least, the complex physics and technology is visualised for public outreach activities.

2 Magnetic confinement of fusion plasmas

Fusion research aims at utilising the energy source of the stars in power plants on earth: fusion of light nuclei, like hydrogen to helium, with release of a large amount of energy occurs, when the "fuel" particles are energetic enough to overcome their electromagnetic repulsion. The basic condition for achieving efficient fusion reactions is to confine the hydrogen particles at high enough temperature and density and for long times. Stars manage to burn protons to helium by the slow p-p reaction cycle at moderately high temperatures (several Million degrees) due to the strong confinement by their own gravity resulting in high densities and long confinement times. On the other hand, d-t reactions between the heavier isotopes deuterium and tritium are favoured for fusion research on earth because of their higher reactions rates, which have a maximum at several 100 Million degrees.

Matter at the high temperatures needed for fusion is in the plasma state, in which the deuterium and tritium atoms are fully ionised. In magnetic confinement fusion research it is used that charged particles (ions and electrons) are confined by a magnetic field on an orbit gyrating helically around the field line. Magnetic field lines are bent into a closed torus, and an additional poloidal component of the field line direction introduces a rotational transform, so that the field lines wind helically around the torus, which is necessary for plasma confinement and stability.

The poloidal magnetic field component can be either achieved by additional external helical field coils in ,,stellarator" configurations, or by the ,,tokamak" concept where a strong toroidal current in the plasma itself is induced. Tokamaks at present are the most advanced fusion configurations from both a technological perspective and from the development of a solid experimental and theoretical plasma physics basis over the last decades. The next major step in fusion research is the ,,International Thermonuclear Experimental Reactor" ITER, which has been commissioned in 2006 to be built near Cadarache in southern France until around 2015. ITER is designed to explore the physics basis of a fusion power plant with an intermediate plasma volume between present day experiments and an eventual commercial scale power plant [ITE, 2007a]. Stellarators allow for more flexibility in optimisation of the plasma shape as the magnetic field is primarily determined by the external coil currents, and flux surfaces already exist in vacuum. A large variety of stellarator configurations with different confinement properties and shapes have been developed and studied experimentally. A promising line

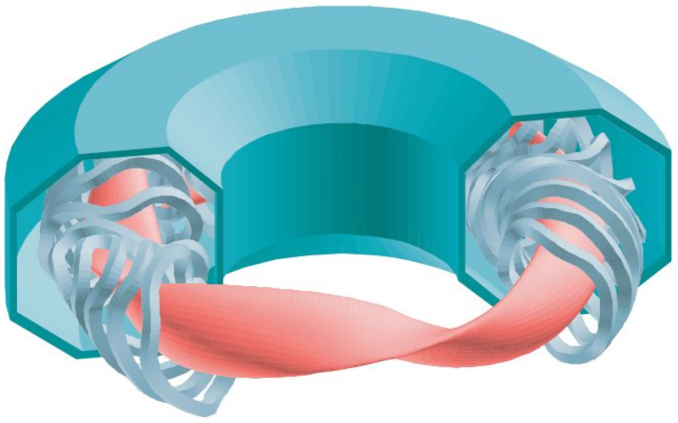


Figure 1: *Layout of components in the stellarator reactor concept HSR 4/18 [Beidler et al., 2001]: the plasma on closed magnetic flux surfaces is shown in red, magnetic field coils are in grey, and the surrounding cryostat chamber is shown in blue. [Figure: HSR]*

of research started with the partially optimized Helical Advanced Stellarator (Helias) experiment ,,Wendelstein 7-AS", that was in operation for 14 years until 2002, and was able to succesfully demonstrate the feasibility of the concept. As the next step (currently under construction in Greifswald, Germany), ,,Wendelstein 7-X" has been developed to test the reactor relevance of Helias configurations [Beidler et al., 2001].

3 Visual modelling of engineering challenges for next generation fusion experiments

A classical application field for visualisation in engineering is of course in the computer aided design (CAD) phase of any new technical system, including analysis of the stability of the designed system against forces and stresses. In magnetic confinement configurations, strong magnetic forces are acting between the field coils themselves and on structural materials. The force and stress distribution is intrinsically complex in the intricate 3D twisted structure of modular advanced stellarator coils (like those shown in Fig. 2).

For this reason, amongst others a new type of aid system for fusion reactor design, to which virtual reality (VR) visualization and sonification tech-

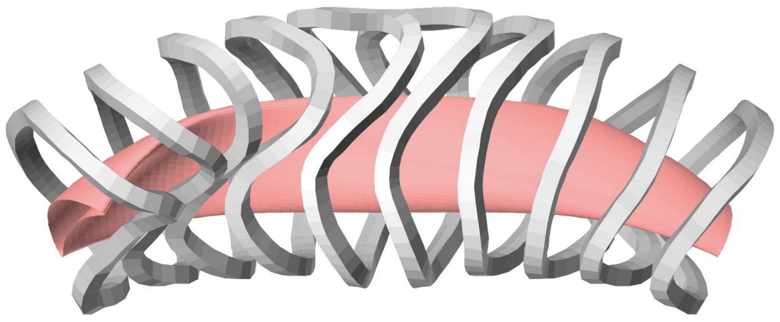


Figure 2: *One period of the coil system of the stellarator HSR 4/18 [Beidler et al., 2001]: Winding pack plus casing and magnetic surfaces. The intricate mounting procedure requires optimisation by means of visual simulation. During operation, the magnetic fields exert strong stresses on the coils and support structures, where overload maxima have to be eliminated beforehand also by computational and visualisation analysis.*

niques are applied, has been developed specifically for stellarator experiments [Mizuguchi et al., 2006]. This system provides with an intuitive interaction environment in the VR space between the observer and the designed objects constructed by the conventional 3D computer-aided design (CAD) system. The design aid tool has recently been first applied and evaluated for a stellarator-type fusion reactor design on the virtual reality system CompleXcope [Tamura et al., 2001] of the National Institute for Fusion Science, Japan.

In addition to such classical engineeering tasks, the mounting procedure of the (quite heavy) twisted stellarator coils upon the modules of the also trickily wound vacuum vessel is a nontrivial endeavour. Therefore, the whole mounting process for the Wendelstein 7-X coils has been simulated, visualised and finally optimised in detail before actual construction. A movie showing this process for one of the coils can be found on the Press and Media web site of the Max-Planck-Institute for Plasma Physics [ipp, 2007].

In burning plasma like in ITER the stray neutron distribution of fusion products that escape the vacuum vessel into the experimental hall during running experiments is also of considerable interest for safety and environmental considerations. Also in this case, visualisation of simulation results is a pertinent task to be solved during the engineering design phase.

In such neutronic operations with deuterium-tritium fuel remote handling of in-vessel components is necessary. The correct working of the handling procedures inside the toroidal vessel chamber is also extensively investigated by means of simulation and visualisation. A number of illustrative movies from these simulations can be found in the ,,Multimedia" section of the ITER organisation website [ITE, 2007a].

4 Plasma diagnostics: a look at the invisible

During operation of fusion experiments a large amount of data from a number of diagnostics and measurements has to be handled, stored and analysed. Little information however can be obtained by purely optical observation: the hot plasma core with hundreds of millions degrees temperature has most of its thermal emission not in the visible spectrum, but rather in the far UV. Material probes may also not be used for measurements on the plasma, so that a range of non-invasive fusion plasma diagnostic tools has been developed. Techniques include e.g. analysis of reflection or transmission properties of sets of electromagnetic wave or particle beams, which often require tomographic reconstruction to obtain real-space visualisation. A number of such diagnostics is e.g. described by Wesson [Wesson, 2004].

A popular set of software tools for data acquisition and storage and a methodology for management of complex scientific data is the package MDSplus [MDS, 2007]. It includes x-windows and java tools for viewing data or for modifying or viewing the underlying structures. Developed jointly by MIT, the CNR Padua and LANL, MDSplus is the most widely used system for data management in the magnetic fusion energy program. A range of coordinated computation and visualisation activities has been growing during recent years in view of ITER in the US, Europe and Japan.

5 Computational plasma dynamics

Numerical simulation of fusion plasma dynamics, turbulence and structure formation is usually performed in curvilinear coordinate systems that are aligned to the magnetic field lines, which exploits symmetries that considerably reduce the computational demands. As input into dynamical simulations at first the detailed geometry of the magnetic surfaces and parameters describing the equilibrium plasma have to be computed, usually based on experimental data. This equilibrium pre-processing delivers the magnetic surface shape and the field strength in basic Cartesian coordinates in terms of specific field aligned coordinates. From these, the metric and further field properties and gradients are obtained, which enter into many terms and differentiation operators of the dynamical equations.

Some insight into the expected dynamical behaviour of the plasma, like specific properties of instabilities and turbulent transport, can already be gained by analysing details of the metric and the field structure [Kendl, 2006]. A first task for visualisation thus appears in an insightful presentation of the 3D toroidal geometric quantities and field properties.

A typical example for a toroidal coordinate system (u^1, u^2, u^3) identifies $u^1 = \rho$ as a generalised coordinate of the minor torus radius, $u^2 = \theta$ as a generalised poloidal angle and $u^3 = \zeta$ as a toroidal angle. The different dynamical character of plasma dynamics parallel and perpendicular to a magnetic field line with $k_{||} \ll k_{\perp}$ motivate the use of the length along a field line on closed flux surfaces as one of the curvilinear coordinates instead of employing a strictly toroidal coordinate system. The Cartesian derivative operators at a position vector $\mathbf{R}$ thus are expressed in terms of an adapted set of curvilinear coordinates $u^i(\mathbf{R})$. In a contravariant basis the vector differentiation operator then becomes $\vec{\nabla} = \mathbf{e}^i(\partial/\partial u^i)$. For example, the Laplacian acting on a scalar f in general curvilinear coordinates can, after application of some vector algebra, be written as $\nabla^2 f = (1/J)\partial_{u^j}(J\ g^{ij}\partial_{u^i})$, including the Jacobian J and metric elements $g^{ij} = \mathbf{e}^i\mathbf{e}^j$.

In the usual labeling for flux coordinates (ρ, θ, ζ), the radial coordinate ρ as a flux-surface label can be identified with the plasma volume V enclosed by contours of constant pressure, or with the poloidal (or toroidal) magnetic flux $2\pi\Psi$ (or $2\pi\chi$) associated with this volume. The generalised ,,angles" θ and ζ are doubly periodic on the torus surface with $\theta = \theta + 2\pi m$ and $\zeta = \zeta + 2\pi n$ for integers m, n. The rotational transform describes the number of windings of the field line on a given flux surface around the torus as the change in poloidal angle $\Delta\theta \equiv \iota$ for a complete toroidal circuit of the toroidal angle $\Delta\zeta = 2\pi$: in straight-field line (SFL) coordinates the rotational transform is identical to $\iota(\rho) = \partial\theta/\partial\zeta$. Construction of a useful set of SFL coordinates involves solution of magnetic differential equations under particluar constraints.

Magnetic shear describes the dependence of field line direction on the coordinate ρ across flux surfaces. Global shear $s = (\rho/q)(\partial q/\partial\rho)$ is defined as the variation of the inverse rotational transform $q = (1/\iota)$ with minor radial coordinate ρ. It describes, how far the initially neighbouring Poincaré intersection points of two field lines on adjacent flux surfaces are separated poloidally after mapping both once around toroidally. The relative distance between two neighbouring flux surfaces defined by the poloidal flux $2\pi\Psi$ is measured by the metric $\nabla\Psi \cdot \nabla\Psi = g^{\Psi\Psi}$, whereas the poloidal distance is described by the field line label $\xi = \theta - \iota\zeta$ as $\nabla\Psi \cdot \nabla\xi = g^{\Psi\xi}$ in a toroidal flux coordinate system (Ψ, θ, ζ). This allows for a more general definition of magnetic field line shear as the change along field lines of the ratio $\Lambda = g^{\Psi\xi}/g^{\Psi\Psi}$ of these two quantities: $S = -\mathbf{B} \cdot \nabla\Lambda$. Whereas s is a constant on a flux surface, the new definition S now allows for local variations of the magnetic field line shear both on and across flux surfaces. Global shear may then be defined as the flux surface average $s = \langle S \rangle_\Psi$ of local shear.

The magnetic field direction vector $\mathbf{b} \equiv \mathbf{B}/|\mathbf{B}| = \mathbf{dR/dl}$ can be defined as the change of the position vector $\mathbf{R}$ with arc length l on the field line curve. The change of the magnetic field unit vector with arc length, $d\mathbf{b}/dl$ is perpendicular to the field vector $\mathbf{b}$. The magnetic field line curvature is defined by $\kappa \equiv (d\mathbf{b}/dl)|_{\text{along } \mathbf{b}} \equiv \mathbf{b} \cdot \vec{\nabla}\mathbf{b} \approx \vec{\nabla}_\perp \ln B$ as the change of the tangent unit vector along the field line. The field line curvature is split into normal and tangent components with respect to the flux surface, which respectively act as spatially dependent prefactors for normal and tangent differentiation operators on dynamical plasma quantities. The (non-unit) normal vector to the surface of constant poloidal flux $2\pi\Psi$ is defined by $\mathbf{n} \equiv \vec{\nabla}\Psi$, the geodesic vector as $\mathbf{g} \equiv (\mathbf{B} \times \vec{\nabla}\Psi)/(\vec{\nabla}\Psi \cdot \vec{\nabla}\chi)$.

The equilibrium metric and field quantities as input for the dynamical equations are static in time, and any standard 3D visualisation tool can in principle be used to analyse their structure. Interactive programming software like IDL or GnuPlot, or toolkits like Amira, are routinely applied. In Fig. 3 (a) the local magnetic shear S and the field strength $|B|$, which were obtained by the procedures outlined above for later use in a plasma turbulence code, are shown on a magnetic surface of Wendelstein 7-X.

Magnetic shear in general has a damping influence on tokamak edge turbulence, whereas geodesic curvature acting through κ_g upon the axisymmetric component maintains the coupling for a loss channel from zonal flow energy to turbulent vortices. Both mechanisms help to reduce turbulent transport. Normal curvature κ_n on the other hand strengthens primarily the interchange forcing of the turbulence.

Many computations of spatio-temporal fusion plasma turbulence with fluid or gyrofluid codes represent the toroidal geometry by a globally consis-

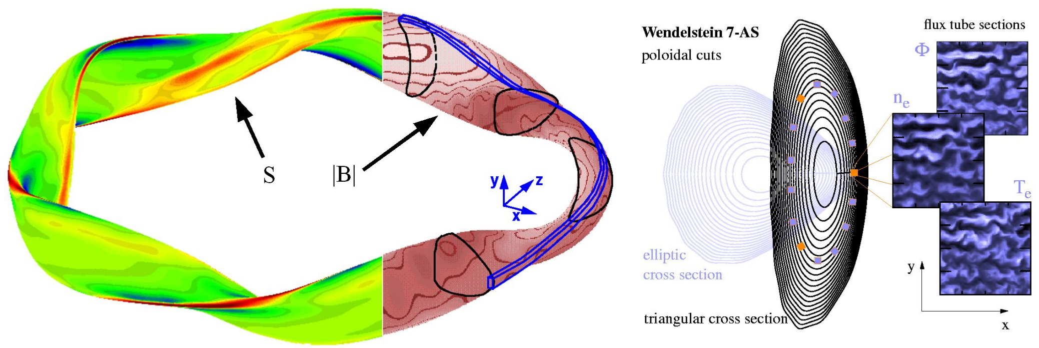


Figure 3: *(a)* **Left:** *Metric quantities on a magnetic equilibrium surface of stellarator Wendelstein 7-X. Left: local magnetic shear $S(\rho, \theta, \zeta)$. Right: magnetic field strength $|B|(\rho, \theta, \zeta)$. The black contours show poloidal cross-sections of various shapes. A typical flux tube computational domain aligned to a magnetic field line for turbulence computation is overlaid (in blue) on the other side [Kendl, 2006]. (b)*

Right: *2D poloidal cuts through the simulation flux-tube domain (here in Wendelstein 7-AS) showing characteristic turbulent quantities [Jenko & Kendl, 2002]. Usually visual analysis is done just in flux-tube space, but novel simulations of multi-scale interaction and structure formation necessitates the use of more sophisticated 4D visualisation tools, which require back-transformation to real space geometry.*

tent flux tube with locally rectangular coordinates (x, y, z). All background parameters (thermal gradients as well as the geometry) are fixed over the poloidal (x, y) region. The computational (x, y) domain is usually set to 64×256 nodes in units of the drift scale $\rho_s = (c/eB)\sqrt{T_e M_i}$ for (x, y) and 16-64 nodes in one field line connection length $(2\pi q R)$ in $-\pi < z < \pi$. Boundary conditions are usually periodic in y and Dirichlet in x. For the typical tokamak edge parameters, a box size of $64\rho_s$ x $256\rho_s$ corresponds to a rectangle of approximately 4 x 16 cm in real space. In comparison, the minor plasma radius of ASDEX Upgrade is $a_0 = 50$ cm, and the poloidal circumference $L_{pol} = 314$ cm.

Although such a flux-tube domain corresponds only to a partial section of a toroidal plasma it well represents the statistical properties of the complete system. This is ensured by choosing the domain size large enough to provide a decorrelation of vortex structures over the radial and perpendicular domain lengths. Globally consistent boundary conditions in y and z guarantee that perpendicular mode numbers are not cut off in the spectrum by the domain size but rather thinned out, so that flows covering the whole flux surface are well represented in a flux tube. The number of grid nodes in parallel direction z is chosen large enough to smoothly map toroidicity and further shaping effects into the geometric factors.

Basic visualisation and some physical insight into the results of turbulence simulations can already be obtained by plotting poloidal (x, y) cross-sections of the fluctuating quantities (at specific times or time animated), or by means of time or space averaged analysis of statistical properties. In Fig. 3 (b) parts of such poloidal cross-sections in simulations of ETG turbulence in the geometry of stellarator Wendelstein 7-AS are shown [Jenko & Kendl, 2002].

However, in some situations multi-scale structure formation processes occur, which invite more global 4D visualisation of the simulation data in order to guide the statistical data analysis procedures. An interesting and relevant example of such a situation is the generation of large scale global flows and their breakdown by meso-scale instabilities into small-scale turbulence. Such ,,ELM" events occur on largely different time scales nonlinearly connected across an order of 3 magnitudes. The possibility to simulate this events with the high-resolution turbulence codes only occured very recently (due to new models and computational capabilities), and both simulation and visualisation have just started [Kendl et al., 2006]. Here the need for 4D time-dependent visualisation in the global 3D toroidal geometry of a whole tokamak or stellarator configuration arises. Therefore the flux-tube (or annular field-aligned) data from computational coordinate systems has to be re-transformed to a Cartesian system by the reverse procedure than the one outlined above. No visualisation toolkit is known to the author which is in the moment able to handle such transformations. Thus, a post-processing of the data is currently performed in standard programming or interactive data languages before re-import of the simulation results into visualisation codes. We currently test Amira for use on the last stage.

6 Public outreach: fly through ITER

Public outreach activities need to clarify the complex technical and scientific issues in fusion research and plasma physics. This is of particular importance, as sometimes fusion is wrongly thought of as just an other kind of nuclear fission, and may thus be ill-considered in the public opinion.

The 3D stereoscopic movie ,,Starmakers", to be seen with passive polarized glasses, brings the spectator on a virtual visit of a fusion reactor based on ITER. This movie, totally made by computer, is a 3-D ,,real virtuality". It has been produced by the Centre de Recherches en Physique des plasmas, Ecole Polytechnique Federale de Lausanne, with a financial support of the European Commission, Direction de la Recherche. The images have been computerized by Digital Studio SA (Paris) based on the ITER Catia Cad integration design (Home-Team Garching). The ,,Starmakers" movie has been presented at the Hannover Universal Exhibition in 2000, where potentially 250'000 to 350'000 people could see it in 3D. It has been selected at the "2000 Beijing International Scientific Film Festival", November 2000. It did receive the "Grand Prix" at the FIFEL, in March 2001 (Festival International du Film sur l'Energie de Lausanne-Switzerland). Recently, it has also been presented in Innsbruck during the ,,FUSION EXPO" show in 2006. A plain 2D version can be downloaded from the internet via the ITER web site [ITE, 2007b].

References

[ITE, 2007a] (2007a). Available from: `http://www.iter.org`.

[ipp, 2007] (2007). Available from: `http://www.ipp.mpg.de/ippcms/de/presse/pi/05\_05\_pi.html`.

[MDS, 2007] (2007). Available from: `http://www.mdsplus.org`.

[ITE, 2007b] (2007b). Available from: `http://www.iter.org/Movies/starmakers.htm`.

[Beidler et al., 2001] Beidler, C., Harmeyer, E., Herrnegger, F., Igitkhanov, Y., Kendl, A., & et al. (2001). The helias reactor hsr4/18. *Nuclear Fusion*, 41, 1759.

[Jenko & Kendl, 2002] Jenko, F. & Kendl, A. (2002). Stellarator turbulence at electron gyroradius scale. *New Journal of Physics*, 4, 35.

[Kendl, 2006] Kendl, A. (2006). Plasma turbulence in complex magnetic field structures.

[Kendl et al., 2006] Kendl, A., Konzett, S., & Scott, B. (2006). 2d modelling and 3d simulation of elms with turbulence codes.

[Mizuguchi et al., 2006] Mizuguchi, N., Tamura, Y., Imagawa, S., Sagara, A., & Hayashi, T. (2006). Development of reactor design aid tool using virtual reality technology. *Fusion Engineering and Design*, 81, 2755–2759.

[Tamura et al., 2001] Tamura, Y., Kageyama, A., Sato, T., Fujiwara, S., & Nakamura, H. (2001). Virtual reality system to visualize and auralize numerical simulation data. *Comp. Phys. Comm.*, 142, 227230.

[Wesson, 2004] Wesson, J. (2004). *Tokamaks.* Oxford University Press.

(Dates of web links: 24.4.07)

A Visualization Toolkit for Lattice Quantum Chromodynamics

Massimo Di Pierro
School of Computer Science, Telecommunications and Information Systems
DePaul University, 243 S Wabash, Chicago, Illinois, USA
mdipierro@cs.depaul.edu

Abstract

Lattice QCD is an algorithmic formulation of QCD, the mathematical model that describes how quarks bind together to form composite particles such as proton and neutron. It has been successful in predicting properties of many newly discovered particles, including their mass and decay time. Unfortunately, lattice QCD is very computationally expensive and comprises of sophisticated algorithmic manipulations of large data-structures whose interpretation is purely statistical. In this paper we provide an overview of both lattice QCD and our work to develop a visualization toolkit to extract information from those data-structures. Our toolkit consists of a set of parallel algorithms for projecting the lattice QCD data structures into 3D scalar fields (for example the topological charge of the vacuum, the energy density, the wave function of the quarks, etc.) and uses VTK for the proper visualization.

1 Introduction

Quantum Chromodynamics [Marciano & Pagels, 1978] (QCD) is the mathematical model that best describes interactions among quarks, the basic constituents of most of ordinary matter. Lattice QCD is a formulation of QCD in terms of discretized space and time (lattice) that is suitable for numerical computation. Lattice QCD has been successful at explaining and predicting properties of composite particles such as the mass and lifetime of protons, neutrons, and many other particles produced by modern particle accelerators such as the Tevatron [FERMILAB-Pub-01/197,] at Fermilab, LEP and LHC at CERN, and Slac at Stanford.

For a compact introduction to lattice QCD see [Pierro, 2006] and references therein.

Lattice QCD algorithms comprise of massive parallel Monte Carlo simulations. Modern state of the art computations are performed on commercial supercomputers (such as Blue Gene [Bhanot et al., 2005] and the Earth Simulator), clusters of workstations (there are nearly 1000 dedicated nodes at Fermilab [Holmgren, 2005] and Jefferson Laboratory), and dedicated machines (Ape [apeNEXT Collaboration, 2003] in Rome, QCDOC [Gottilieb, 2006] at Columbia University and Brookhaven National Laboratory).

In many Lattice QCD computations, the only published output consists of one number with two or three significative digits. At the same time, hundreds of terabytes of data are generated in the intermediate steps of the computation and are not usually looked at because their interpretation is purely statistical: they are random points in a Monte Carlo ensemble.

The goal of our project is twofold: identifying a set of projection operators that would map this data into 3D scalar fields of physical significance for the purpose of extracting information in a visual format; incorporating these operators into a visualization toolkit that will interface with existing Lattice QCD software libraries such as MILC [Gottlieb, 2002], FermiQCD [Pierro, 2002] and SciDAC [Brower, 2006].

In the next section we will give a brief description of Lattice QCD from a computational point of view and we will discuss examples of projection operators of physical interest. In the third section we will present the high level design of our toolkit. Finally we will draw conclusions, show some images produced by our system and discuss the current status of the project.

2 Theoretical Foundations

The ingredients of any Lattice QCD computation are the following:

- In nature, there are 6 *flavors* of *Quarks*. They are the basic constituents of protons and neutrons, as well as of other composite particles that can be produced in particle accelerators and are occasionally produced naturally by cosmic rays. Each quark flavor can best be described as a complex field $q_{x,\alpha,i}$. The index $x = (\vec{x}, t)$ labels a point on the hypercubic lattice that corresponds to a position in space $\vec{x}$ and time t. At point x in space-time, a quark exhibits a number of local degrees of freedom that are parameterized by the indices $\alpha = 0, 1, 2, 3$ and $i = 0, 1, 2$. α is referred to as spin index and i as color index. $|q_{x,\alpha,i}|^2$ is the probability of finding the quark at a given position $\vec{x}$ at time t, in a spin state α and color state i.

- Quarks interact with each other by exchanging *gauge bosons*, also known as *gluons*. Gauge bosons can be best described as a complex field $U_{x,\mu,i,j}$. Similarly to the case of quarks, the index x labels a point in space-time, while the indices μ, i and j parameterize the local degrees of freedom. If we define $P_{x,\mu,\nu} = U_{x,\mu}U_{x+\mu,\nu}U^\dagger_{x+\nu,\mu}U^\dagger_{x,\nu}$ and $U_{x,-\mu} = U^\dagger_{x-\mu,\mu}$ then[1]

$$F^a_{x,\mu,\nu} = \frac{1}{8} Re\ Tr[\lambda^a(P_{x,\mu,\nu} + P_{x,-\mu,\nu} + P_{x,-\mu,\nu} + P_{x,-\mu,-\nu} \tag{1}$$

$$- P^\dagger_{x,\mu,\nu} - P^\dagger_{x,-\mu,\nu} - P^\dagger_{x,-\mu,\nu} - P^\dagger_{x,-\mu,-\nu})] \tag{2}$$

  is the chromo-electro-magetic tensor associated to gluons of type a. For each gluon type, $E_{x,i} = F_{x,0,i}$ is the chromo-electric field and $B_{x,k} = \sum_{i,j} \epsilon_{i,j,k} F_{x,i,j}$ is the chromo-magnetic field.

- QCD is a *Quantum Field Theory*. This means that there is no deterministic time-evolution for the above fields. In fact, the only meaningful physical observables in any Quantum Field Theory are the *correlations* between the degrees of freedom. Lattice QCD provides prescription rules on how to measure physical observables (for example the mass of a proton) using correlations among field variables. These correlations are computed numerically by averaging the corresponding operator over multiple *field configurations*, also known as *paths* or *histories*. Each configuration represents a possible evolution in time of a small portion of space of about (10^{-14} meters)^3 for about 10^{-22} seconds.

- Field configurations are generated via a Markov Chain Monte Carlo (MCMC) algorithm using a transition probability that encodes the physical laws of QCD. The "Quantum" aspect of the theory is represented by the stochastic field fluctuations present in the gauge configurations and averaged over.

[1]here λ^a is any of the 8 generators of the $SU(3)$ group

- For practical purposes any Lattice QCD computation is divided into three main steps: 1) gauge field configurations are generated using the MCMC; 2) for each gauge configuration one places the quarks in a certain state and let them evolve according to the Dirac equation in the background gauge field (the solution of the Dirac equation on a given gauge configuration is called a *quark propagator*); 3) the indices of a gauge configuration and its corresponding quark propagators are contracted together to compute the operator corresponding to a given physical observable, which is then averaged over all the gauge configurations.

- A typical gauge configuration U has a size of 96 points in time and 64^3 points in space, corresponding to 7 gigabytes of data. A typical quark propagator q for a single source on the above gauge configuration has a size of 2.5 gigabytes. Most Lattice QCD computations involve about 1000 gauge configurations and a minimum of 12 quark sources each, thus generating $10 \div 100$ terabytes of data. Usually gauge configurations and quark propagators are stored and are reused for multiple observables in a semi-industrial fashion. Typical computations require 1-10 million hours of computing time for a Pentium 4 @ 3GHz equivalent CPU.

- One complication consists in the fact that some of the degrees of freedom in the gauge field U are redundant but cannot be eliminated in the computation. In fact $U_{x,\mu,i,j}$ must be unitary matrices in the indices i, j and only operators invariant under the simultaneous transformations $U_{x,\mu,i,j} \to \sum_{m,n} \Lambda_{x,i,m} \Lambda^*_{x+\mu,j,n} U_{x,\mu,m,n}$ and $q_{x,\alpha,i} \to \sum_m \Lambda_{x,i,m} q_{x,\alpha,m}$ have a physical significance[2]. The above symmetry is called *gauge invariance* and it puts a major constraint on what can be visualized. The gauge invariance symmetry represents the principle of local indistiguishibility among quarks of different colors (they are 3 but one cannot tell them apart). This symmetry, motivated by experiments, poses a strong contraint on the model and it almost completely determines the inteaction term between quarks and gluons in the Dirac equation for QCD.

- The only input of a Lattice QCD computation are parameters that

[2]Here $\Lambda_{x,i,j}$ is an arbitrary field of unitary matrices in the indices i, j

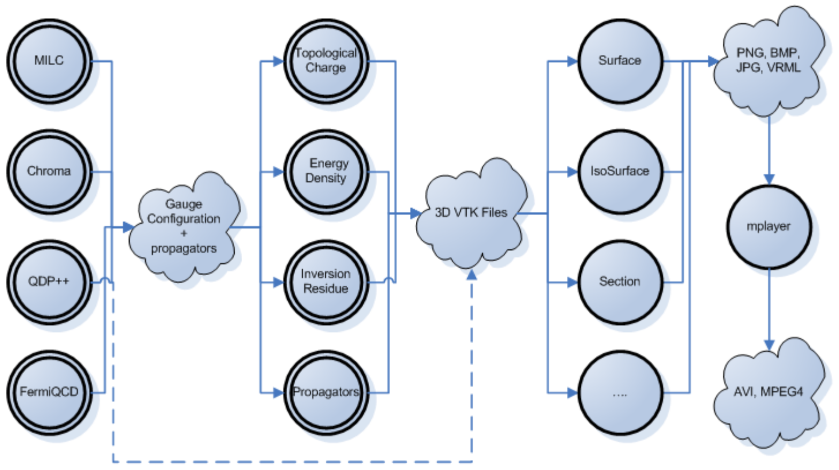


Figure 1: The figure shows the data flow in our visualization toolkit. The double circles represent parallel components including Lattice QCD algorithms and projection plug-ins. The single circle represent Mayavi visualization plug-ins.

loosely correspond to the masses of the quarks and to the lattice discretization scale. Physical observables have a dependence on the discretization scale that is known as *running*. For a sufficiently small discretization scale, the effect of the running can be corrected for and the results of the computation, in physical units, can be made independent on it. Physical observables also depend on the quark masses and, in fact, these are currently determined by a fine tuning the input parameters in Lattice QCD computations.

There are multiple reasons why extracting visual information from Lattice QCD data is important and here we list some of them:

- The gauge configurations have a non-trivial topological structure that can be isolated by removing short term fluctuations, a process called *cooling*. It is known that the total topological charge changes very slowly under MCMC steps, but it is not know how the local charge evolves.

- Some observables are themselves extended objects (for example the wave function of a quark inside a hadron or the energy density in presence of a quark-antiquark couple) but so far only 1D or 2D sections are usually visualized.

- The algorithms used for the MCMC and to invert the Dirac operator are both iterative. Therefore it is interesting to visualize how information propagates, step by step, across the lattice in order to understand the local convergence of these algorithms.

- Because of the size of the data structures involved, Lattice QCD algorithms are implemented as tightly-coupled parallel programs written in C/C++. Visualization can help monitor the communication patterns and identify bugs and network problems.

- Lattice QCD requires knowledge of multiple disciplines and therefore it has a very steep learning curve. The visualization of actual computations can serve a didactic purpose thus fostering a better intuitive understanding of QCD and pushing scientific progress.

3 Implementation

The main architecture of our system, fig. 1, is comprised of two main parts: a set of projection operators implemented in C/C++ as parallel MPI programs and a graphical interface based on the Enthough Workbench [wor, 2007] and Mayavi 2.0 [Ramachandran, 2001], which implements a Python interface to VTK [Schroeder et al., 2000]. We will refer to the former as a projection plug-in as opposed to the visualization plug-ins provided by Mayavi.

An example of a projection plug-in is a parallel algorithm that reads gauge configurations, cools it, computes the topological charge in 4 dimensions, takes a time slice and saves it as 3D scalar field in the VTK file format.

An example of a visualization plug-in is an algorithm that reads the above VTK file and generates iso-surfaces.

The Workbench, fig. 2, provides a GUI for the entire system and allows users to interactively manipulate the VTK files: display, rotate, edit, save them in some other standard graphical formats, including JPG, PNG, and VRML.

Mayavi is a Python interface to VTK and allows scripting of the above operations. A typical script would loop over a large set of fields, process each

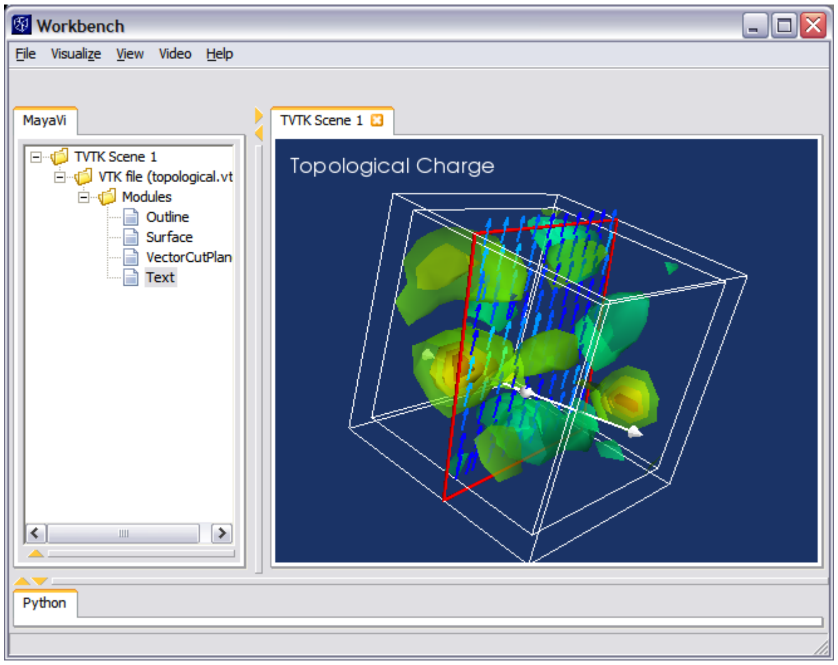


Figure 2: A screenshot of the Enthought Workbench showing the topological charge for a time-slice of a 4D gauge configuration.

of them using the same plug-ins and produce the individual frames as an animation.

Independently on the set of plug-ins used we identified three general recurrent patterns:

- Given one configuration, project and visualize the different time-slices.
- Given one set of configurations and one time-slice t, project and visualize the same time slice for each configuration.
- Given a set of configurations, for each time-slice, project the time-slice on each configuration, average over all configurations and visualize the average time-slice.

In order to remove visual unpleasant effects of high-frequency quantum fluctuations, when necessary, we adopted the following gauge-invariant smooth-

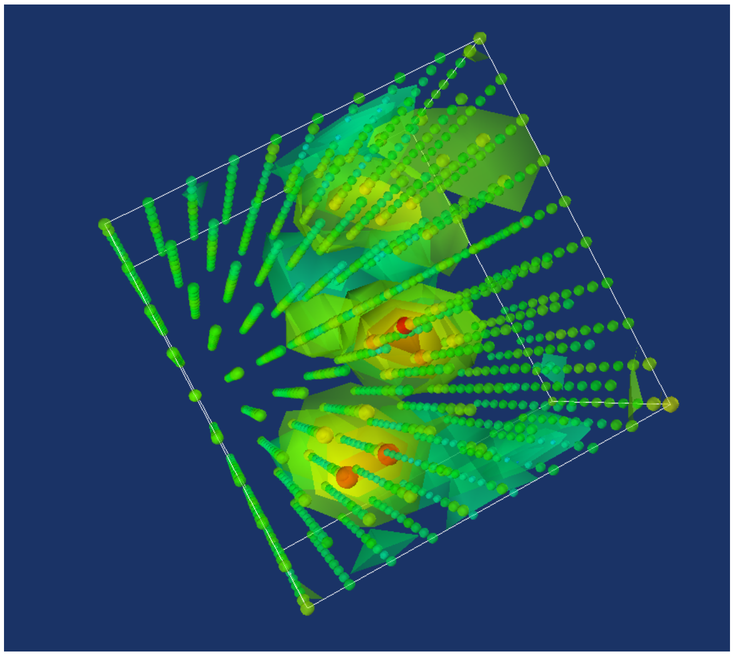


Figure 3: The figure shows iso-surfaces for the topological charge for a time-slice of small test gauge configuration. The spheres show the lattice geometry.

ing algorithm for the gauge configuration:

$$U_{x,\mu,i,j} = \mathcal{P}[\xi U_{x,\mu,i,j} + (1-\xi)\sum_{n,m} U_{x,\nu,i,m} U_{x+\nu,\mu,m,n} U^*_{x+\mu,\nu,j,n} \tag{3}$$

$$+ U^*_{x-\nu,\nu,m,i} U_{x-\nu,\mu,m,n} U_{x+\mu-\nu,\nu,n,j}] \tag{4}$$

Here $x \pm \mu$ are the coordinates of a lattice point shifted from x in direction $\pm\mu$, ξ is an arbitrary smoothing parameter, and $\mathcal{P}$ is a projection operator into the $SU(3)$ group.

4 Examples and Conclusions

This project started about six months ago. We have successfully produced a set of prototype projection plug-ins that compute topological charge, wave functions, energy density, and density of heatbath hits. Examples of images are shown in figs. 3 and 4.

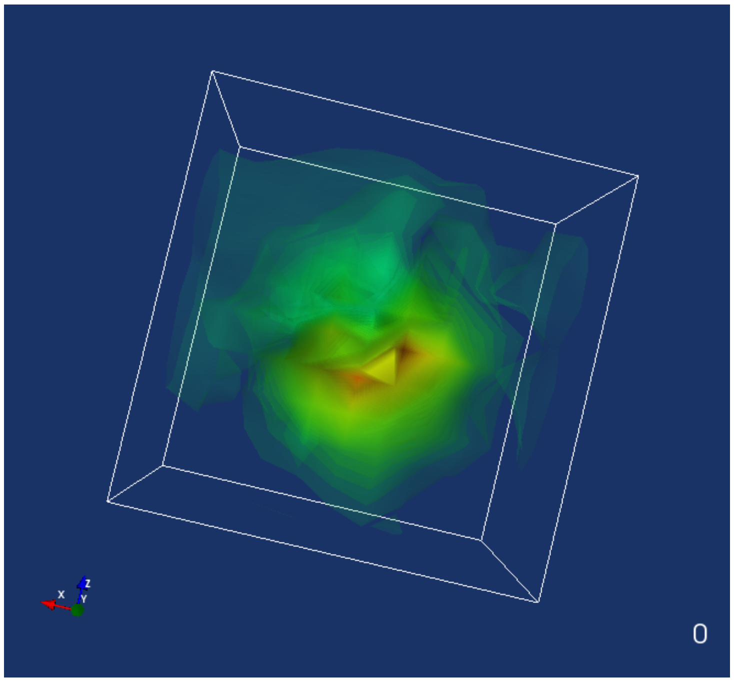


Figure 4: Wave function of a heavy-light meson computed on a single gauge configuration.

The field of visualization of Lattice QCD data is still in its infancy, but has a large potential impact for the physics community. It will help us better understand local properties of the algorithms and the spatial characteristics of extended physical objects such as mesons, hadrons and glue-balls. We also strongly believe that visualization is important to better explain what Lattice QCD is and to attract more students to this exciting area of research.

5 Acknowledgments

This project is supported by the Department of Energy under the Scientific Discovery Through Advanced Computing grant DE-FC02-06ER41441. We wish to thank the Fermilab theory group for continuing to share their data and their computing resources with the author. We also thank Dr. P. Ramachandran for developing Mayavi and his email support to this project. We finally thank the DePaul students involved in the project: Joel McGraw,

Tony Garcia, and Martin Hulth.

References

[wor, 2007] (2007). Workbench. Available from: `http://www.enthought.com`.

[apeNEXT Collaboration, 2003] apeNEXT Collaboration (2003). The apenext project. *Nuclear Physics B - Proc. Suppl.*, 119, 1038–1040.

[Bhanot et al., 2005] Bhanot, G. et al. (2005). Qcd on the bluegene/l supercomputer. *Nuclear Physics B - Proc. Suppl.*, 140, 823–825.

[Brower, 2006] Brower, R. (2006). National software infrastructure for lattice quantum chromodynamics. *Journal of Physics - Conf. Series*, (pp. 142).

[FERMILAB-Pub-01/197,] FERMILAB-Pub-01/197. *B Physics at the Tevatron: Run II and Beyond*. Technical report.

[Gottilieb, 2006] Gottilieb, S. (2006). Guest editor's introduction: Special purpose computers. *IEEE: Computing in Science and Engineering*, 8, 15.

[Gottlieb, 2002] Gottlieb, S. (2002). *Nuclear Physics B - Proc. Suppl.*, 106-107, 1031–1033.

[Holmgren, 2005] Holmgren, D. J. (2005). Pc clusters for lattice qcd. *Nuclear Physics B - Proc. Suppl.*, 140, 183–189.

[Marciano & Pagels, 1978] Marciano, W. & Pagels, H. (1978). Quantum chromodynamics. *Physics Reports*, 36, 135.

[Pierro, 2002] Pierro, M. D. (2002). Fermiqcd: A toolkit for parallel lattice qcd applications. *Nuclear Physics B - Proc. Suppl.*, 106-107, 1034–1036.

[Pierro, 2006] Pierro, M. D. (2006). An algotihmic approach to quantm field theory. *International Journal of Modern Physics A*, 21(3).

[Ramachandran, 2001] Ramachandran, P. (2001). Technical report.

[Schroeder et al., 2000] Schroeder, W. J., Avila, L. S., & Hoffman, W. (2000). Visualizing with vtk: A tutorial. *IEEE Computer Graphics and Applications*, 20(5), 20–27.

A Fast CDESSO-Based Image Retrieval System

Yung-Kuan Chan[1] and Yi-Tung Liu[2] and
Tso-Yu Liao[3] and Meng-Husiun Tsai[1]

[1]Management Information Systems, National Chung Hsing University, Taiwan

[2]Information Management, Chaoyang University of Technology, Taiwan

[3]Computer Science, National Chung Hsing University, Taiwan

`ykchan@nchu.edu.tw`,`s9014613@mail.cyut.edu.tw`,
`charles1302@hotmail.com`,`mht@nchu.edu.tw`

Abstract

This paper proposes a CDESSO (color differences on edges in spiral scan order) feature to describe the principle pixel colors and color complexity of an image. This paper also takes the feature to develop a CDESSO-based image retrieval system. The system can supply a high accuracy rate for finding out the database images that satisfy the users' requirement. In addition, it can resist the scale, rotation, and shift variations of images. This paper applies the technique of self-organization map (SOM) model of neural network to speed up the system as well.

1 Introduction

The primary goal of this paper is to develop an image retrieval system so as to assist users in retrieving the desired images instantly and effectively. In order to accelerate the system, this paper provides a filter based on the self-organization map (SOM) neural network [Bird & Chapman, 1997, Han et al., 1995] to quickly filter out most of undesired database images.

A user can use a tool like a scanner to input a query image Q into the system. This system then compares the feature values of the query image with those of the database images, which have been extracted in advance and saved in database. The database image that is most similar to Q would be delivered to the user.

This paper proposes a color differences on edges in spiral scan order feature (CDESSO feature) to describe the texture of an image. The CDESSO feature cannot only describe the principle pixel colors and texture of an image, but also the color difference between two adjacent objects on an image. Based on the CDESSO feature, this paper proposes a CDESSO-based image retrieval system. When given a query image, the system extracts the CDESSO feature of the query image, compares this feature with the CDESSO feature of each databases image, and delivers users the database images which are most similar to the query image. This system is efficient in recognizing two different images; besides, it is insensitive to the scale, shift, and rotation variations of images. To speed up the CDESSO-based image retrieval system, this paper also integrates the SOM (self-organizing map) neural network technology into the system.

2 CDESSO Feature

Before extracting the CDESSO feature of an image, all the pixels on the database images are categorized into K clusters by using K-means algorithm [Su & Chou, 2001] according to the similarity of their colors. The mean of all the pixel colors in each cluster is considered to be one color value in a color palette. The color palette containing K different colors is used as the common color palette CCP of all images (including all database images and query images). To extract the CDESSO feature of an image I, each pixel color C on I is replaced by one color in CCP that is most similar to C so as to create an image $I^{'}$, which is as large as I and the colors in CCP are its all possible pixel colors. This image $I^{'}$ is called a color-reducing image of I.

To extract the CDESSO feature of I, a corresponding variable v_i is given for every color C_i in CCP, and each pixel on $I\prime$ is scanned. The scanning process starts at the central pixel of $I\prime$ in spiral order outwardly one by one. The scanning order is shown in Figure 1. Let $P_j^{'}$ and P_j be the j-th pixels of $I\prime$ and I, respectively. During the scanning process, once the colors of the currently scanned pixel $P_j^{'}$ and its next scanned pixel $P_{j+1}^{'}$ are different, then the difference d between the colors of P_j and P_{j+1} is added to the variable v_i if the pixel color of $P_j^{'}$ is the i-th color of CCP. By repeating the procedure

26	27	28	29	30	31	32
49	10	11	12	13	14	33
48	25	2	3	4	15	34
47	24	9	1	5	16	35
46	23	8	7	6	17	36
45	22	21	20	19	18	37
44	43	42	41	40	39	38

Figure 1: The scanning order to extract the CDESSO feature

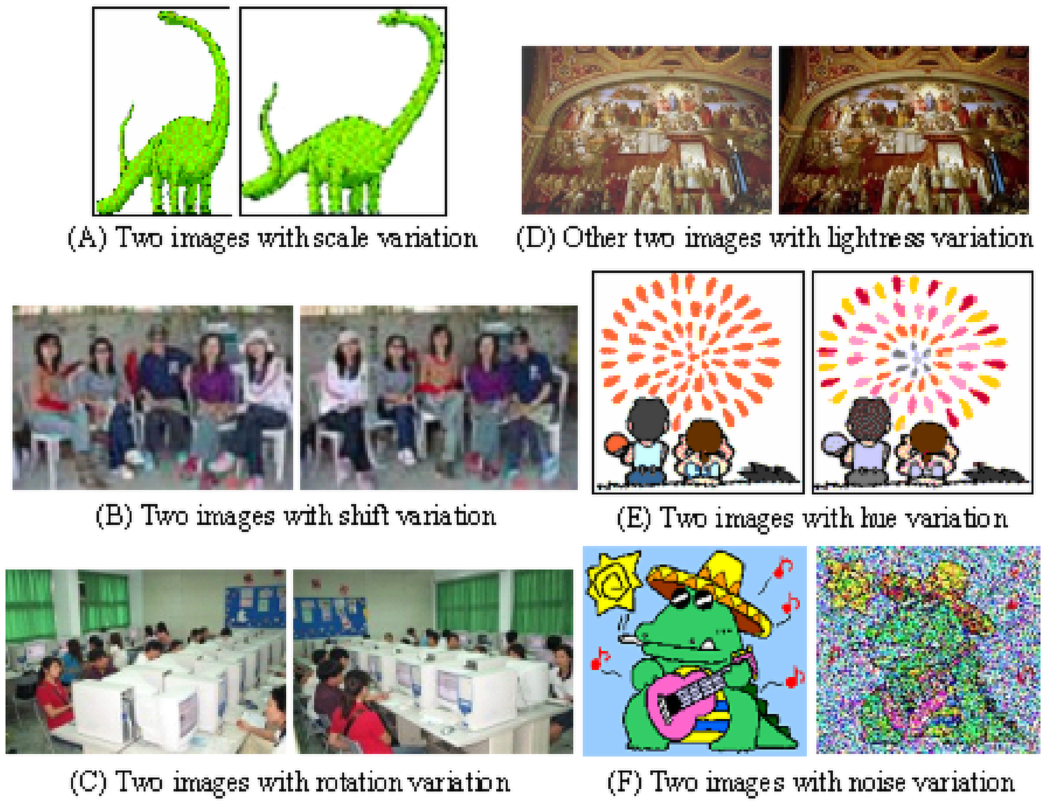

(A) Two images with scale variation
(D) Other two images with lightness variation
(B) Two images with shift variation
(E) Two images with hue variation
(C) Two images with rotation variation
(F) Two images with noise variation

Figure 2: The variation images

till the end, the values of variables $(v_1, v_2, \cdots, v_k)$ are the CDESSO feature of I. Suppose (R_j, G_j, B_j) and $(R_{j+1}, G_{j+1}, B_{j+1})$ respectively represent the three color components R, G, and B of P_j and P_{j+1}. The difference d can be computed by the following formula:

$$d = \sum_{i=1}^{K} \sqrt{(R_j - R_{j+1})^2 + (G_j - G_{j+1})^2 + (B_j - B_{j+1})^2}$$

When using a tool like a scanner to input an image, the image may be enlarged or shrunken because of different scanner resolution setups. We call this phenomenon a scale variation of the image. Figure 2(A) shows one image pair with scale variation. Let I consist of $H \times W$ pixels. To remedy the problem of scale variation, this paper divides each variable v_i by $(H + W)$. Hence, the CDESSO feature has great robustness in resisting the scale variation of images.

$(v_1, v_2, \cdots, v_k)$ are used to represent the CDESSO feature of an image, and v_i corresponds to the i-th color of CCP. Hence, the CDESSO feature can characterize the principal pixel colors of an image. The variation of the pixel colors in an image is called the color complexity of the image. The variation degree of the pixel colors on an image is called the color complexity of the image. When an image includes more regions and the colors between two adjacent regions are quite different, the CDESSO feature of the image is bigger; otherwise it is smaller. Therefore, the CDESSO feature can state not only the principal pixel colors but also the color complexity of the image.

3 The CDESSO-Based Image Retrieval System

The CDESSO feature can describe the principle pixel colors and the color complexities of an image. This paper uses this feature to develop an image retrieval system which is called CDESSO-based image retrieval system. The system finds out the database images that have the minimal image matching distances to the query image, and then delivers them to the user.

Let $(v_1^q, v_2^q, \cdots, v_k^q)$ and $(v_1^d, v_2^d, \cdots, v_k^d)$ be the CDESSOs of the query image Q and a certain database image D. The CDESSO-based image retrieval system defines the image matching distance *Dist* between Q and D as follow:

$$Dist = \sqrt{\sum_{i=1}^{K}(v_i^q - v_i^d)^2}$$

The smaller the value of *Dist* is, the more similar Q is to D. Finally, the system delivers to the user the database images which have the minimum *Dist*.

In a particular image, exchanging the positions of its objects and rotating its objects in certain degrees are separately called the shift and rotation variations of an image. Figures 2(B) and 2(C) respectively show the image pairs containing the rotation and shift variations. The CDESSO-based image retrieval system is indifferent to the scale, rotation, and shift variations of images.

However, the CDESSO-based image retrieval system is sensitive to lightness, hue, and noise variations of images. For a particular image, a user may apply a tool, such as a scanner, to generate different images with different light or hue, since different light bulbs in a scanner can be used. Figures 2(D) and 2(E) individually present the images with the lightness and hue variations. For a particular image being added a great number of noises, it may be regarded as a different image from the original one. As in Figure 2(F), there is one image pair that has the noise variation.

4 Fast CDESSO-Based Image Retrieve System

The CDESSO-based image retrieval system computes the image matching distance between the query image and each database image. To speed up the system, this paper is hence engaged in developing a fast filter to prune off a large number of unqualified database images. We call it fast CDESSO-based image retrieval system. The system uses the SOM neural network [Bird & Chapman, 1997, Han et al., 1995] to categorize the database images into G groups according to the similarity of their CDESSO features, and computes a representative CDESSO feature for each group. Since each CDESSO feature contains K components and the database images are divided into G groups, the input layer and output layer of the SOM neural network are designed respectively to contain K input neurons and G output neurons, where $G = \lfloor log_2 N \rfloor^2$ and N is the total number of database images.

Let $(v_{j1}^i, v_{j2}^i, \cdots, v_{jk}^i)$ be the CDESSO feature of the j-th database image in group i, and N_i be the number of database images in group i. The representative CDESSO feature $(v_{i1}, v_{i2}, \cdots, v_{ik})$ of group i can be defined as

$$v_{ih} = \sum_{j=1}^{N_i} v_{jh}^i, for\ h = 1, 2, \cdots, K$$

When given a query image Q, the system first extracts the CDESSO feature $(v_1, v_2, \ \cdots, v_k)$ of Q, and selects out S groups of database images where the matching distance between $(v_1, v_2, \ \cdots, v_k)$ and the representative CDESSO features of the S selected groups are minimal. The matching distance D between $(v_1, v_2, \ \cdots, v_k)$ and the representative CDESSO feature of group i is:

$$D = \sqrt{\sum_{h=1}^{K} (v_h - v_{ih})^2}$$

From the selected S groups, the CDESSO-based image retrieval system mentioned in previous section is then used to retrieve the user L database images which are most similar to Q.

For given a query Q, the CDESSO-based image retrieval system computes each image matching distance between Q and every database image. The querying time is $T_1 = N \times T_m + O(NlogN)$, where T_m is the running time to compute the matching distance of two images, and $O(NlogN)$ is the time in sorting the N image matching distances. On average, each group contains $\frac{N}{G}$ database images. For the fast CDESSO-based image retrieval system, the average querying time is $T_2 = G \times T_m + S \times \frac{N}{G} \times T_m + \frac{S \times N}{G} log \frac{S \times N}{G} = (G + S \times \frac{N}{G}) \times T_m + O(S \times \frac{N}{G} log \frac{N}{G})$. Here, $G \times T_m$ is the time to select out the S most similar groups of the database images, and $S \times \frac{N}{G} \times T_m$ is the time to compute the image matching distances between Q and the $S \times \frac{N}{G}$ database images in the S selected groups. When G and $S \times \frac{N}{G}$ are much less than N, T_2 is extremely less than T_1.

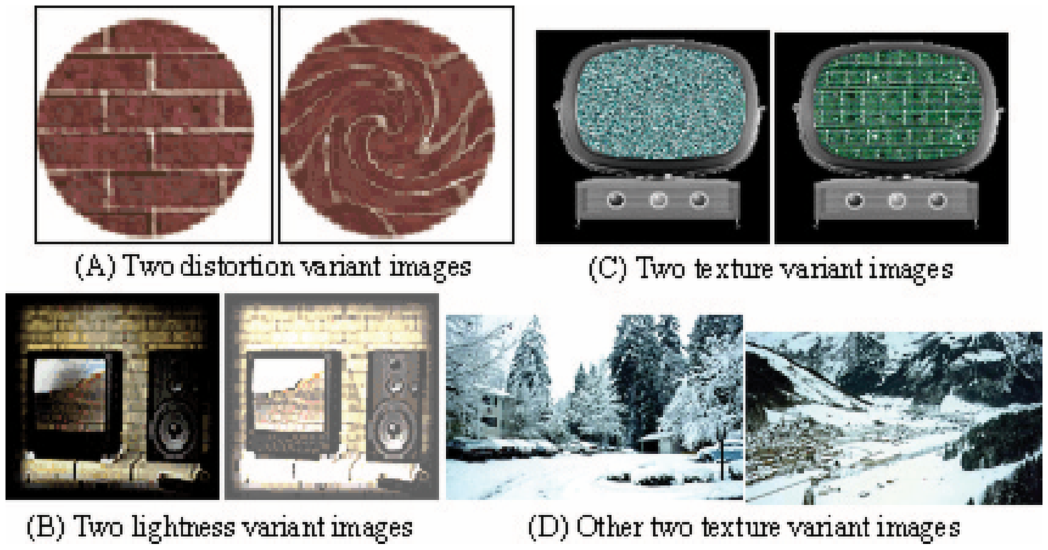

(A) Two distortion variant images
(C) Two texture variant images
(B) Two lightness variant images
(D) Other two texture variant images

Figure 3: The similar image pairs

5 Experiments

The aim of these experiments is to investigate the performance of the fast CDESSO system. $Set_D = \{I_1, I_2, \cdots, I_{1000}\}$ and $Set_Q = \{I_1^{'}, I_2^{'}, \cdots, I_{1000}^{'}\}$ are two image sets, each of which contains 1000 full color images. Some images in these two sets are drawn from animations, where each image pair $(I_i, I_i^{'})$ is randomly selected from a same animation. Some image pairs $(I_i, I_i^{'})$ considered to be similar images by human eyes are the copies from different natural images or trademark images obtained by a camera, as well as others are downloaded from internet. This paper uses them as the test images. The images in Set_D are the database images, while those in Set_Q are the query images. In these experiments, the common color palette applied in the CDESSO-based image retrieval system consists of $K = 64$ respective colors.

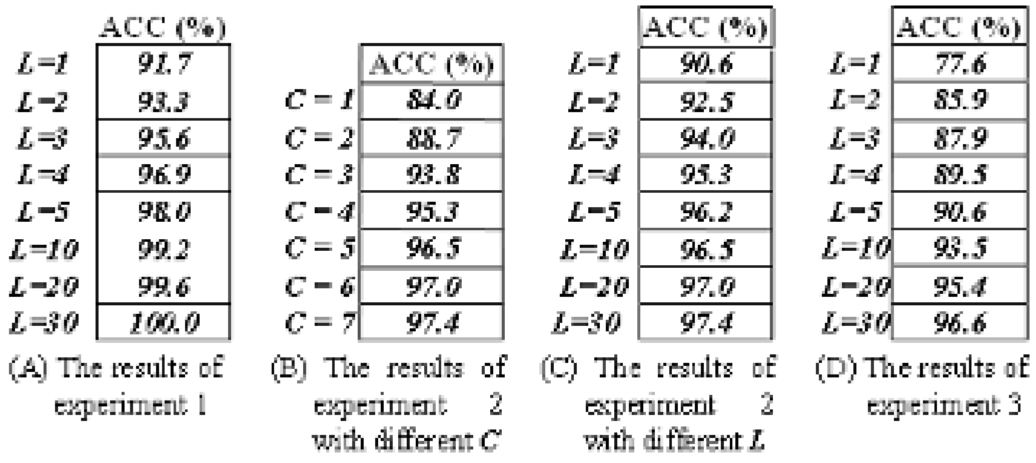
In these experiments, whenever a certain $I_i^{'}$ is selected as a query image, the image retrieval system delivers the user L database images whose matching distances to $I_i^{'}$ are shortest. If I_i is one of these L transmitted images, we say the system correctly finds out the desired image. Otherwise, the system fails to respond the desired image. In the following experiments, the accuracy rate of responding the correct answer will be explained with ACC.

The first experiment is to investigate the performance of the CDESSO-based image retrieval system. Figure 4 demonstrates the experimental results. On average, the experiment spends 191.37 seconds answering the 1000 queries. The experimental results show that the CDESSO-based image retrieval system can resist the scale, shift, and rotation variations of images. For example, it respectively regards each image pair in Figures 2(A) to 2(C) as similar images. Although the CDESSO-based image retrieval system would have lower tolerance for the noise variation of images, it can still correctly identify the left image pair in Figure 2(F). The right image pair in Figure 2(F) is from adding Gaussian noises to the left image of Figure 2(F) by using the image software Photoshop where Gaussian noise index is set to be 175. Besides, it can still identify the image pair with distortion variation, as in Figure 3(A).

The CDESSO-based image retrieval system has lower resistance capacity for the hue and lightness variations of images. For example, it cannot correctly retrieve the image pairs in Figures 2(E) and 3(B). However, it can retrieve the image pairs in Figure 2(E) with the slight lightness variations. The CDESSO-based image retrieval system would also easily be affected by texture variation in images, so it cannot correctly retrieve the image pairs as in Figure 3(C). However, it can correctly retrieve the image pair that contains little texture variation, as shown in Figure 3(D).

The second experiment is to scrutinize the performance of the CDESSO-based image retrieval system. Let N be the number of database images. The number of groups is set to be $(log_2 N)^2 = (log_2 1000)^2 \approx 100$. Therefore, the input layer and output layer of the SOM neural network respectively consist of 64 input neuron units and 100 output neuron units. In this experiment, the parameters maximum iteration number = 3000, initial learning rate = 1, final learning rate = 0.001, initial neighborhood radius = $n - 1 = 7$, and final neighborhood radius = 0.005.

The experimental results show that each group averagely comprises about 10 database images, the largest group contains 21 database images, and the smallest group holds 3 database images. Table (A) in Figure 4 shows the experimental results where for given one query image the fast CDESSO-based image retrieval system responds C groups of database images whose respective CDESSO feature is most similar to the CDESSO feature of the query image. Table (B) in Figure 4 demonstrates the experimental results where for each querying the fast CDESSO-based image retrieval system first selects out 7 groups of database images whose respective CDESSO feature are most similar to the CDESSO feature of the query image and then it



	ACC (%)
L=1	91.7
L=2	93.3
L=3	95.6
L=4	96.9
L=5	98.0
L=10	99.2
L=20	99.6
L=30	100.0

(A) The results of experiment 1

	ACC (%)
C = 1	84.0
C = 2	88.7
C = 3	93.8
C = 4	95.3
C = 5	96.5
C = 6	97.0
C = 7	97.4

(B) The results of experiment 2 with different C

	ACC (%)
L=1	90.6
L=2	92.5
L=3	94.0
L=4	95.3
L=5	96.2
L=10	96.5
L=20	97.0
L=30	97.4

(C) The results of experiment 2 with different L

	ACC (%)
L=1	77.6
L=2	85.9
L=3	87.9
L=4	89.5
L=5	90.6
L=10	93.5
L=20	95.4
L=30	96.6

(D) The results of experiment 3

Figure 4: The experimental results

picks out L database images from the 7 selected groups of database images as well as delivers the user the L database images which are most similar to the query image. In this experiment, the total execution time needed for answering 1000 queries is 52.69 seconds, faster about 3.63 times than that the CDESSO-based image retrieval system requires. Let each dimension of the CDESSO feature be described by 4 bytes. The system consumes $(1000 \times 64 + 100) \times 4 = 256400$ bytes to hold the CDESSO features of all the database images and the respective CDESSO features of all groups of database images.

Experiment 3 is to explore the performance of Huang's image retrieval system [Huang & Dai, 2003]. In this experiment, an image is split up into its RGB colour channels, and individual colour channels are used to create a grey-level image. Then each grey-level image is first decomposed to 3 levels by using discrete wavelet transform. Except the lowest frequency sub-band, each of other 9 sub-bands is treated as a subimage and its gradient vector is constructed. To reduce the length of a gradient vector, every successive 20^o directions are grouped together to form one bin. Therefore, the total number of bins in a histogram of gradient vector is =18. Besides, the different fuzzy range ϵ is given to be 0.08. Table (D) in Figure 4 lists the results of this experiment. This experiment takes 973.73 seconds in answering the 1000 queries and $1000 \times (9 \times 18 \times 3 \times 4 + \frac{2}{8}) = 1944250$ bytes in holding the sub-band gradient vectors and EDP-strings of all the database images.

6 Conclusions

This paper introduces a CDESSO feature that can depict the principal pixel colors and color-complexity of an image. Based on the feature, this paper presents a CDESSO-based image retrieval system which is indifferent to the scale, shift, distortion, and rotation variations of images, but it is sensitive to the hue and lightness variations. To accelerate the CDESSO-based image retrieval system, this paper uses the SOM neural network to classify the database images. During image querying, the system first filters out most of unqualified groups of database images, and then picks the expected ones out from the remainders. The experimental results show that the CDESSO-based image retrieval system is more efficient in accurate rate, running time, and extra memory space than Huang's system.

References

[Bird & Chapman, 1997] Bird, C. L. & Chapman, S. G. (1997). Image clustering using content-based techniques. In *International Conference on Image Processing and Its Applications*, volume 1 (pp. 385–389).

[Han et al., 1995] Han, K. A., Lee, J. C., & Hwang, C. L. (1995). Image clustering using self-organizing feature map with refinement. In *IEEE International Conference on Neural Networks*, volume 1 (pp. 465–469). Perth, Australia.

[Huang & Dai, 2003] Huang, P. W. & Dai, S. K. (2003). Image retrieval by texture similarity. *Pattern Recognition*, 36, 665–679.

[Su & Chou, 2001] Su, M. C. & Chou, C. H. (2001). A modified version of the k-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6), 674–680.

Nucleus and Cytoplasm Contour Detector of Cervical Smear Image

Meng-Husiun Tsai[1] and Yung-Kuan Chan[1] and Zhe-Zheng Lin[1] and Yun-Ju Chen[1] Chien-Shueh Chen[1] and Shys-Fan Yang-Mao[2] and Po-Chi Huang[3]

[1]Department of Management Information Systems, National Chung Hsing University

[2]Department of Computer Science, National Chung Hsing University

[3]Departments of Pathology, Taichung Hospital, Department of Health, Executive Yuan, Central Taiwan University of Science and Technology

Abstract

This paper develops a cytoplasm and nucleus contour (CNC) detector to sever the nucleus and cytoplast from a cervical smear image. In this paper, a bi-group enhancer is proposed to make a clear-cut separation for the pixels laid between two objects, and maximal color difference (MCD) method to draw the aptest nucleus contour. The CNC detector adopts a median filter to sweep off noises, the bi-group enhancer to suppress the noises and brighten the object contours, the K-mean algorithm to discern the cytoplasm from the background, and the MCD method to extract the nucleus contour on a cervical smear image.

1 Introduction

Cervical smear screening is the most popular method detecting the presence of abnormal cells arising from the cervix. The purpose of smear screening is to diagnose pre-malignant cell changes before they become cancerous. The introduction of machine assisted screening will bring significant benefits to the community, which can reduce financial costs and increase screening accuracy. An effective boundary detection algorithm locating the contours of the cytoplasm and nucleus plays an important role in developing a useful computer-assisted diagnostic system.

Many cytoplasm and nucleus morphological segmentation methods have also been proposed in the related literatures [Russo & Lazzari, 2005]. However, the results are based on tedious hand-segmentation of images. Martin [Martin, 2003] and Norup [Norup, 2005] also take the CHAMP software to segment and classify cervical smear images. Unfortunately, the CHAMP software cannot provide a satisfying segmentation performance, especially for abnormal cervical cells. The aim of this paper is to develop an automated image segmentation system to sever the cytoplasm and nucleus from a cervical smear image, without a priori knowledge of the image objects.

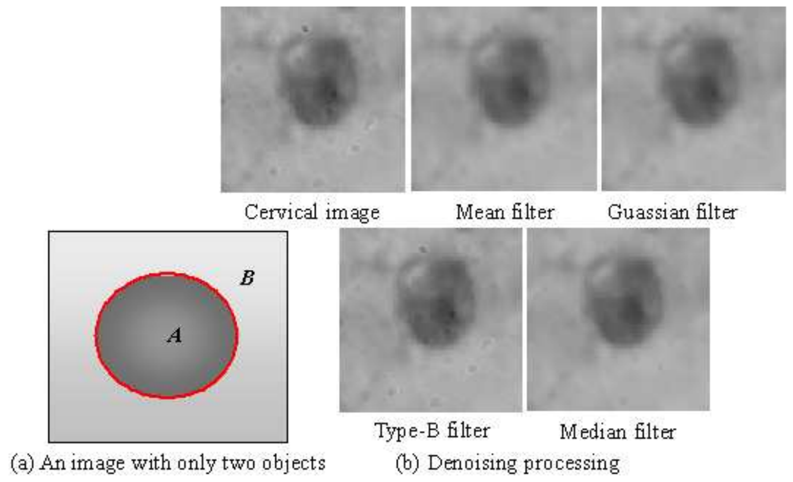


Figure 1: An image with two objects, and the denoising results

This paper proposes a bi-group enhancer to eliminate the noises on an image and to sharpen the contours of objects before extracting the object. It then employs the K-mean algorithm to discern between the cytoplasm and background on the image. Consider an image with only two different objects, as shown in the image in Figure 1(a) for example. In this image, the red circle indicates the boundary of object A where the color difference of the pixels on the inside and outside of the circle is maximal. Based on the previously mentioned properties, this paper proposes a cytoplasm and nucleus contour

(CNC) detector to perceive the cytoplasm and nucleus contours of a cervical cell.

2 The CNC Detector

Segmentation is the process of dividing an image into its constituent parts for further analysis. We commonly refer to such parts as regions of interest (ROI). Cervical smear images are frequently contaminated and the cytoplasm and nucleus contours of cervical cells are often vague, especially for abnormal cervical cells. This paper, hence, adopts a bi-group enhancer to intensify the contours of objects in an image. It also presents a maximal color difference (MCD) method to segment one object from others based on their color differences. The CNC detector contains four approaches: denoising, bi-group, cytoplasm contour detection, and nucleus contour detection approaches. This section will introduce the four approaches in detail.

2.1 Denoising Approach

The generation of an accurate edge map becomes a very crucial issue when the images are corrupted by noises. There have been several denoising techniques presented in the past studies, such as, the mean filter [Busam et al., 2001], the median filter [Busam et al., 2001], the Gaussian filter [Busam et al., 2001], and the type-B filter [Sezgin & Sankur, 2004]. Figure 1(b) is a cervical image with some noises. The median filter can efficient eliminate the noises. Therefore, this paper adopts the median filter to discard the noises.

Let $p_{i,j}$ be the pixel located at the coordinates (i, j) on a cervical smear image f_0, and $W_{i,j}$ be the corresponding window of $p_{i,j}$ which is the central pixel of $W_{i,j}$ consisting of the $m \times m$ pixels $p_{i-\lfloor\frac{m-1}{2}\rfloor,j-\lfloor\frac{m-1}{2}\rfloor}$, $p_{i-\lfloor\frac{m-1}{2}\rfloor,j-\lfloor\frac{m-1}{2}\rfloor+1}$, $\cdots$, $p_{i-\lfloor\frac{m-1}{2}\rfloor,j-1}$, $p_{i-\lfloor\frac{m-1}{2}\rfloor,j}$, $p_{i-\lfloor\frac{m-1}{2}\rfloor,j+1}$, $\cdots$, $p_{i-\lfloor\frac{m-1}{2}\rfloor,j+\lfloor\frac{m-1}{2}\rfloor}$, $p_{i-\lfloor\frac{m-1}{2}\rfloor+1,j-\lfloor\frac{m-1}{2}\rfloor}$, $p_{i-\lfloor\frac{m-1}{2}\rfloor+1,j-\lfloor\frac{m-1}{2}\rfloor+1}$, $\cdots$, $p_{i-\lfloor\frac{m-1}{2}\rfloor+1,j-1}$, $p_{i-\lfloor\frac{m-1}{2}\rfloor+1,j}$, $p_{i-\lfloor\frac{m-1}{2}\rfloor+1,j+1}$, $\cdots$, $p_{i-\lfloor\frac{m-1}{2}\rfloor+1,j+\lfloor\frac{m-1}{2}\rfloor}$, $\cdots$, $p_{i+\lfloor\frac{m-1}{2}\rfloor,j-\lfloor\frac{m-1}{2}\rfloor}$, $p_{i+\lfloor\frac{m-1}{2}\rfloor,j-\lfloor\frac{m-1}{2}\rfloor+1}$, $\cdots$, $p_{i+\lfloor\frac{m-1}{2}\rfloor,j-1}$, $p_{i+\lfloor\frac{m-1}{2}\rfloor,j}$, $p_{i+\lfloor\frac{m-1}{2}\rfloor,j+1}$, $\cdots$, $p_{i+\lfloor\frac{m-1}{2}\rfloor,j+\lfloor\frac{m-1}{2}\rfloor}$. In the median filter, the median of all pixel colors in $W_{i,j}$ substitutes for the color of $p_{i,j}$.

2.2 Bi-Group Approach

Many contextual edge detection techniques based on suppression have been previously proposed. Russo and Lazzari [Russo & Lazzari, 2005] provide a type-A filter to enhance the contours of objects. L. Yina [Corcuff et al., 1996] also presents an automatically adaptive window-level selection algorithm (so-called adaptive image optimization (AIO)) for achieving on improved performance.

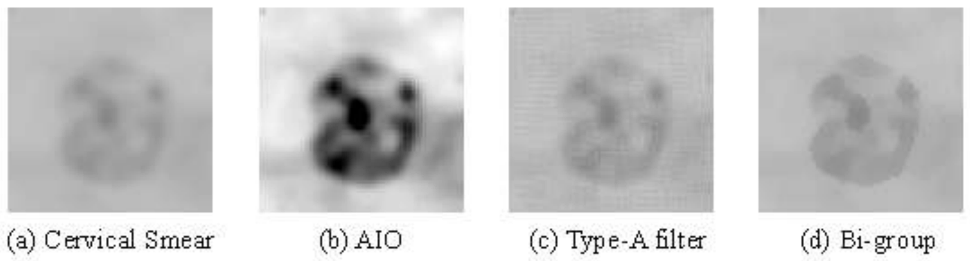


Figure 2: (a) A cell image; the cell images processed by some enhancers

Figure 2 illustrates the images processed by the type-A prefiltering and AIO algorithm. From these two images, in both methods, however, suppression has no effect on nearby edges which are definitely considered to be the background, cytoplasm, or nucleus pixels.

After the denoising approach, f_0 becomes an image f_t. Let $p_{i,j}$ be the pixel located at the coordinates (i, j) in f_t, and $W_{i,j}$ be the corresponding window of $p_{i,j}$ where $p_{i,j}$ is the central pixel of $W_{i,j}$ consisting of $m \times m$ pixels. Assume $C_{i,j} = \{c_1, c_2, \ldots, c_{m^2}\}$that are the colors of the pixels on $W_{i,j}$, and the color values in $C_{i,j}$ are sorted increasingly, that is, $c_1 \leq c_2 \leq \ldots \leq c_{m^2}$.

The Bi-group enhancer defines the interval between $\frac{m^2-1}{2} \times \sum_{i=1}^{\frac{m^2-1}{2}} c_i$ and $c_{\frac{(m \times n+1)}{2-l}}$ as well as the interval between $c_{\frac{(m \times n+1)}{2+l}}$ and $\frac{m^2-1}{2} \times \sum_{i=\frac{m^2+3}{2}}^{m^2} c_i$ as indefinite intervals since it is difficult to recognize whether $p_{i,j}$ is in an object or in a background while the color c of $p_{i,j}$ lies in both intervals. Set $index = \lceil \frac{m^2}{2} \rceil$ with m considered to be an odd number, then the two intervals are defined as $[c_1, c_{index}]$ and $\lfloor c_{index}, c_{m^2} \rfloor$. Hence, the bi-group approach replaces c with c', where c' is defined as follows:

$$c' = \begin{cases} \frac{1}{index}\sum_{i=1}^{index} c_i, & \text{if } \frac{1}{index}\sum_{i=1}^{index} c_i \le c \le \frac{1}{m^2}\sum_{i=1}^{m^2} c_i \\ \frac{1}{index}\sum_{i=index}^{m^2} c_i, & \text{if } \frac{1}{m^2}\sum_{i=1}^{m^2} c_i \le c \le \frac{1}{index}\sum_{i=index}^{m^2} c_i \\ c, & \text{otherwise} \end{cases}$$

If c is in the indefinite intervals, the bi-group enhancer changes c into the average color c_f of the first half of $C_{i,j}$'s when c is closer to c_f, or c is supplanted by the average color of the later half of $C_{i,j}$'s. Figure 2 illustrates that the bi-group enhancer can more effectively separate the object pixels from other object pixels.

2.3 Cytoplasm Contour Detection Approach

The region of nucleus is generally much smaller than those of a cytoplasm and the background on a cervical smear image. The areas of the cytoplasm and the background are larger. The CNC detector hence takes the K-mean algorithm to discern between the pixels on a cytoplasm and those on a background.

Assume that the pixel colors of a cervical smear image are from n_0 to n_1, where $0 \le n_0$, $n_1 \le 255$. Since there are only a small quantity of nucleus pixels on a cervical smear image and the pixel colors of a nucleus are darker. In order to prevent the clustering via K-mean algorithm from being influenced by the pixel colors of a nucleus, the CNC detector divides the pixels with colors from $C_{i+\varepsilon}$ to C_n into two groups. It then considers the pixels in the group with a higher pixel color to be the background pixels, and the others to be the cytoplasm pixels.

The following steps explain how to use the K-mean algorithm to partition the pixels with colors between $C_{i+\varepsilon}$ and C_n into two groups:

Step 1: Randomly select 2 different values from the interval between $C_{i+\varepsilon}$ and C_n to be the respective values of the two groups.

Step 2: Categorize each pixel of the image with color within $C_{i+\varepsilon}$ and C_n into one of the two groups according to its distance compared to the representative value of each group.

The representative value of each group is then substituted for the average color of the pixels in the group. After that, Step 2 is repeated until each group is unchanged or the iteration count is greater than a given constant.

2.4 Nucleus Contour Detection Approach

The regions of most nuclei are quite small and they have high variations in color intensity. Thus, K-mean algorithm cannot precisely discriminate them from those of cytoplasms. The CNC detector hence proposes a maximal color difference (MCD) method to detect the nucleus contour. The MCD method can be used to discern an object B from another object A. In this method, an initial contour S is first specified to partition an image into two regions inside and outside S. When given an initial contour S, the MCD method repeatedly moves the pixels on S inward or outward according to the color difference D of the regions inside and outside S until D is maximal. Here, we call D the color difference of S. As a result, S is the expected contour when D of S is maximal.

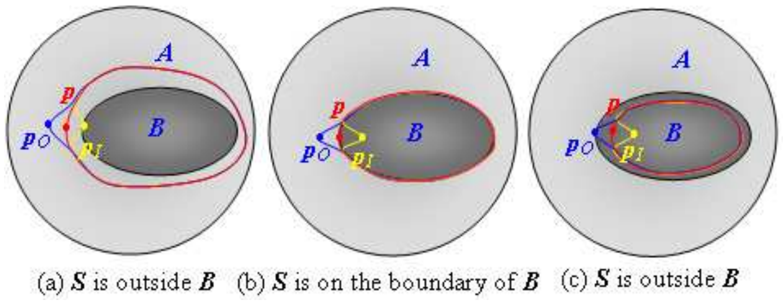


Figure 3: Three different location of S

Figure 3 demonstrates three S represented by three red circles respectively located inside, on the boundary of, and outside object B, where A and B are two different objects and p is one pixel on S. The color difference of S depicted by a red circle in Figure 3(b) is maximal among those delineated by red, blue, and yellow circles. Apparently, S in Figure 3(b) is the circle closest to the boundary of B. In Figure 3(a), the color difference of S marked by a yellow circle is maximal; in Figure 3(c), it is indicated by a blue circle which is maximal. Hence, the MCD method respectively moves the pixel p in Figures 3(a) and 3(c) to p_I and p_O.

Given an initial contour S, let n_O and n_I be the numbers of pixels outside and inside S; D, D_O, and D_I the color differences of S, S_I, and S_O; p a pixel on S; p_O.and p_I the two pixels which are closest to p on the normal line of S at p; C, $_O$, and C_I the colors of p, p_O, and p_I; S_I (resp. S_O) the same as

S, only that p is moved to p_I (resp. p_O). Following the procedure mentioned previously, the MCD method continually moves each pixel p on S until the color difference of S is maximal; moves p to p_I, if the color difference of S_I is greater than that of S_O; or else, move p to p_O.

When given an initial contour S, the MCD method repeatedly moves each pixel p on S, until the color difference of S is maximal by the following approaches in Figure 4:

```
repeat
    $D_O = \left| \frac{C_I n_I + C}{(n_I + 1)} - \frac{C_O n_O - C}{(n_O - 1)} \right|$,  $D_I = \left| \frac{C_I n_I - C}{(n_I - 1)} - \frac{C_O n_O + C}{(n_O + 1)} \right|$,
    if $D_O$=max($D$, $D_O$, $D_I$) then
        $n_O$=$n_O$-1, $n_I$= $n_I$+1
        $D$=$D_O$, $S$=$S_O$
        move $p$ to $p_O$,
    else if $D_I$=max($D$, $D_O$, $D_I$) then
        $n_I$= $n_I$-1, $n_O$= $n_O$+1
        $D$=$D_I$, $S$=$S_I$
        move $p$ to $p_I$,
    try to move the next pixel on $S$
until no pixel on $S$ is moved
```

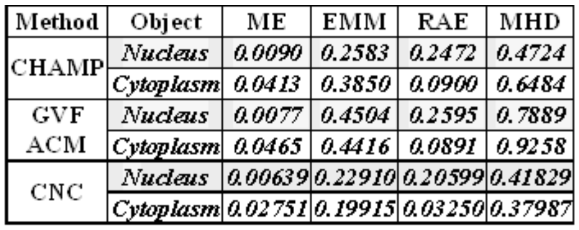
Figure 4: The approaches of the MCD method

The CNC detector moves each pixel on the detected cytoplasm contour, detected in cytoplasm contour detection approach, t pixels inward to generate the initial contour S. It then uses the MCD method to extract the nucleus contour from the detected cytoplasm.

3 Experiments

The experiments use 26 cervical smear images of 64×64 pixels as the test data. Figure 5 displays these test images and their target cytoplasm and nucleus contours which were manually drawn by an experienced doctor.

This experiment employs the CNC detector, the CHAMP software, and the GVF-ACM method to extract the cytoplasm and nucleus contours of the test images where m, ‘, and t are given to be 3, 30, and 5, and α, β, κ are



Method	Object	ME	EMM	RAE	MHD
CHAMP	*Nucleus*	*0.0090*	*0.2583*	*0.2472*	*0.4724*
	Cytoplasm	*0.0413*	*0.3850*	*0.0900*	*0.6484*
GVF ACM	*Nucleus*	*0.0077*	*0.4504*	*0.2595*	*0.7889*
	Cytoplasm	*0.0465*	*0.4416*	*0.0891*	*0.9258*
CNC	*Nucleus*	*0.00639*	*0.22910*	*0.20599*	*0.41829*
	Cytoplasm	*0.02751*	*0.19915*	*0.03250*	*0.37987*

Figure 5: The average error measures

given to be 1. Figure 5 displays a part of the cytoplasm and nucleus contours obtained in this experiment.

The performance criteria, misclassification error (ME), edge mismatch (EMM), relative foreground area error (RAE), and modified Hausdorff distance (MHD) [Sezgin & Sankur, 2004], are often used to evaluate the performances of the CHAMP software, the GVF-ACM method, and the CNC detector. Table 1 respectively lists their average measurements for the extracted cytoplasm and nucleus contours on the 26 tested images. In this table, all ME, EMM, REA, and MHD tell that the CNC detector can give a much better performance than the GVF-ACM method and the CHAMP software.

4 Conclusions

This paper develops a cytoplasm and nucleus contour (CNC) detector to segment the nucleus and cytoplast from a cervical smear image. A bi-group enhancer is proposed to isolate the pixels on one object from those on another object; it can effectively suppress the noises and enhance the object contours. This paper still uses the MCD method to segment one object from another based on their color differences. The MCD method can precisely move the contour of an object to its boundary. Additionally, the CNC detector uses K-mean algorithm to partition the nucleus and cytoplast on a cervical smear image. The results of experiments reveal that the CNC detector gives an impressive performance of object segmentation than the GVF-ACM method and the CHAMP software. Besides cervical smear images, these proposed techniques can still be employed by the object segmentation of other images.

References

[Busam et al., 2001] Busam, K. J., Hester, K., Charles, C., Sachs, D. L., Antonescu, C. R., Gonzalez, S., & Halpern, A. (July 2001). Detection of clinically amelanotic malignant melanoma and assessment of its margins by in vivo confocal scanning laser microscopy. *Archives of Dermatology*, 137(7), 923–929.

[Corcuff et al., 1996] Corcuff, P., Gonnord, G., Pierard, G. E., & Leveque, J. L. (August 1996). In vivo confocal microscopy of human skin: A new design from cosmetology and dermatology. *Scanning*, 18(5), 351–355.

[Martin, 2003] Martin, E. (2003). Pap-smear classification. *Master's Thesis, Technical University of Denmark: Oersted-DTU, Automation.*

[Norup, 2005] Norup, J. (2005). Classification of pap-smear data by transductive neuro-fuzzy methods. *Master's Thesis, Technical University of Denmark: Oersted-DTU, Automation.*

[Russo & Lazzari, 2005] Russo, F. & Lazzari, A. (February 2005). Color edge detection in presence of gaussian noise using nonlinear prefiltering. *IEEE Transactions on Instrumentation and Measurement*, 54(1), 352–358.

[Sezgin & Sankur, 2004] Sezgin, M. & Sankur, B. (January 2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1), 146–165.

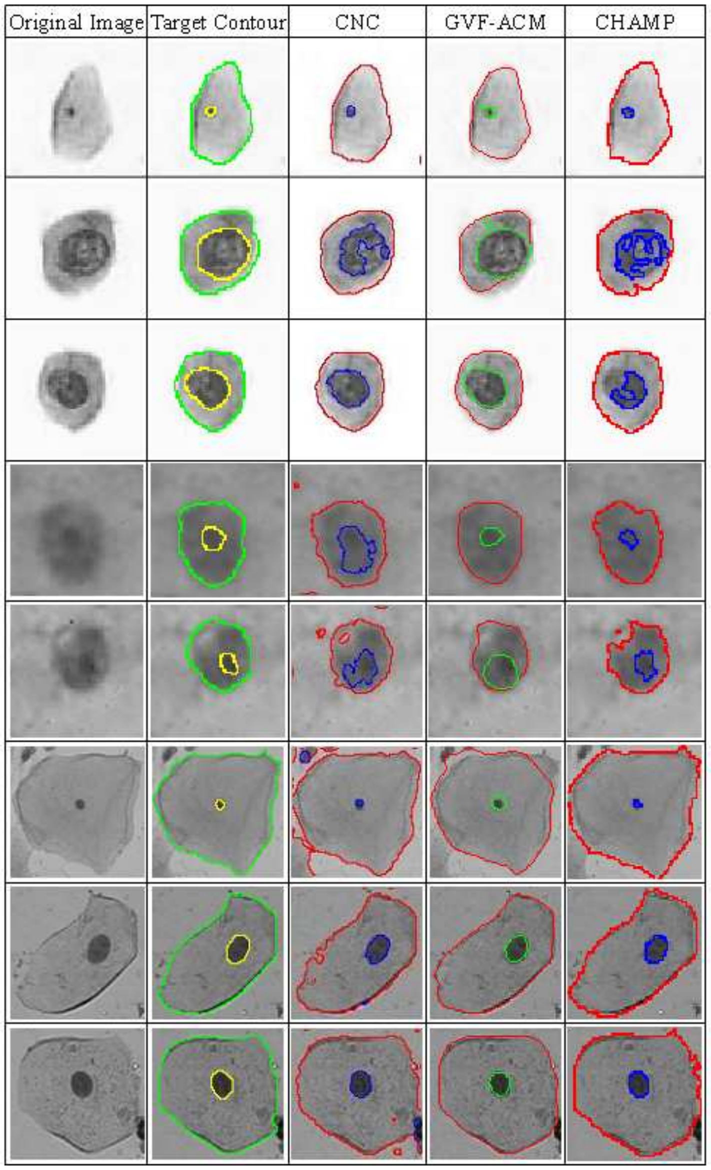


Figure 6: The segmentation results of partial images

Visualisation of Polynomials Used in Series Expansions

Philipp Schwaha, Carlos Giani,
René Heinzl, and Siegfried Selberherr
Institute for Microelectronics, TU Wien, Austria
{schwaha|heinzl|selberherr}@iue.tuwien.ac.at

Abstract

Boltzmann's Equation describes a myriad of phenomena from gas and fluid flow to electrons in semiconductors and hence plays an essential role in todays physics. However, calculating a solution to this seven-dimensional partial differential equation is very difficult. The high dimensionality of the equation poses a problem for its solution as traversal mechanisms for these high dimensions are not generally available. We present spherical harmonics which serve as the basis for an alternative, direct solution method to Boltzmann's equation. Here the solution is calculated by expanding the it into spherical harmonics and determining the corresponding coefficients. We visualise the spherical harmonics themselves, their changes due to the recursion relations, and compare their evolution.

1 Introduction

Series expansion has been a common and powerful method for the solution of complex equations for a long time. Polynomial series are often used for this task, as they obey clear rules and can be calculated with relative ease.

Boltzmann's equation which can be employed to describe the physics behind a large variety of problems, can also be treated in this manner. While it was originally incepted to govern the distribution of particles in gasses and fluids, it can also be used to describe the distribution of electrons in semiconductors [Vecchi et al., 1997] and in fact any phenomenon which involves particles that are not in a kind of thermodynamic equilibrium.

Spherical harmonics [Abramowitz & Stegun, 1964], which are themselves based on associated Legendre polynomials, are chosen for expansion. The visualisation of these polynomials and their specifications are the topic of this paper along with a means of solving the equation system resulting from the aforementioned expansion.

2 Boltzmann's Equation

As already mentioned, Boltzmann's equation can be used to describe electron transport in semiconductors. It is commonly given in the form presented in Equation 1.

$$\frac{\partial}{\partial t} f + \vec{v} \cdot \mathrm{grad}_r f + \vec{F} \cdot \mathrm{grad}_k f = \frac{\partial}{\partial t} f|_{\mathrm{collisions}} \tag{1}$$

Its solution depends not only on time and position (three dimensions) of the particle, but also on the particle's momentum (three dimensions), thus resulting in a seven-dimensional space in total. Due to the complexity inherent in this seven dimensional first order partial differential equation (PDE), simpler models, such as the drift-diffusion model which forms a mainstay of technology computer aided design (TCAD), are often derived from it [Selberherr, 1984] in order to increase calculation efficiency.

The ongoing development of smaller and faster semiconductor devices and circuits makes the solution of Boltzmann's equation even more pressing, as physical phenomena become influential, which simpler models cannot accommodate.

In addition other fields of research, which make use of Boltzmann's equation, such as gas dynamics or weather simulations, cannot easily make simplifications. It is therefore becoming an increasingly important matter to obtain a rigorous solution of this equation.

The established method of solving Boltzmann's equation in the field of TCAD is the Monte Carlo method [Jungemann & Meinerzhagen, 2003]. However, this method is computationally expensive and the statistical nature can cause problems in several situations. It is therefore desirable to also have different solutions methods available [Liotta & Struchtrup, 2000]. Series expansion using spherical harmonics offers one powerful alternative.

To this end the distribution function f is expanded using spherical harmonics Y_n^m in the following way

$$f = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} f_n^m\left(\vec{r}, k, t\right) Y_n^m\left(\vartheta, \varphi\right) \tag{2}$$

and inserted into Equation 1. This procedure is further discussed in Section 4.

3 Spherical Harmonics

Several fields of application from quantum mechanics, to investigations regarding gravity [S. T. Sutton, 1991] make use of spherical harmonics denoted as $Y_n^m(\vartheta, \varphi)$. In general they can be easily applied to problems with spherical symmetries, which can be expanded to multi-poles using spherical harmonics.

The expansion using spherical harmonics can also be viewed as a generalisation of a Fourier series expansion which works very well with periodic phenomena as found in semiconductor crystals. It is therefore obvious to apply spherical harmonics expansion to the momentum space of electrons, described by the vector $\vec{k}$, which is derived from this periodic structure.

The affinity to spherical symmetries becomes evident, when considering that spherical harmonics form the angular part of the solution of Laplace's equation in spherical coordinates (Equation 3).

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial f}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 f}{\partial\varphi^2} = 0 \tag{3}$$

Spherical harmonics form a complete system of orthogonal functions. In this work we make use of a normalised form which reads:

$$Y_n^m(\vartheta, \varphi) = (-1)^m \sqrt{\frac{2n+1}{4\pi}\frac{(n-m)!}{(n+m)!}}\; P_n^m\left(\cos\vartheta\right)\; e^{im\varphi} \tag{4}$$

P_n^m are associated Legendre polynomials [Abramowitz & Stegun, 1964]. Figure 1 shows an example of a spherical harmonic.

Recursion relations which link a spherical harmonic $Y_n^m(\vartheta, \varphi)$ to other spherical harmonics $Y_a^b(\vartheta, \varphi)$ of different order b and/or degree a, are among the properties inherited from the Legendre polynomials. A simple recursion

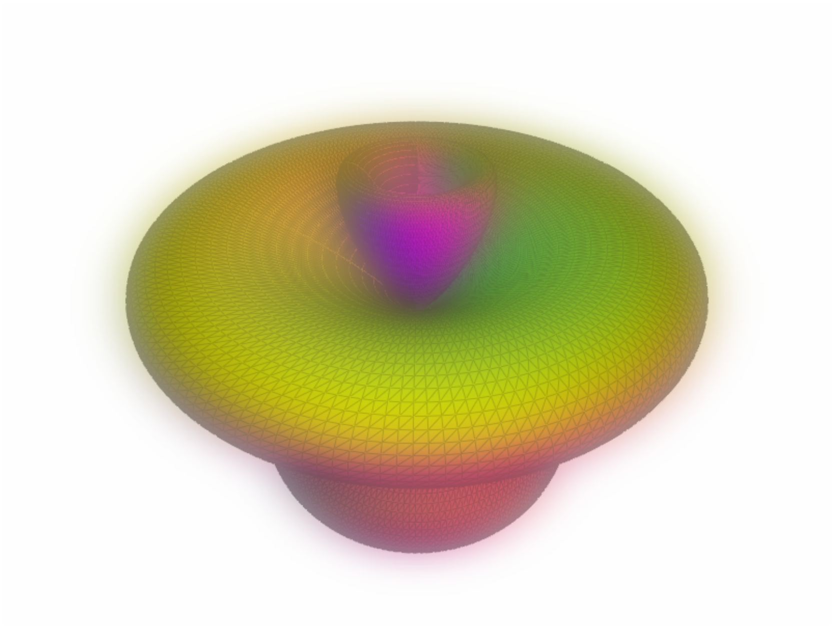


Figure 1: Spherical harmonic Y_n^m with $n = 5$ and $m = 2$.

for the index n is:

$$\cos\vartheta \; Y_n^m = \sqrt{\frac{n+m+1}{2n+1}}\sqrt{\frac{n-m+1}{2n+3}} \; Y_{n+1}^m + \; \sqrt{\frac{n+m}{2n-1}}\sqrt{\frac{n-m}{2n+1}} \; Y_{n-1}^m \tag{5}$$

A visualisation of this recursion relation is given in Figure 2. An example for a more complex recursion also involving the index m is

$$\begin{aligned} \frac{1}{\sin\theta} \; Y_n^m = \frac{1}{2m}\sqrt{\frac{2n+1}{2n+3}} \big(&\sqrt{n-m+1}\sqrt{n-m+2} \; Y_{n+1}^{m-1} \\ &- \sqrt{n+m+1}\sqrt{n+m+2} \; Y_{n+1}^{m+1}\big) \end{aligned} \tag{6}$$

It should be noted that recursions involving m are considered numerically unstable [Deuflhard, 1976], but can still greatly reduce effort, when they are used in analytical expressions. Figure 3 gives a graphical representation of this recursion.

Numerical stability has to be kept in mind not only when calculating spherical harmonics by recursion relations. The faculty in the normalising

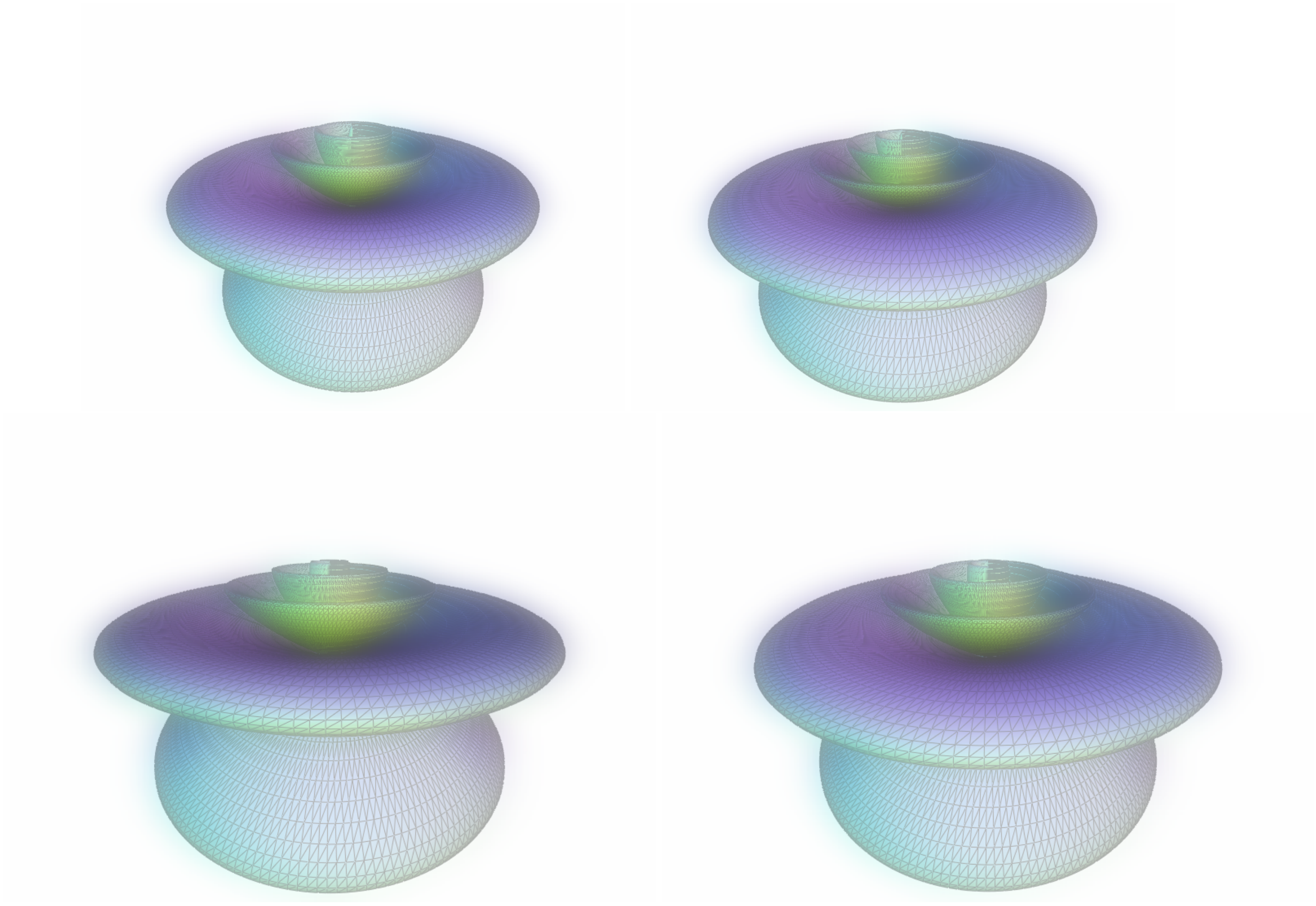


Figure 2: The recursion relation given in Equation 5. The upper left figure shows the result for $n = 8$ and $m = 2$, the upper right figure for $n = 10$ and $m = 2$. The lower left figure depicts the result of the weighted addition and corresponds to $\cos\vartheta\, Y_9^2$ and the lower right figure shows Y_9^2

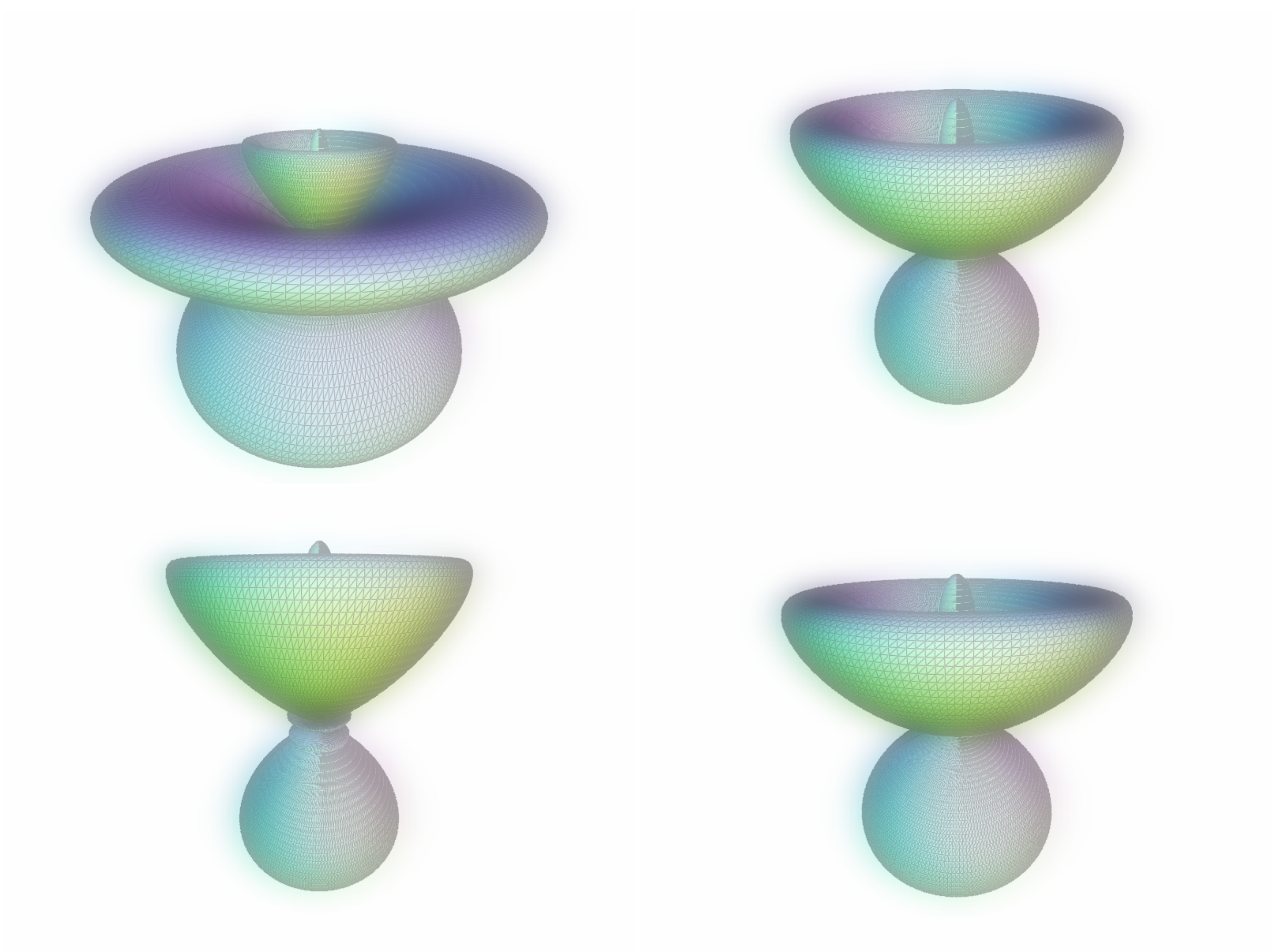


Figure 3: The recursion relation given in Equation 6. The upper left figure shows the result for $n = 6$ and $m = 2$, the upper right figure for $n = 6$ and $m = 4$. The lower left figure depicts the result of the weighted addition and corresponds to $\frac{1}{\sin\vartheta}\, Y_5^3$ and the lower right figure shows Y_5^3

factor also requires special attention for implementation, to avoid erroneous results. Simple algebra can be used to eliminate common factors in the denominator and the enumerator and data types with extended range can be used in the calculations, but these measures only push the limit of the reliably calculable spherical harmonic. While overflows of numerical values may be detected for some data types, it is not possible to discover numerical issues of recursion relations.

The left part of Figure 4 shows a spherical harmonic deformed by numerical errors. The image on the right hand side shows the correct shape of the spherical harmonic.

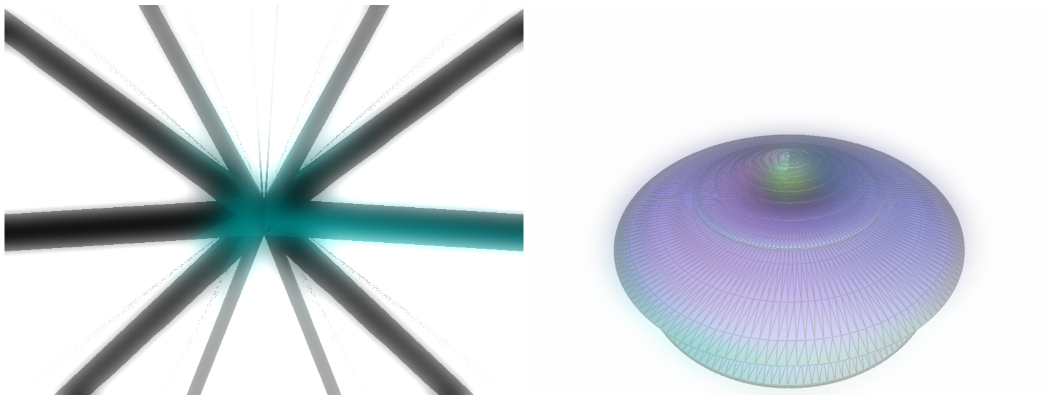


Figure 4: Two images showing Y_{21}^{2}. The spherical harmonic on the left has been calculated incorrectly by not taking proper care of numerics. The image on the right shows the result of a correct calculation.

The order of the expansion required to obtain a given accuracy depends on how well the expanding functions mimic the symmetries and the anisotropy of the solution. While it is possible to perform all calculations without any idea of the shape of the spherical harmonics, a much better understanding can be achieved when the shapes and symmetries of the expanding functions can be grasped. Visualisation of the spherical harmonic basis functions themselves is very valuable to accomplish this task.

4 Application

An equation system for the weighting factors of the spherical harmonic can be obtained by inserting the spherical harmonic expansion into Equation 1, multiplying with conjugate spherical harmonics $\overline{Y}_n^m(\vartheta, \varphi)$, and integrating over the orthogonality interval. Inserting the spherical harmonic expansion into Equation 1 and assuming that the velocity is isotropic results in:

$$\begin{aligned}&\alpha(n,m)\left[v(k)\frac{\partial}{\partial x}f_{n-1}^m + F\left(\frac{\partial}{\partial k}f_{n-1}^m - \frac{n-1}{k}\right)\right]\ Y_n^m\\ &+\beta(n,m)\left[v(k)\frac{\partial}{\partial x}f_{n+1}^m + F\left(\frac{\partial}{\partial k}f_{n+1}^m + \frac{n+2}{k}\right)\right]\ Y_n^m \end{aligned} \tag{7}$$

The integration is simple, because the $Y_n^m(\vartheta, \varphi)$ are orthogonal. In case the velocity $\vec{v}$ is anisotropic, recursion relations can be used to obtain a form that is also quite simple to integrate. The integration of Equation 7 leads directly to the matrix for the coefficients of the spherical harmonics expansion.

For the solution of Boltzmann's equation it is necessary on the one hand side to be able to integrate spherical harmonics in order to obtain the entries for the system matrix. On the other hand side it is necessary, to be able to evaluate the spherical harmonics efficiently to reassemble the solution using the appropriate coefficients.

The locality of integration and evaluation are shown in Figure 5. Evaluation yields a single point on the surface of the spherical harmonic. Integration involves the whole structure of the spherical harmonic.

5 Implementation

Our implementation of spherical harmonics makes use of several modern programming concepts such as template meta-programming to ease specification and ensure high performance. The structure of a spherical harmonic can be exploited at compile time to greatly simplify runtime calculations. This is accomplished by implementing recurrence relations as presented in Equation 5 and Equation 6 using template mechanisms [Abrahams & Gurtovoy, 2004], [Veldhuizen, 2000].

Integration of two terms containing spherical harmonics can be simplified tremendously due the orthonormality of the spherical harmonics. Because of the use of the semantic structural information available at compile time, run time evaluations of the resulting expressions are greatly simplified compared to a full integration which would otherwise have to be performed. The following C++ source code illustrates how to specify two spherical harmonics of different degree and order at compile time

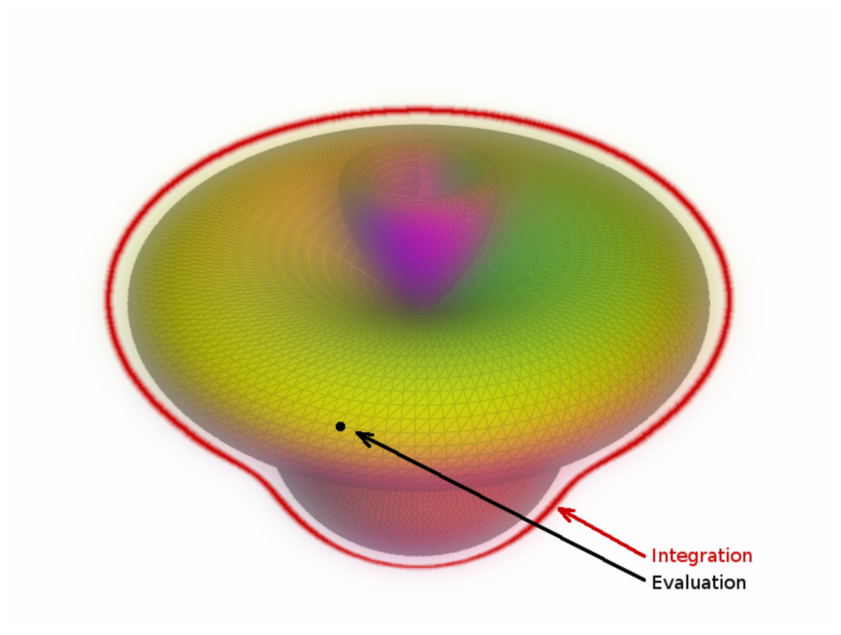


Figure 5: The locality of the operations of evaluation and integration.

```
// definition of structure
// spherical_harmonic<n,m>
spherical_harmonic<2,3> sp_2_3;
spherical_harmonic<3,0> sp_3_0;
```

Specifying degree and order defines the structure of the spherical harmonics and determines their interactions. Integration (indicated by the line surrounding the spherical harmonic in Figure 5) can make use of this structure.

```
// integration at compile time
integrate<sp_2_3, sp_3_0> integral_a;
integrate<sp_2_3, sp_2_3> integral_b;
// integral_a::value = 0
// integral_b::value = 1
```

Evaluation of single values as shown by the dot in Figure 5 can be accomplished in the following manner:

```
// evaluation at run time
std::cout << sp_2_3(0,0) << std::endl;
```

6 Conclusions

We have presented visualisations for spherical harmonics and their recurrence relations. Visualisation provides a quick and efficient way to determine, if recurrence relation based calculations are correct. It thereby provides a valuable tool for developing and debugging applications based on these methods. It also shows how naive implementations result in erroneous results, due to numerical issues.

References

[Abrahams & Gurtovoy, 2004] Abrahams, D. & Gurtovoy, A. (2004). *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth Series).* Addison-Wesley Professional.

[Abramowitz & Stegun, 1964] Abramowitz, M. & Stegun, I. A. (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* New York: Dover.

[Deuflhard, 1976] Deuflhard, P. (1976). Algorithms for the summation of certain special functions. *Journal Computing*, 17(1), 37–48.

[Jungemann & Meinerzhagen, 2003] Jungemann, C. & Meinerzhagen, B. (2003). *Hierarchical Device Simulation. The Monte Carlo Perspective.* Springer.

[Liotta & Struchtrup, 2000] Liotta, S. & Struchtrup, H. (2000). Moment Equations for Electrons in Semiconductors: Comparison of Spherical Harmonics and Full Moments. *Solid-State Electronics*, 44, 95–103(9).

[S. T. Sutton, 1991] S. T. Sutton, H. N. Pollack, M. J. J. (1991). Spherical Harmonic Representation of the Gravitational Potential of Discrete Spherical Mass Elements. *Geophysical Journal*, 107(1), 77–82.

[Selberherr, 1984] Selberherr, S. (1984). *Analysis and Simulation of Semiconductor Devices.* Wien–New York: Springer.

[Vecchi et al., 1997] Vecchi, M., Mohring, J., & Rudan, M. (1997). An Efficient Solution Scheme for the Spherical-Harmonics Expansion of the Boltzmann Transport Equation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 16, 353–361.

[Veldhuizen, 2000] Veldhuizen, T. L. (2000). Five Compilation Models for C++ Templates. In *1st Workshop on C++ Template Programming.* Available from: `http://oonumerics.org/tmpw00/`.

Visualization in the Geosciences with Paraview and Geowall

Christoph Moder[1] and Hans-Peter Bunge[1]and
Heiner Igel[1] and Bernhard Schuberth[1]

[1]Department of Earth and Environmental Sciences (Geophysics)

Ludwig-Maximilians-Universität München

moder@geophysik.uni-muenchen.de

Abstract

Numerical simulations are becoming increasingly important in the Geosciences — partly because of new sophisticated methods and tools that have been developed, but also because of the available computing power. At the Geophysics institute at the LMU Munich, the computing cluster is particularly used for simulations of the convection in the Earth's mantle and seismic wave propagation. Since the structures in the deep interior of the Earth are neither completely understood nor can be directly seen or examined, a good visualization is essential for the understanding, and a fast path from the raw data into meaningful images is crucial. In our investigations, we found that Paraview is a good tool for doing that — especially because its flexible architecture allows to run it on the very same computing cluster where the simulation has been carried out, so also large datasets can be handled; the automated preprocessing is done with the aid of VTK scripts. As a front-end, a "Geowall" can be used (a low-cost stereo projection device consisting of a PC with a dual-head graphics card, two digital projectors with polarizing filters, and a metallic screen), which facilitates the understanding of complex structures.

1 Introduction: Numerical Simulations in Geodynamics

Although the physical laws that form the structure of the Earth are rather simple, the understanding of its inner properties is made difficult by its size and the timescale of the processes. To analyze the behavior of the Earth's mantle, one can neither drill a hole and apply direct measurements, nor look inside with something like an X-ray machine. Additionally, the structures are too big and develop too slow to create them in a laboratory experiment. So there is especially the surface where measurements can be taken; methods like seismic tomography, ocean drilling, gravimetry and geodetic techniques have revealed many structures inside the Earth during the last decades, but they show only snapshots, they cannot explain their development.

This is the point where numerical modelling becomes necessary — in a highly interdisciplinary way, knowledge from many different scientific areas are used as boundary conditions to develop a model of the processes inside the Earth. With increasingly available computing power, these models have become more sophisticated in the last years and allow to address new aspects and to integrate new datasets as boundary conditions (like INSAR data or polar wandering). Visualization is the last step that supports an intuitive understanding also for non-experts — which is important especially in such interdisciplinary areas.

2 Hardware

The simulations are done on a Linux cluster, consisting of 80 nodes with two AMD Opteron processors each. Because of the comparatively low communication requirements, the nodes are connected with cheap switched Gigabit Ethernet.

The visualization is usually done on normal desktop computers (3 GHz clock frequency, 2 GB RAM). Additionally, there is a "Geowall" — a low-cost 3D projection facility ([Steinwand et al., 2002]), consisting of a PC with a dual head graphics card (Nvidia Quadro FX 1400), two NEC digital projectors with DLP technology (type: NEC LT245; screen resolution: 1024×768), a silver screen, and linear polarizing filters in front of the projectors (covered by glass, for heat protection) (see figure 1). To increase the performance, it is planned to install a small visualization cluster for distributed hardware-accelerated rendering.

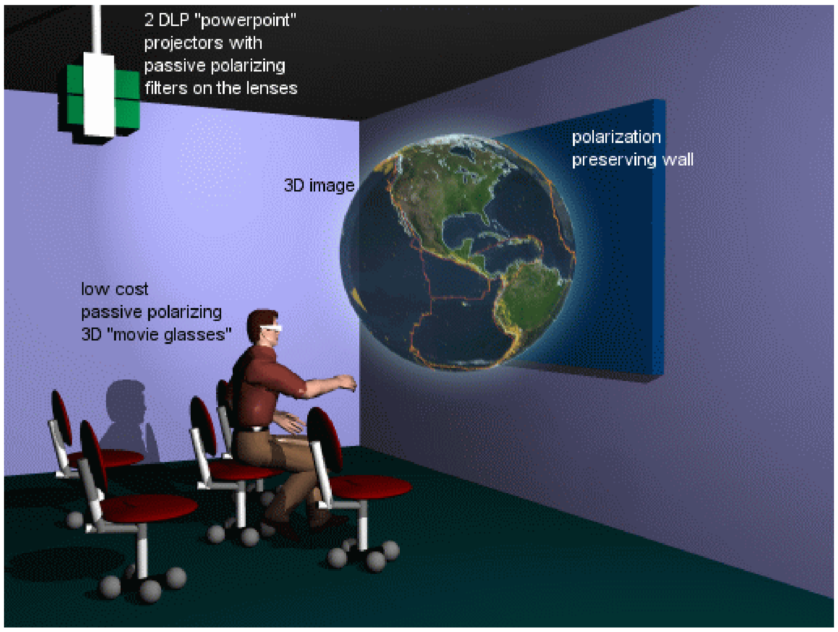


Figure 1: Basic layout of a Geowall (taken from http://geowall.org/)

3 Software and Workflow

Both Paraview ([Ahrens et al., 2000]) and VTK ([Schroeder et al., 1996]) are used to visualize the simulation data. Paraview has been compiled in 64-bit version together with the MPICH library, so it can be run on the cluster. A second version has been compiled in 32-bit for the use on desktop PCs. Thus, the program can run either standalone on a desktop computer, or as a client-server version, where the data resides distributed over the cluster nodes (where also the filtering operations take place), and a client program runs on any desktop PC.

With a little patch that was described on the Paraview mailing list, the program can also use a stereographic mode designed for shutter glasses. Together with a Nvidia Quadro graphics card, the two stereo pictures can be distributed over the two monitors — this means, Paraview can also be used with a passive stereo setup like a Geowall. This was a major breakthrough, because usually VRML viewers are used on a Geowall, which can handle only about 100,000 grid points. Paraview can handle at least one million grid points on a desktop PC, and running in client-server mode, there are no limits ([Moreland & Thompson, 2003]) — 100 million grid points were possible in our setup.

3.1 Example: Mantle Convection Simulations

The mantle convection simulations are done by solving the Navier-Stokes equation with the program TERRA ([Bunge & Baumgardner, 1995]), which is written in FORTRAN and runs on a computing cluster using MPICH as message passing library. TERRA uses an icosahedral grid; two adjacent triangles are combined to diamonds and subdivided into 32^2 subdomains, and 128 layers of this grid are used to describe the Earth's mantle. This yields a grid spacing of about 20 km, which is necessary to resolve the thermal boundary layer of the mantle with a realistic viscosity.

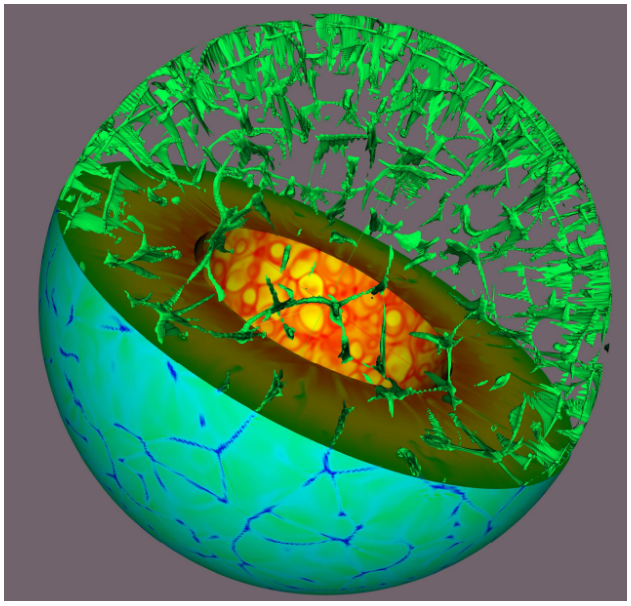


Figure 2: Mantle convection in an isoviscous earth model (with absolute temperatures): cross-section and isosurface at $T = 800$ K

Each TERRA process writes its own output file, consisting of a series of

ASCII data values (temperature and velocity field). At first, these data files were compiled to a large VTK file (ASCII; until now, only the temperature data is being used for visualization), which was possible for coarser grid resolutions — but in the resolution described above, each timestep occupies around 2.7 GB (binary data, float32); together with the grid itself, around 7 GB are needed, which cannot be handled by a single computer. (But normally, a downsampled version of the data with around 1 GB is used for visualization.)

Later, these single data files were combined each with a matching grid file to 128 single VTK files (ASCII *.vtk); these files could then be converted to a binary format (*.vtu) using VTK scripts (written in Tcl). With a wrapper file (*.pvtu), these files can be loaded by Paraview running on the cluster, so every Paraview process loads one piece of the dataset.

The typical processing consists of:

- Subtract the mean value of every radial layer from the individual temperature data values — the temperature increases from the crust to the core, but the local variations are most interesting. This optional processing step is currently done with a AWK script.

- Cut away the outer 1% of the sphere, because this is the thermal boundary layer; convection structures are only visible below. The size of the convection cells determine the dimensions of the tectonic plates that can develop on the surface (see figure 2: the blue structures in the lower half and at the isosurfaces in the upper half denote where cold material sinks down towards the core).

- Create a cross-section.

- Create isosurfaces, which reveals structures like subducting slabs and mantle plumes. Temperatures above zero means warm upwelling material, meanwhile subducting slabs at the continental margins are cold and can be shown with isosurfaces of temperatures below zero.

- In simulations where our current plate configuration has been given as a boundary condition, it makes sense to show some topography and coastlines (like in figure 3).

The latter steps are done interactively with Paraview (running on the cluster); but these steps can also be automated with VTK scripts (written in

Tcl), so the resulting file (only surfaces, instead of volumetric data) can also be viewed on a single desktop computer. The conversion scripts can also be run in parallel on several computers; thus, simulation data can be converted in a short time.

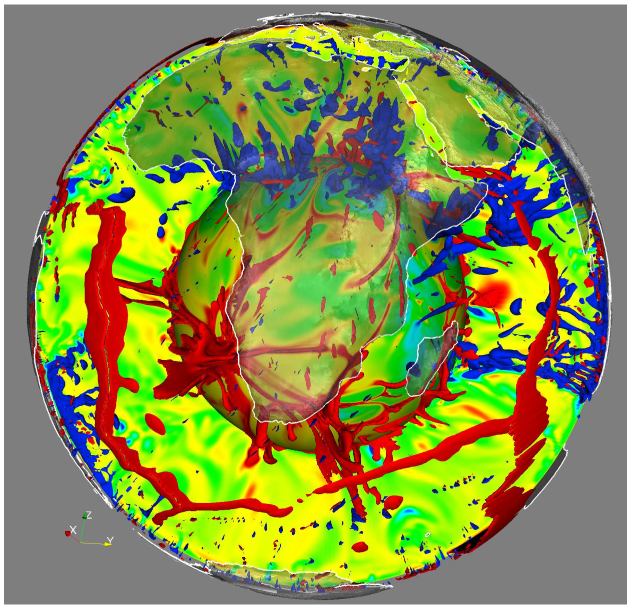


Figure 3: Convection: up- and downwellings under Africa, shown as isosurfaces (relative temperatures, subduction zones at −500 K and mid-ocean ridges at +300 K)

3.2 Example: Seismology

Also Seismology is a field of application for 3D visualization. In figure 4, an isosurface from a convection simulation (yellow structure) has been combined with the observed locations of earthquakes with magnitudes greater than 5.0 that have occured in the last decades (the focal mechanisms are displayed as "beach balls"). It turns out that both datasets are in good agreement: the hypocenters lie on the upper surface of the subducting slab, and most of the fault planes (= between the black and white parts of the spheres) are parallel to the subduction zone where most of the friction occurs..

A further example are earthquake locations under Mt. Hochstaufen (near Bad Reichenhall, Bavaria). Earthquake swarms are observed under this

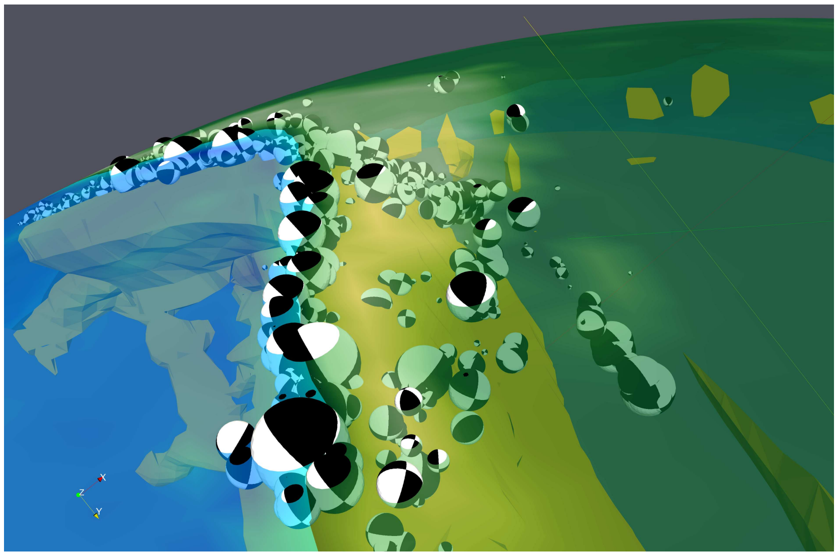


Figure 4: Earthquakes under South America (seen from Chile towards Peru); earthquake data taken from the Harvard CMT catalogue ([Harvard CMT, 2006])

mountain range, and there is strong evidence that they are triggered by torrential rain ([Kraft et al., 2006]). An animation has been created using Paraview (see figure 5); it shows that the hypocenters are clustered in a narrow area beneath the mountain, and that the seismicity migrates downwards during a swarm event.

4 Conclusion

Paraview has proved to be a valuable tool for processing large datasets interactively. Together with some VTK scripts, there is now a fast processing path from the raw data to final 3D sceneries that can be watched either on a desktop computer (usually) or on the Geowall — which does make sense for a greater audience or more complicated 3D structures, at the cost of lower speed (because the client process needs twice as much memory).

Although we are able to process our data, the conversion into the VTK format could still be improved. Since there does not seem to be any common data format, we often have to write converters. Additionally, Paraview/VTK support only cartesian coordinates — thus, for every dataset that uses the

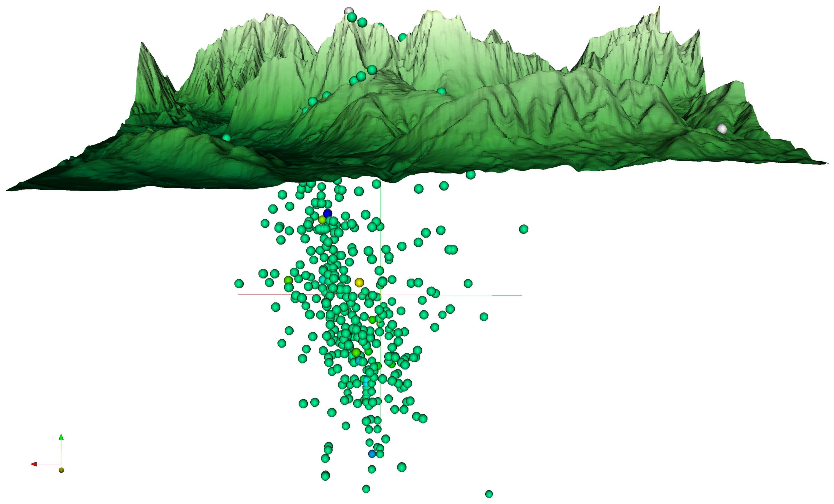


Figure 5: Hypocenter locations of the earthquakes under MtḢochstaufen in the year 2002; the topography is three times exaggerated

geographical latitude/longitude coordinate system, a grid in cartesian coordinates has to be created. But once in the VTK file format, the workflow is quite smooth.

References

[Ahrens et al., 2000] Ahrens, J., Law, C., Schroeder, W., Martin, K., & Papka, M. (2000). A parallel approach for efficiently visualizing extremely large, time-varying datasets. Available from: `http://citeseer.ist.psu.edu/ahrens00parallel.html`.

[Bunge & Baumgardner, 1995] Bunge, H.-P. & Baumgardner, J. R. (1995). Mantle convection modeling on parallel virtual machines. *Computers in Physics*.

[Harvard CMT, 2006] Harvard CMT (2006). Cmt catalog search. Available from: `http://www.seismology.harvard.edu/CMTsearch.html`.

[Kraft et al., 2006] Kraft, T., Wassermann, J., Schmedes, E., & Igel, H. (2006). Meteorological triggering of earthquake swarms at mt.

hochstaufen, se-germany. *Tectonophysics*. Available from: `http://www.sciencedirect.com/science/article/B6V72-4K9C55N-1/2/56a918df%3aec06670e36cae850a0c777`.

[Moreland & Thompson, 2003] Moreland, K. & Thompson, D. (2003). From cluster to wall with vtk. *Proceedings of IEEE 2003 Symposium on Parallel and Large-Data Visualization and Graphics*.

[Schroeder et al., 1996] Schroeder, W. J., Martin, K. M., & Lorensen, W. E. (1996). The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In R. Yagel & G. M. Nielson (Eds.), *IEEE Visualization '96* (pp. 93–100). Available from: `http://citeseer.ist.psu.edu/schroeder96design.html`.

[Steinwand et al., 2002] Steinwand, D., Davis, B., & Weeks, N. (2002). Geowall: Investigations into low-cost stereo display systems. *USGS Open File Report*. Available from: `http://geowall.geo.lsa.umich.edu/papers/Geowall.pdf`.

Acknowledgments

Thanks to ...

- Kitware Inc. (`http://kitware.org/`) for creating Paraview and VTK
- the participants of the Paraview mailing list
- Jens Oeser who is responsible for the computing infrastructure at the LMU Geophysics

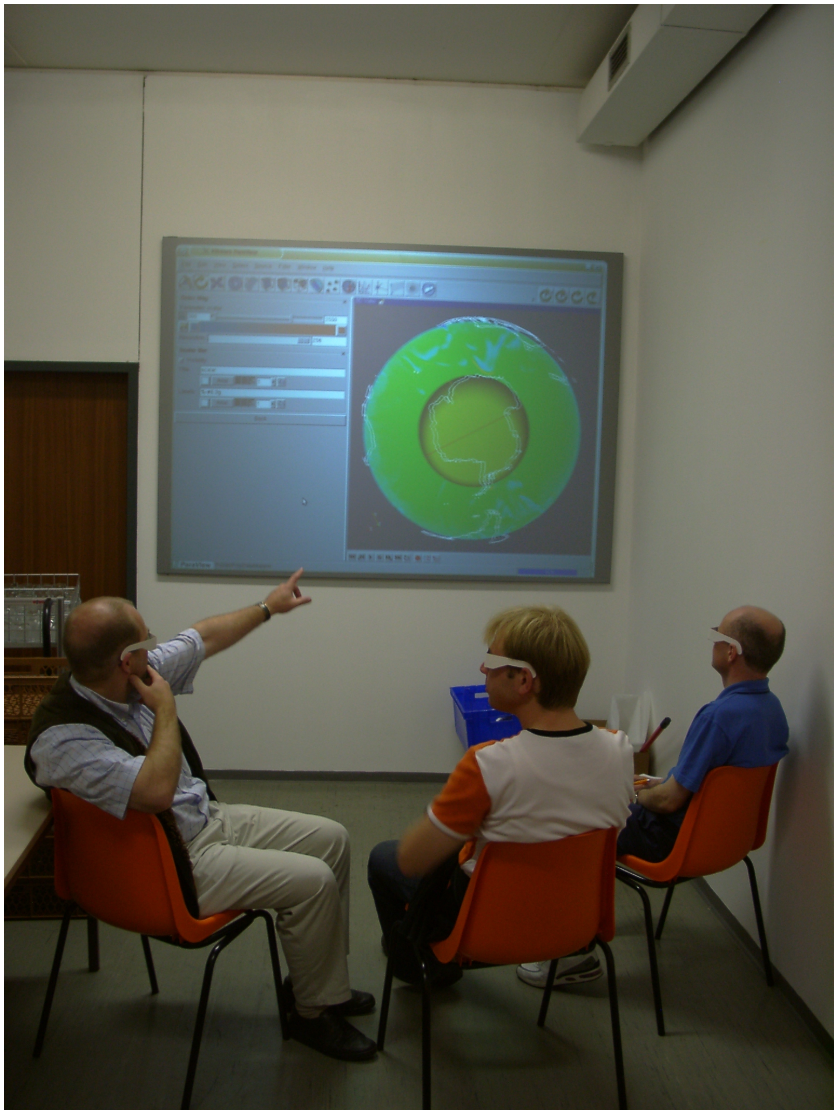


Figure 6: Geowall at the Geophysics section, LMU Munich

Some Development Directions and Comprehensive Scientific Visualization Examples of Numerical Simulation Results

Mogilenskikh D.V. and Pavlov I. V. and Petunin S.A.
Russian Federal Nuclear Centre – All-Russian Scientific and Research Institute of Technical Physics, Snezhinsk, Russia

Abstract

Some development directions of comprehensive scientific visualization are considered here as important methods of information enhancement from numerical simulation visualizations in the field of mathematical physics. These directions are identified as: consistency of algorithms, application of information layers, generalization of algorithms and ideas of scientific visualization. The proposed comprehensive approach is efficiently applied and it's usefulness is demonstrated in several examples through scientific visualization for corresponding domains.

1 Introduction

The authors propose a solution methodology for the problem of enhancement of information from scientific visualization (ISV) based on:

1. Accounting of the problem domain peculiarities.
2. Development of new functions of scientific visualization (FSV) and optimization of current ones.
3. Calculation of derived information based on the primary information explicitly written in numerical form.

4. Application of comprehensive scientific visualization (SV).

Foundations of the solution methodology for the problem of ISV enhancement are based on the following notions:

- information from scientific visualization;
- information from calculation results;
- information from FSV;
- information from SV systems.

ISV Enhancement of numerical simulation (NS) results is one of the main tasks of SV.

According to the proposed methodology for ISV enhancement, FSV are to be developed and systematically grouped to account for the problem domain peculiarities, for example, classes of problems or NS methods, dimensionality of NS geometric space, methods of description of geometry and discretization methods. Thus, FSV within the frameworks of one SV system or collection of SV systems determine consistent functionality for conducting comprehensive SV. Algorithms and methods of ISV enhancement, based on which the associated FSV are constructed, are a part of this functionality. Let us consider some directions of development of comprehensive SV, which directly refer to solution of the problem of ISV enhancement:

1. Consistency of algorithms.
2. Application of information layers.
3. Generalization of algorithms and
4. Ideas of SV.

2 Consistency of algorithms

Consistency allows to perform correct comprehensive SV with different FSV and to check adequacy and exactness of a given algorithm for a specific FSV. Let us consider this issue using several examples:

1. Consistency of the color graphic replenishment algorithm for scalar physical variable (PV) (algorithm "DLIN") [Mogilenskikh & Pavlov, 2002] with the approximation of isolines (IL) algorithm. Let us consider application of these algorithms for one model with PV distribution over a difference mesh. In the result of application of algorithm "DLIN" the boundaries of color changes correspond to specific isoline levels. If we match a number of colors in a palette for graphic replenishment with a number or a step for multiple isolines, then lines of color changes should be consistent with isolines. Figure 1 presents an example of consistency. We applied 14 tints and 15 uniform levels for isolines. Figure 2 shows application of 9 different colors and 10 uniform levels for isolines. The left figure shows filling of cells without graphic replenishment inside cells, and the right figure — with application of the graphic replenishment algorithm "DLIN". It is evident from this figure that consistency of the algorithms is important.

2. In 2D SV, FSV of space filling between consequent isolines is an essential step. For this, it is necessary to find the oriented closed polygons between isolines, that may not always be a simple task. This FSV can be implemented by the use of the graphic replenishment algorithms demonstrated earlier (Fig. 2).

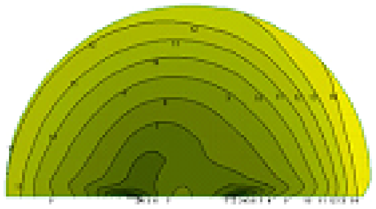


Figure 1: Example of consistency of the graphic replenishment algorithm "DLIN" and isolines approximation

1. To implement the isolines approximation algorithms one needs data about scalar PV in mesh nodes, however, PV is usually specified in mesh cells. Therefore, the problem of PV interpolation from mesh cells to mesh nodes arises. Different interpolation methods are applied. Figs. 1 and 2 show application of the method, where PV was transferred into a node, and that PV is an average of adjacent cells over a given node. Another method is

often applied as well — construction of an auxiliary grid, nodes of which are geometric centers of cells masses. The auxiliary grid has the isolines, which are further visualized at the background of actual difference mesh. At first sight these methods are comparable for consistency. However, in practice, the first method provides better results. Figure 3 presents an example for comparison. The left figure presents the second method and the right figure — the first method of PV interpolation. In the left figure, the inconsistency of interpolation method with the graphic replenishment algorithm "DLIN" is evident.

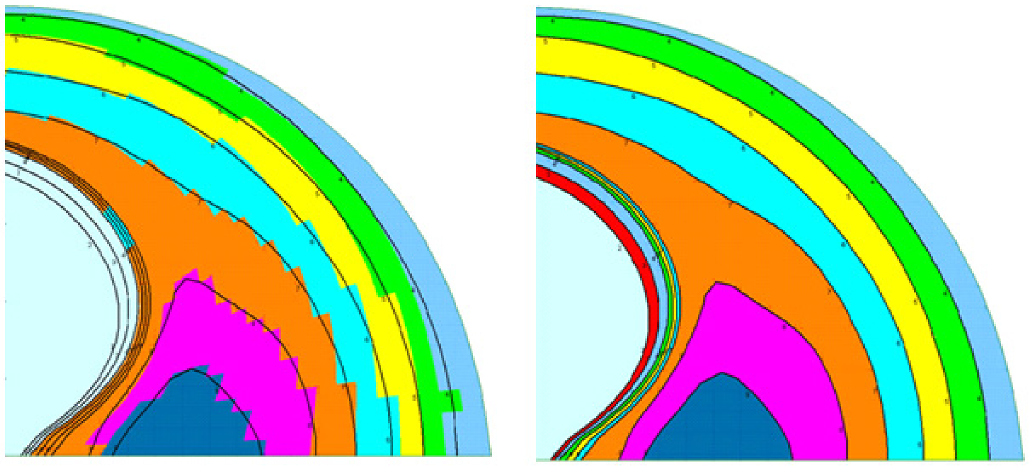


Figure 2: Examples of color visualization of PV distribution with isolines visualization

1. Figure 4 presents consistent 3D visualization of two FSVs based on the two algorithms. The example presents joint visualization of a surface of 3D volume, where geometry is specified combinatorially (algorithm TRIAN) [KM06], and flat section is shown with inside filling (algorithm CONTOUR) [Mog03], [KM05].

3 Application of information layers

Let us introduce another notion *Information Layer* – a graphic image, which is a result of an operation of a given FSV. Simultaneous application of different FSV for one image formation is an overlay of information layers. However,

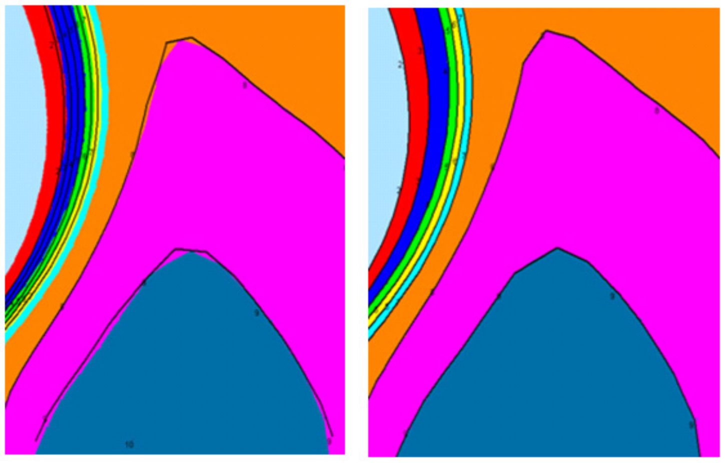


Figure 3: Comparison of isolines approximations

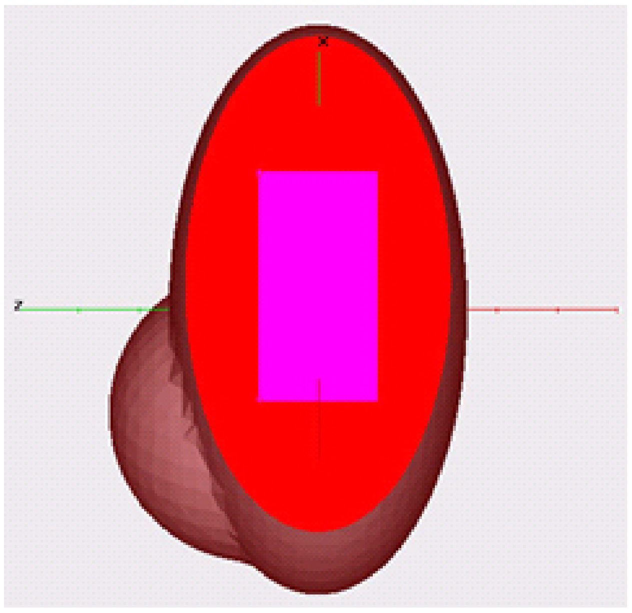


Figure 4: Consistency of 3D visualization with two different FSVs

it is should not be interpretted as overlay of one picture onto another. Let us present some examples of forming information layers composites:

1. Figure 5 presents the example of consistent comprehensive SV of discrete vector field from 2D results of NS. The vector field is visualized with four FSVs – current streamlets (red) [Dedkova et al., 2005], vectors (specified in cells nodes and colored as their module increases from white to black), half-tone filling (reflects velocity module as it increases from black to white through green), and the graphic replenishment algorithm "DLIN" applied at cells filling [Mogilenskikh & Pavlov, 2002].

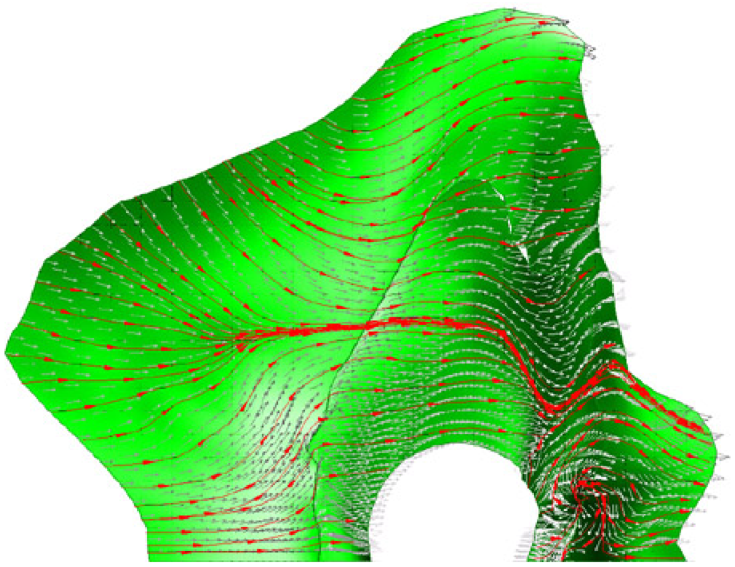


Figure 5: Example of comprehensive SV of mesh vector field

1. Figure 6 presents the example of comprehensive 3D visualization by overlay of information layers. The examples presents four FSVs – semitransparent visualization of a 3D object form, isosurfaces, cellular cutting and vectors.

1. Figure 7 presents the example of comprehensive 2D visualization of discrete scalar field using 2D results of NS. In the top section of the figure, the overlay of information layers from two FSVs is not evident, however, the algorithm of formation of nonlinear weighted consistency function

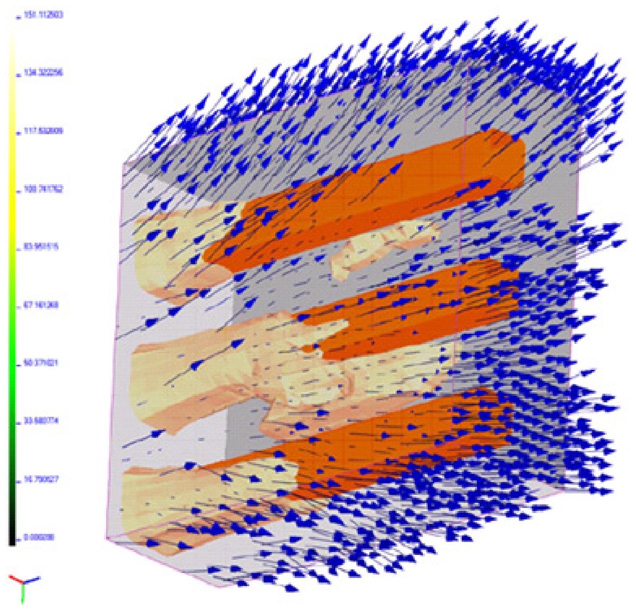


Figure 6: Comprehensive 3D SV by overlay of information layers

(WCF) between PV and indices of colors in a color palette [Mog00] and the graphic replenishment algorithm inside a cell of the difference mesh "DLIN" [Mogilenskikh & Pavlov, 2002] are used. The bottom section of the figure is shown without application of these two FSVs.

One should note *additional degrees of freedom* available during SV, which are important for comprehensive SV to form additional information layers:

1. Color space, which is especially necessary to interpret PV distribution and to obtain the effect of illumination for the analysis of a geometric 3D form. Figures 1, 2, 3, 6, 7 present the role of color space in the problem of ISV enhancement for MS results.

2. Lead time of the process under simulation, based on which the dynamic SV is performed. Dynamic SV considerably enhances ISV with additional time measurement. Figure 8 presents a sequence of time-frames from an animation of the dynamic SV with use of nonlinear WCF [Mog00]. The data are the numerical simulation results for the flight of an icy meteorite by the particle method [PSE*96].

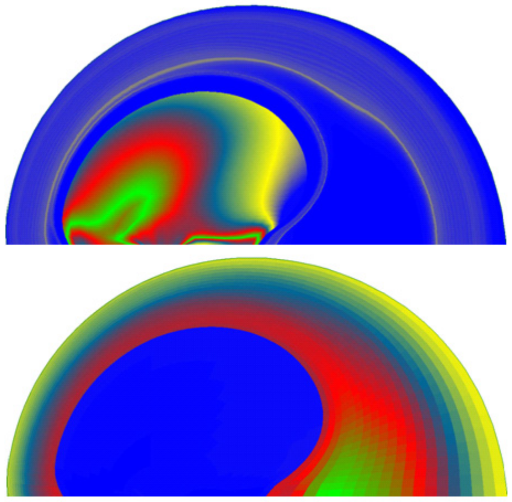


Figure 7: To the up — comprehensive 2D visualization by overlay of information layers of two FSVs (Top), the same object without application of these two FSVs (Bottom)

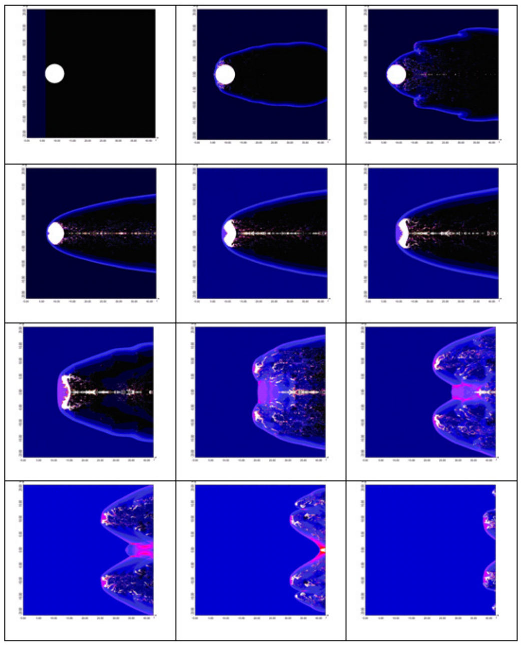


Figure 8: Time-Frames from animation of the dynamic SV

1. Ability to manipulate the visualized objects through rotations, transformations, cutting etc., is also a way to perform the dynamic SV for ISV enhancement.

2. Data of different geometric dimensionality can be reduced to larger or smaller dimensionality. For example, if we consider 2D distribution of PV on a mesh as a function of two variables, then PV can be visualized geometrically by introducing rising height over the domain. Figure 9 presents the example of 3D representation of two functions of two variables over the domain. With this method one can effectively determine extrema, "bad" cells and so on. When forming WCF, the multidimensional information about PV discrete distribution on a mesh reduces to 1D vector of weighting coefficients, that allows to select a consistency function from the color palette to PV value for ISV enhancement of NS results in automated fashion. It is clearly shown in fig. 10 for a 3D case with hidden parts for 3D distribution of PV.

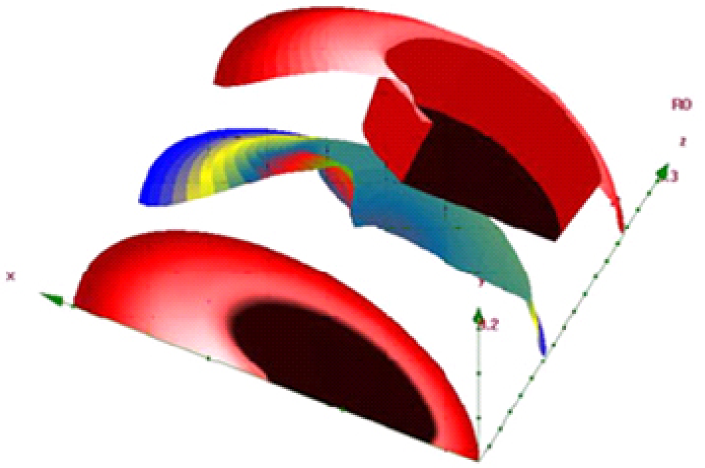


Figure 9: 3D presentation of two functions of two variables

4 Generalization of algorithms and ideas of SV

It is important to identify the critical directions for SV application where ISV enhancement is an important task:

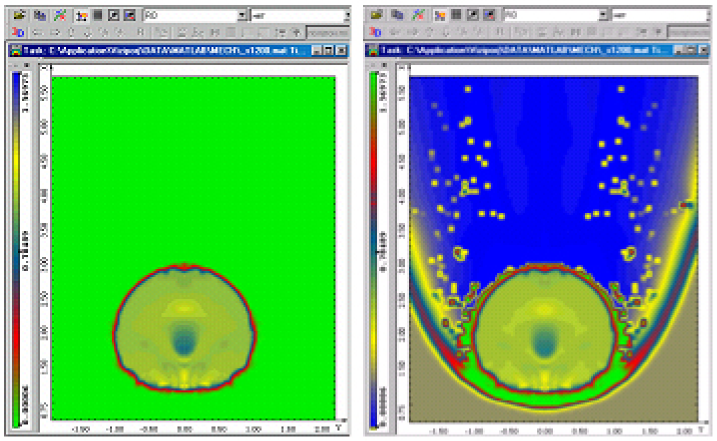


Figure 10: Comparative example of color visualization of density distribution with different CFs. linear FC (left), nonlinear WCF (right)

1. **Graphic interpretation of NS results.** It is the most important objective of SV. Result of SV, typically an image on a screen or on paper, should maximally contribute towards correctness of a solution or finding an error by using algorithms and methods of ISV enhancement. More than often, one can find an error faster visually rather than analysing the numerical results.

2. **Specification of initial data.** SV has become indispensable for a variety of tasks of initial geometry formulation, computational mesh generation, specification of initial physical state. It allows to perform visual control, provides interactive ways for specifying the necessary elements that enhance the process efficiency.

3. **Processing of NS results.**Earlier, SV systems were the ones with one-way information transmission – from MS results into SV system. Creation of feedback from SV systems has become an important task.

To this end, we list the importance of generalization of algorithms and ideas in comprehensive SV. Let us consider an example.

The underlying ideas in the methods of Gouraud and Phong [PB75] for illumination approximation inside flat polygons at 3D visualization of geometric forms are generalized and adapted to the algorithm of PV graphic replenishment inside mesh cells with the help of color at 2D SV (algorithm DLIN)

[Mogilenskikh & Pavlov, 2002] or to the algorithm of formation of current streamlets approximation [Dedkova et al., 2005]. Figure 11 presents a comparative example of color 2D visualization of scalar PV distribution on the difference mesh by various ways. In fig. 11 (right), algorithm “DLIN” was applied, where the idea of Phong interpolation was applied. Figure 12 presents approximation of current streamlets of the vector field specified on the difference mesh.

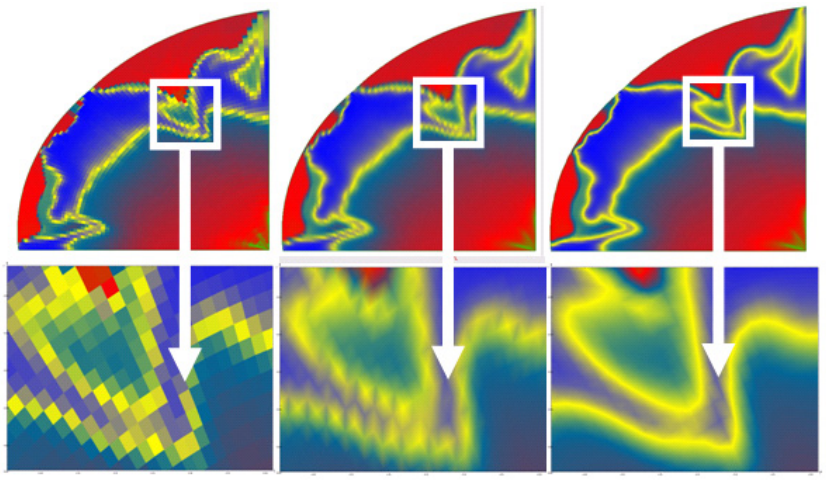


Figure 11: Comparative example of scalar field visualization. visualization by cellular method (left), visualization with replenishment in a cell with OpenGL (center), visualization with application of the replenishment algorithm “DLIN” (right)

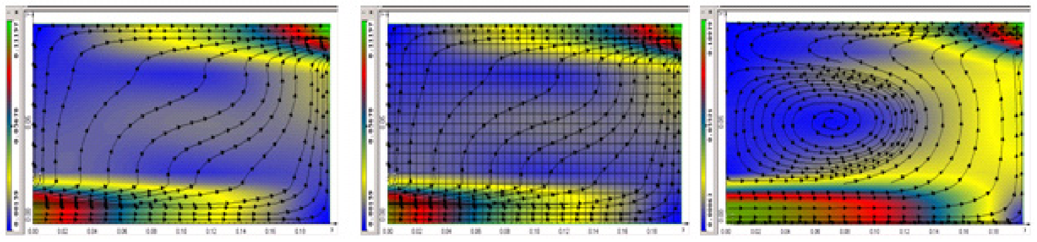


Figure 12: Vector field is presented using multiple current streamlets at different stages of a developing continuum flow

5 Conclusion

In this research paper, we have presented several directions for the development of comprehensive SV with the examples. The comprehensive approach to the solution of a SV task is an important method to enhance information content and efficiency of SV technology in different application domains, including numerical simulation for mathematical physics. Object-oriented approach to SV combined with these directions of comprehensive SV can contribute significantly towards the goal of information enhancement of SV.

The directions presented for comprehensive SV have been applied over past many years in a family of systems SV VIZI and were presented in papers [Mogilenskikh et al., 1998b], [Mogilenskikh et al., 1997], [Mogilenskikh et al., 1998a], [Melnikova et al., 2000]. The NS results from various NS codes were used for illustrations [PSE*96], [Anu76], [GGK*00], [Anuchina et al., 2004].

References

[Anuchina et al., 2004] Anuchina, N., Volkov, V., Gordeychuk, V., Es'kov, N., Ilyutina, O., & Kozyrev, O. (2004). Numerical simulation of rayleigh-taylor and richtmyer-meshkov instability using mah-3 code. *Computational and Applied Mathematics*, (168), 11–20.

[Dedkova et al., 2005] Dedkova, K., Mogilenskikh, D., Pavlov, I., & Fedorov, V. (2005). Visualization of stream lines and methods of comprehensive visualization of the discrete vector fields. *Problem of Atomic Science and Technique*, (pp. 71–79).

[Melnikova et al., 2000] Melnikova, S., Mogilenskikh, D., Pavlov, I., Fedorov, V., & Sapozhnikova, E. (2000). Principles of construction and functional meaning of visualization system for analysis of scalar and vector fields specified on 2d regular grids. In *International Workshop on Supercomputations and Mathematical Simulation* (pp. 53–55).: Sarov, Russia.

[Mogilenskikh et al., 1998a] Mogilenskikh, D., Fedorov, V., & Pavlov, I. (1998a). Visualization of 2d results of numerical simulation of physical processes. system of visualization vizi in os windows. In *The V Zababakhin Scientific Talks. International Conference. Snezhinsk* (pp. 505–511).: Snezhinsk. Published by RFNC-VNIITF.

[Mogilenskikh et al., 1997] Mogilenskikh, D., Kryukov, V., & Fedorov, V. (1997). Scientific visualization tool of analysis of mathematical simulation results. *Problem of Atomic Science and Technique*, (pp. 26, 56).

[Mogilenskikh & Pavlov, 2002] Mogilenskikh, D. & Pavlov, I. (2002). *Color Interpolation Algorithms in Visualizing Results of Numerical Simulations*, volume 972 of *Visualization and imaging in transport phenomena*, chapter Enhanced Visualization, (pp. 43–52). K-M Research/CCP.

[Mogilenskikh et al., 1998b] Mogilenskikh, D., Pavlov, I., & Sapozhnikova, E. (1998b). Methods of 3d graphic presentation of 2d data of results of solution of mathematical physics problems. In *The V Zababakhin Scientific Talks. International Conference. Snezhinsk* (pp. 521–525).: Snezhinsk. Published by RFNC-VNIITF.