Fiberbundle-Based Visualization of a Stirred-Tank Flow

Werner Benger
Feng Jijao
Center for Computation & Technology
at Louisiana State University
239 Johnston Hall
USA, LA 70803, Baton Rouge
werner@cct.lsu.edu
fjiao@cct.lsu.edu

Marcel Ritter
Center for Computation & Technology at LSU
and
Department of Computer Science
University of Innsbruck
Technikerstrasse 21a
Austria, A-6020 Innsbruck
marcel@cct.lsu.edu

Sumanta Acharya
Somnath Roy
Louisiana State University
Department of Mechanical Engineering
USA, LA 70803, Baton Rouge
acharya@me.lsu.edu
sroy13@lsu.edu

ABSTRACT

We describe a novel approach to treat data from a complex numerical simulation in a unified environment using a generic data model for scientific visualization. The model is constructed out of building blocks in a hierarchical scheme of seven levels, out of which only three are exposed to the end-user. This generic scheme allows for a wide variety of input formats, and results in powerful capabilities to connect data. We review the theory of this data model, implementation aspects in our visualization environment, and its application to computational fluid dynamic simulation of flow in an impeller-stirred tank. The computational data are given as a velocity vector field and a scalar pressure field on a mesh consisting of 2088 blocks in curvilinear coordinates.

Keywords

Computational fluid dynamics, data model, stream lines, scientific visualization.

1 INTRODUCTION

Computational fluid dynamics (CFD) has advanced significantly in the last few years and can now provide high fidelity temporally and spatially resolved numerical data. This data is based on meshes that range from a few million cells to tens of million cells, and for several hundred thousand time steps, with data files that are of the order of terabytes. A key challenge therefore is the ability to easily and cost-effectively mine this data for key features of the flow field and to display these spatially evolving features in the space-time domain of interest. In this work, we present an integrated interdisciplinary effort that takes utilizes of a generic approach to handle scientific data sets leading to new visualization capabilities in a natural way.

The CFD dataset is obtained from a large eddy simulation (LES) of flow inside a stirred tank reactor (STR), such as depicted in Figure 1. The STRs are widely used as mixing devices in chemical industry. The STR that we investigated here is a cylindrical tank with a hemispherical bottom, vertical baffles mounted along the cylindrical walls, and rotating impellers consisting rectangular blades with 45° pitch angle mounted on a shaft passing through its center, see Figure 2.

Figure 1: Stir Tank
(Courtesy Dow Chemicals)

As the impeller rotates, its blades pump the fluid axially downward towards the bottom of the tank. The fluid-jet then hits the hemispherical bottom wall and sets in a circulating motion of fluid within the tank promoting mixing between the top and the bottom of the tank.

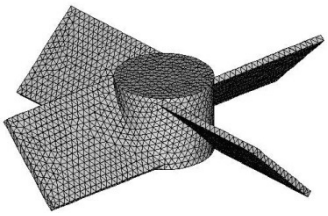

Figure 2: Impeller geometry

The calculations are done on a multi-block curvilinear mesh as shown in Figure 3. Calculations are done on nearly three million cells using a multi-block body fitted finite volume flow solver. Since the tank contains a set of impeller blades that are rotating and also contains stationary

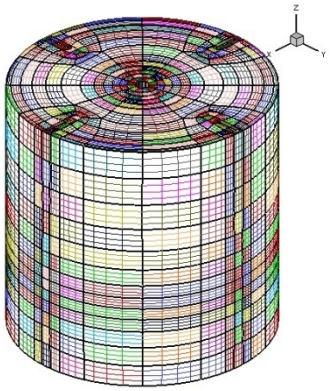

Figure 3: Boundaries of the 2088 curvilinear blocks of the computational mesh

features such as baffles on the outer walls, special algorithms (Immersed Boundary Methods) are used to accommodate moving interfaces. The method intrinsically uses a fixed curvilinear mesh fixed to the stationary features of the geometry and tracks or flags the location of the nodes adjacent to the moving surfaces or boundaries. At these flagged nodes, forcing terms/interpolations are performed to satisfy the boundary conditions at the moving surfaces. This approach has been extensively tested in an earlier communication [TRHA07] where computational solution for a similar stirred tank flow has been validated with experimental observations.

The visualizations presented here show the axial pumping as well as the circulation zones suggesting the mechanisms of fluid mixing inside the tank. As the impeller blade moves, the fluid jet stream coming out of the top and side wall of the impeller blade interact with each other to form a vortical motion following which fluid elements move in azimuthal direction [RA07]. These vortex structures are identified from the existences of pressure minima.

While traditional commercial visualization packages such as Tecplot (www.tecplot.com) and Ensight (www.ensight.com) are commonly used for visualization, they have inherent limitations. It is therefore important to be able to develop sophisticated visualization capabilities that allow a variety of features and controls not commonly available with the commercial software. The VISH visualization environment [BRH07] employed here has been designed to allow a step in this direction.

2 DATA MODEL CONCEPT

"*A common denominator for scientific data already exists, we just have to use them.*" was the paradigm under which Butler & Pendley [BP89] proposed the mathematics of fiber bundle as a foundation for a data model that is able to describe all cases needed for scientific visualization in a common way, since the mathematics behind all the different implementation is an universal language of science already. Yet a common data model has hardly been used, with the IBM Data Explore one of the few exceptions. The idea of using a fiber bundle as inspiration for a common data model has been revived by Benger [Ben04] to handle the complexity of data stemming from general relativity. These concepts have been built upon and further refined by Heinzl [HEI07] to develop generic software components. In these approaches, concepts have been based on the mathematics of topology and differential geometry to describe data sets.

2.1 Mathematics of Fiber Bundles

A fiber bundle is in the mathematical sense is set of a total space E and a base space B with a projection map f such that the union of fibers of a neighborhood U of each point of B is homeomorphic to $U \times F$ with F the so called fiber space and the projection of $U \times F$ is U again. It is also said that the space F "fibers" over the base space B. If the total space E can be written globally as $E=B \times F$, then E is called a "trivial bundle". The paradigm is that numerical data sets that are usually needed for scientific visualization can be formulated as trivial bundles. In practice it means that we may distinguish data sets by their properties in the base space and the fiber space. At each point of the – discretized – base space we have some data in the fiber space attached.

The structure of the *base space* is described as a CW-complex, which categorizes the topological structure of an n-dimensional base space by a sequence of k-dimensional *skeletons*, with the dimensionality of the skeletons ranging from zero to n. These skeletons carry certain properties of the data set: the 0-skeleton describing vertices, the 1-skeleton the edges, 2-skeleton the faces, etc., of a triangulation of some mesh. For structured grids the topological properties are given implicitly.

The structure of the *fiber space* is (usually) not discrete and given by the properties of the geometrical object residing there, such as a scalar, vector, co-vector, tensor. Same as the base space, the fiber space has a specific dimensionality, though the dimensionality of the base space and fiber space is independent. If the fiber space has vector space properties, then the fiber bundle is called a vector bundle. The most simple *vector bundle* is the *tangential bundle* of a manifold, which consists of the manifold as base space and the space of tangential vectors at each point. With n the dimensionality of the manifold, the dimension of the tangential bundle is $2n$.

Basically a fiber bundle is a set of points with neighborhood information attached to each of them. An n-dimensional array is a very simple case of a fiber bundle (with neighborhood information given implicitly).

2.2 Benefits of Fiber Bundles

The concept of a fiber bundle leads to a natural distinction of data describing the base space and data describing the fiber space. This distinction is not common use in computer graphics, where topological properties (base space) are frequently intermixed with geometrical properties (coordinate representations). Operations in the fiber space can however be formulated independently from the base space, which leads to a more reusable design of software components. Coordinate information, formally part of the base space, can as well be considered as fiber, leading to further generalization. The data sets describing a fiber are ideally stored as contiguous arrays in memory or disk, which allows for optimized array and vector operations. Such a storage layout turns out to be particularly useful for communicating data with the GPU using vertex buffer objects: fibers are basically vertex arrays in the notation of computer graphics.

2.3 The 7-Level Hierarchy

In the data model implementation of [Ben04] data are formulated in a graph of maximally seven levels, each level representing a certain property of the entire data set. These levels, constituting a "Bundle", are:

1. Slice
2. Grid
3. Skeleton
4. Representation
5. Field
6. (Fragment)
7. (Compound Elements)

Actual data arrays are stored only below the "Field" level. An actual data set is described by which data sets exist in which level. The actual meaning of each level is described elsewhere [Ben04], [BRH07], [Ben08]. Only two of these hierarchy levels are exposed to the end-user, these are the "*Grid*" and "*Field*" levels.

2.4 Bundles, Grids and Fields

An entire dataset, including all time steps or any information given on a parameter space in general, is denoted as a *Bundle*, following the mathematical term of a fiber bundle. The objective is to formulate all data that is used for scientific visualization within this *Bundle*. A *Grid* is subset of data within the Bundle that refers to a specific geometrical entity, for instance a mesh carrying data such as a triangular surface, a data cube, a set of data blocks from a parallel computation, etc. A *Field* is the collection of data sets given as numbers on a specific topological component of a *Grid*, for instance floating point values describing pressure or temperature.

The names of *Grids* and *Fields* are arbitrary and specified by the user. All other levels of the data model have pre-defined meanings and values which are used internally to describe the properties of the *Bundle* as construction blocks. The usage of these construction blocks constitutes a certain language to describe a wide range of data sets. For instance, a *Slice* is identified by a single floating point number representing time (generalization to arbitrary-dimensional parameter spaces is possible); a *Skeleton* is identified by its dimensionality, index depth (relationship to the vertices of a Grid) and refinement level; a *Representation* is identified via some reference object, which may be some coordinate system or another *Skeleton*. The meaning of such identifiers is only used internally, but transparent to the user. The lowest levels of *Fragments* and *Compound* describe the internal memory layout of a *Field* data set and are optional. They are described in detail in [Ben08].

2.5 Formulating Data

The existence of data sets in the hierarchy of a *Bundle* defines a data set. We will give some examples here on the data sets that have been involved in our work with the stir tank computational fluid dynamics data.

2.5.1 Isosurface

An isosurface is the explicit polygonal representation of the points where a possibly time-dependent scalar field given on a volumetric manifold has constant value. It consists of the following properties:

1. A sequence of *Grids*, one for each time step
2. Coordinates for each vertex, which is a field on the Skeleton of index depth 0 ("Vertices") on each grid, as represented in Cartesian coordinates.
3. Connectivity information for each triangle, which is a field on the Skeleton of dimension 2 and index depth 1 (faces) on each Grid, as represented in the Vertices.
4. Normal vectors define a vector field (precisely: a bi-vector or co-vector field) given on the Vertices Skeleton of the Grid.
5. Optional data fields on the vertices, such as another 3D field mapped on the surface.

An Isosurface is a sequence of Grid objects with two Skeletons defined on it, with the Skeleton for the Vertices carrying two or more fields.

2.5.2 Line Set

A set of lines, such as the result of the computation of stream lines of a vector field, is given by

1. A sequence of Grids, one for each time step, for the set of lines valid at each time step
2. Coordinates for each point along the lines, a field on the Skeleton of index depth 0
3. Connectivity information for the points building up a line, which is a field on the Skeleton of dimension 1 and index depth 1 (edges).
4. Tangential vectors define a vector field given on the Vertices Skeleton of the Grid
5. Optional data fields on the vertices may exist, as retrieved from a 3D field mapped on the surface.

A set of lines is very similar to an isosurface in this data model, though a different Skeleton on the Grid objects is employed.

2.5.3 Multiblock Data

The numerical data as provided by the computational fluid simulation are given as a collection of 2088 three-dimensional arrays describing coordinate location, pressure and fluid velocity for each grid point. These fields are all fibers on the vertices in the fiber bundle data model, where we also treat the coordinates as a field over the vertices. As the topological structure is regular the edges and faces are given implicitly. The decomposition of the data into blocks is represented as the internal memory layout structure of the fields, thus as field fragments visible in the 6^{th} level of the data hierarchy. The topological structure of the blocks is thereby transparent to the user, but algorithms operate on collections of arrays instead of contiguous arrays, which is relatively straightforward extension to existing algorithms such as the computation of isosurfaces. To store information that is specific to each block, we consider each block as a collection of volume cells. Volume cells are topologically 3-Skeletons in a triangulation, and of index depth 1 in the fiber bundle data model. Collections of such elements are thus of index depth 2. Fibers on this Skeleton are thus a natural place to store e.g. the geometrical bounding box information for a block, or the min/max data range of a scalar field (which speeds up repeated isosurface computations) The layout of a multiblock data set thus consists of:

1. A sequence of Grid objects, one for each time step
2. A coordinate field on the Vertices Skeleton, which is shared among all time steps if the geometry remains constant over time
3. A scalar field on the Vertices Skeleton
4. A vector field on the Vertices Skeleton
5. A Skeleton of dimension 3 and index depth 2 describing a collection of volume elements, which are the respective blocks.
6. Optional scalar fields on the (3,2) Skeleton with min/max information of a scalar field per block, or coordinate fields for bounding box.

2.6 Data Operations

Given the certain components constituting the data model, we may formulate abstract operations among such components. These operate on abstract high-level objects without requirement to know the internal structure of the objects, though their concrete implementation will have to deal with them.

2.6.1 Isosurface Computation

The computation of an isosurface is an operation that takes a Field as input and yields a Grid object, and will be called for each instance of a time sequence. The operation is parameterized by some isolevel value. Certain conditions must be fulfilled by the input field for this operation to succeed, such as being a scalar-valued field residing on a regular grid. However, more advanced implementations rather than the standard marching cubes may well be formulated through the same interface, such as direct computation of magnitude isolevels of vector fields, or isosurface computation on tetrahedral grid. The high-level operation

```
Grid = ComputeIsosurface(Field, float);
```

remains the same, and no changes in the user interface are required. For instance, more advanced operations could be invoked via some runtime plugin mechanism, transparent to the user, as it is supported by VISH.

2.6.2 Grid Evaluation

Given Field data on one Grid instance, like a 3D volume, they may be evaluated on another Grid, like a surface or a line set. Such is formulated as an EvalGrid operation which requires specification of a destination grid and a source field (implicitly given on another Grid object):

```
Field = EvalGrid(Grid Dest, Field Source);
```

Certain constraints may apply to the input Grid, since the evaluation of a field is not possible on an arbitrary Grid. For instance, data given on a surface cannot be uniquely extrapolated into an entire 3D volume. However, a wide range of operations can be specified through this common API.

3 STREAMLINE STRATEGIES

Visualizing vector fields is a common need in CFD for investigating velocity fields. [LaH05] describes several tools for investigating CFD data. [PWS06] shows the combination of different geometry based and texture based techniques in a CFD application. More applications of texture based vector field visualization can be found in [LEG].

With more complicated and large data sets it is increasingly important to have a variety of tools for feature extraction and data exploration at hand. Thus, developing a flexible framework seems the right way to meet the increasing requirements effectively. We use streamlines in our approach because it is a well known standard technique and they can be described well within the context of fiber bundles.

The challenge here was to deal with the multi-block curvilinear data structure (as shown in Figure 3) and to verify the applicability of the *Fiber Bundle* data model. Using this model, our streamline visualization implementation modules separate into four groups: *Vector Field Data*, *Seed Point Data*, *Streamline Computation* and *Line Rendering*. The software modules are connected in a *directed graph* communicating data using *Grids* and *Fields*. This allows to exchange modules by other modules (high reusability) and to combine modules in different ways (high flexibility).The *Streamline Computation Module* takes a vector *Field* and an arbitrary *Grid* defining the seed points as inputs and outputs a *Line Set* as *Grid*.

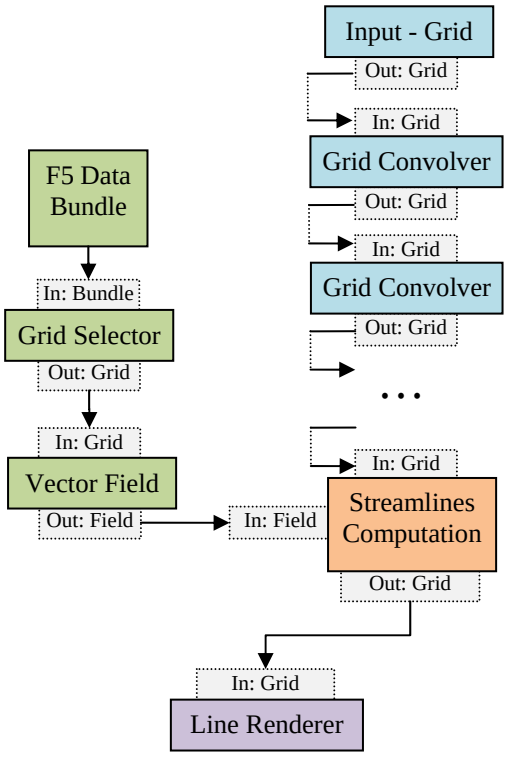

Figure 4: Streamline modules and dataflow

The computation of the streamline involves finding the location of a given *world position* in the dataset by identifying its block, cell and local curvilinear cell coordinates. These local coordinates are then used for linear interpolation of the vector field. The interpolated vector is used to advance to the next streamline point of the streamline, another world position.

Firstly, a kD-Tree is employed to find blocks, which might contain the *world position*, secondly a look up data structure called *UniGridMapper* that maps a uniform grid cell to curvilinear cells that might contain the world position is used and finally a Taylor approximation and Newton iteration, as described in [STA98], retrieves the local curvilinear cell coordinates.

Besides storing the calculated stream lines as *Line Sets* in the output *Grid* additional *Fields* are stored that carry the information necessary for interpolation. Block IDs and local cell coordinates are stored. This information can then be used by other independent modules to evaluate other data *Fields* on the streamline *Grid*, e.g. to evaluate a pressure *Field*. The final *Line Rendering* module employs *illuminated stream lines*, similar to those described in [SZH97]

3.1 Defining Seed Point for Streamlines

To visualize the characteristics of a certain vector field by streamlines it is important to find good starting points (seed points) for the streamline integration.

3.1.1 Grid Convolver

An operation called *Grid Convolver* allows the user to create sophisticated seed point geometries by 'convoluting' vertices of an input *Grid* with a set of parameters into an output *Grid*, similar to the mathematical convolution operation. Possible *Grid* convolution operations are *Point*, *Line*, *Rectangle*, *Circle*, *Ellipsoid* and *Uniform Grid*.

Figure 5 demonstrates a setup for creating seed points involving three *Grid Convolvers*. The first *Grid Convolver(a)* gets one point as input *Grid*. It 'convolutes' this input on a vertical *Line* with a subdivision of three points. This is then outputted into the second *Grid Convolver(b)* which 'convolutes' the three-point-*Line* on its horizontal two-point-*Line*. This results in the output *Grid* seen in (c). A final *Grid Convolver(d)* now 'convolutes' on its *Circle* geometry resulting in the final seed points in the output *Grid* of (d), shown in (e).

Since the module can be connected to any other modules that output *Grid* objects it is possible to create many different seed geometries. The module thus is highly reusable and flexible. Figure 4 illustrates a typical dataflow involving *Field* and

Grid objects. Figure 6, 7, and 8 demonstrate how this module was used to investigate the stir tank velocity field with different settings of position and rotation and different number of connected *Grid Convolvers.*

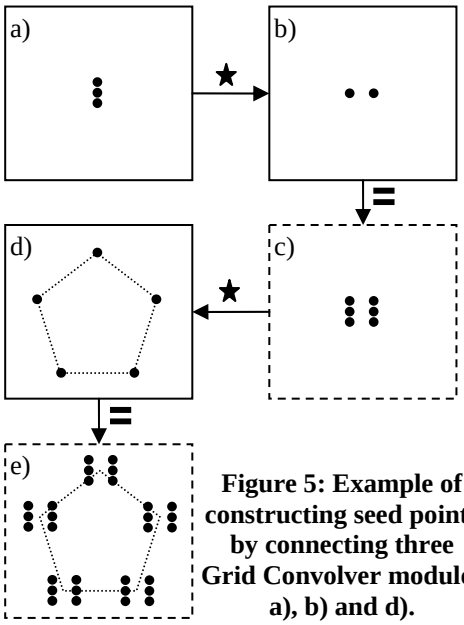

Figure 5: Example of constructing seed points by connecting three Grid Convolver modules a), b) and d).

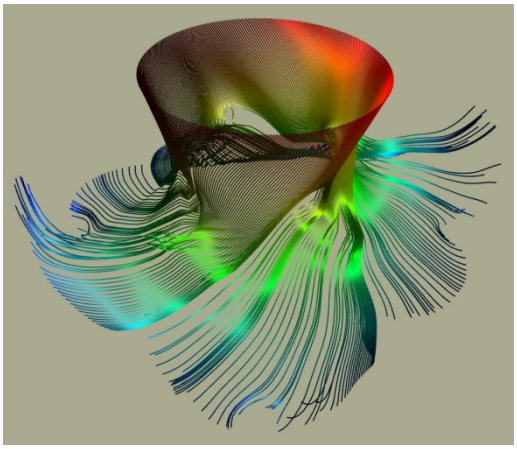

Figure 6: Streamlines with seed points on a circle.

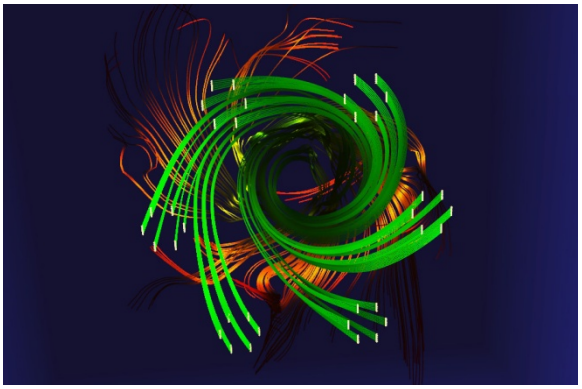

Figure 7: Streamlines emitted from seed points on a circle of circles of lines by using three Grid Convolvers.

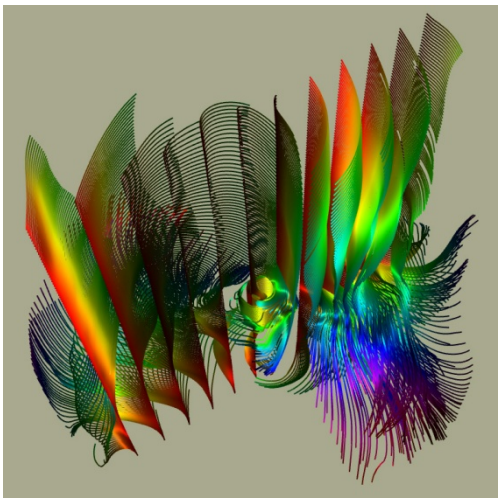

Figure 8: Streamlines emitted on an array of lines.

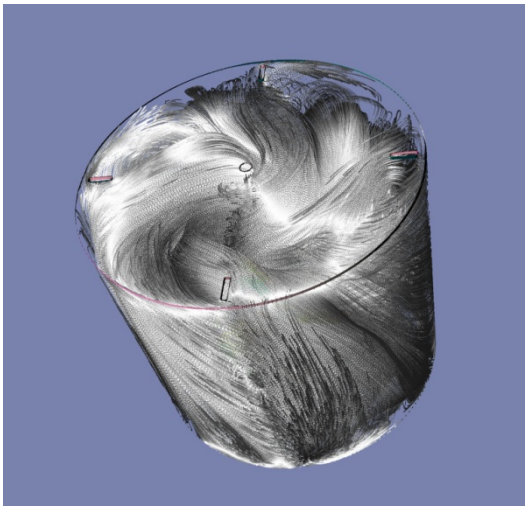

Figure 9: Space-filling streamlines in the STR

Besides interactively specifying seed points it is also possible to utilize other seeding methods such as those found in AMIRA [SWH05], where a threshold on a scalar field can specify seed points for a vector field, or similar to Weinkauf's method of computing the curvature measures of a vector field to find critical regions as indicators where streamlines were most interesting [WTH02]. Even a full coverage of the entire volume may provide worthwhile information (Figure 9).

3.1.2 Seed Points by Iso Surfaces

The usage of a *Grid* input at the *Streamline Computer* allows usage of arbitrary compatible *Grid* objects for defining seed points. For example, the *Grid* of an iso surface (the vertices of the triangular surface) computed on the pressure scalar field can be used as input, as shown in Figure 9 and Figure 10. With such streamline seeding the vector field close to the surface can be explored, similar to a texture based technique applied to the surface such as [LSH04]. This is a unique feature of VISH that can potentially be exploited to better understand the flow physics.

4 PERFORMANCE RESULTS

Computation of 100 streamlines in the given multiblock dataset of 2088 curvilinear blocks required about 7 seconds using an Euler integration scheme for 100 steps on a Intel Xeon CPU, 2.5GHz. Tecplot required 35 seconds using a comparable setup. We could not compare with Amira, since this data type is not supported there. The computation time of the isosurface crucially depends on the chosen level, and is below $1/30^{th}$ second for most values, but may require up to 2 seconds in particular cases. Computing streamlines from the isosurface vertex requires about 5-10 seconds for the setup as shown in Figure 11, but will linearly depend on the chosen length. Future improvements will utilize higher order integration schemes and parallelization, for which we expect to be able to reduce computation times under one second. The frame rates for rendering itself once the streamlines are computed are under $1/30^{th}$ of a second in all cases except Figure 9, where we got about 10 frames per second on an NVidia Quadro FX 5600 graphics board.

5 DISCUSSION

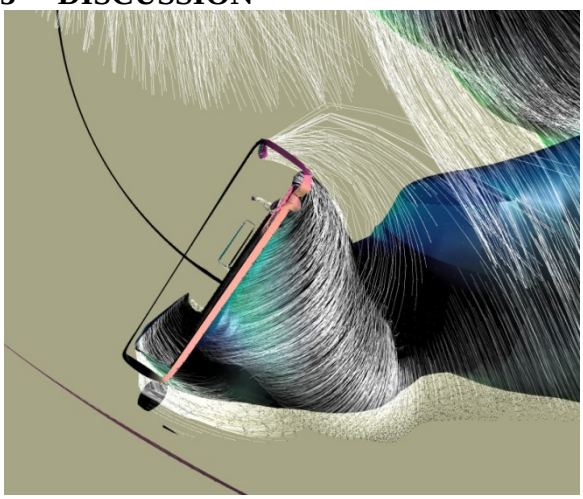

Figure 10: Detail of the emission of streamlines from the pressure isosurface.

Figure 10 shows velocity vectors adjacent to the impeller blades. It can be clearly seen that as the impeller rotates in the clockwise direction, the pitched blade surfaces force the fluid in its surface-normal direction imposing both radially outward and downward velocities on the flow along with an azimuthal velocity component due to its rotation. The pressure isosurface depicts the boundary layer formed over the impeller blade surface and its evolution downstream. Circular depressions (marked as P in Figure 11) can be observed in the pressure isosurface. These depressions point to local pressure minima suggesting formations of vortical structures in the direction opposite to the impeller rotation. These vortices (commonly termed as trailing-edge vortices) convect fluid in the azimuthal direction and play a key role in mixing within the tank.

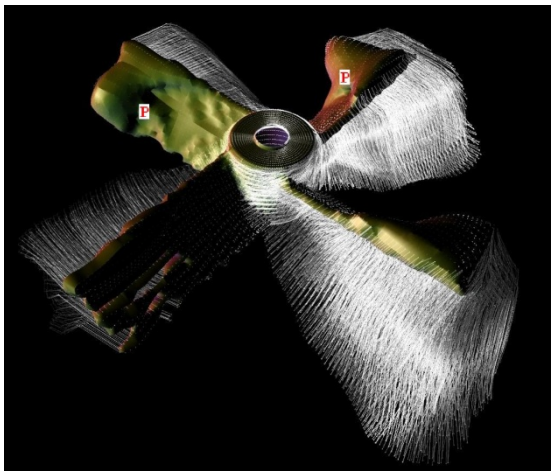

Figure 11: Velocity micro-streamlines and pressure isosurfaces over the impeller blade

Figure 12 shows the streamlines coming out of an impeller blade. The downward pumping from the impeller is clearly observable. Also, the streamlines are observed to bend near the hemispherical bottom of the tank and establish a circulating motion. The colors along the streamlines indicate the residence time of the fluid particles. The suction of the upper fluid is identified by the green color, the yellow colored impeller jet stream has both radial and axial component as the streamlines show almost a 45° bend. Later (orange color) these lines hit the bottom of the tank and bend upwards (red coloration) and the fluid is again convected to the upper part of the tank. This circulation is how the fluid mixing operation takes place in the stirred tank.

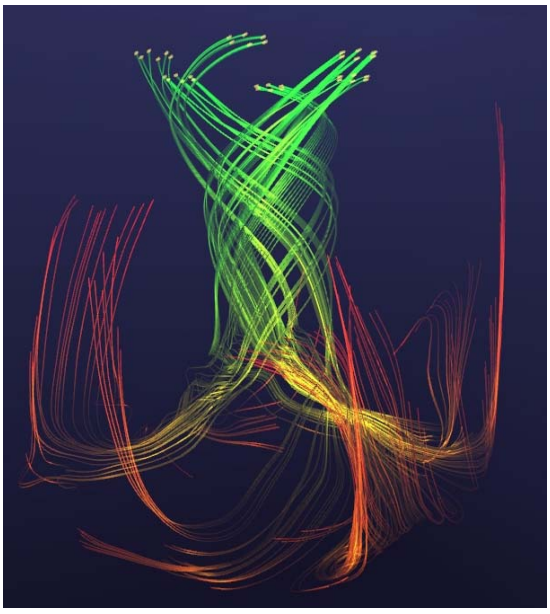

Figure 12: Streamlines over the tank volume

6 SUMMARY

We have described and applied the concept of "Grid objects" as elementary tools within a highly modular visualization environment to provide powerful seeding mechanisms for streamline computation to visualize flow in a stirred tank. Operations such as "Grid convolution" and seeding by pressure isosurfaces are natural consequences of utilizing the described data model. It is argued here that this approach offers specific capabilities that several other visualization platforms are unable to provide.

7 ACKNOWLEDGMENTS

This research employed resources of the Center for Computation & Technology at LSU, which is supported by funding from the Louisiana legislature's Information Technology Initiative. Portions of this work were supported by NSF/EPSCoR Award No. EPS-0701491 (CyberTools). We thank Georg Ritter for his industrious work on the VISH infrastructure.

8 REFERENCES

[Ben04] Benger, W. **Visualization of General Relativistic Tensor Fields via a Fiber** , PhD - thesis FU Berlin, 2004

[BRH07] W. Benger, G. Ritter and R. Heinzl, **The Concepts of VISH**, Proc. 4th High End Visualization Workshop Obergurgl, p. 26-39, 2007, Lehmann's Media

[Ben08] Benger, W. **Colliding galaxies, rotating neutron stars and merging black holes - visualizing high dimensional data sets on arbitrary meshes**, *New J. Phys. 10 (2008) 125004*, http://stacks.iop.org/1367-2630/10/125004.

[BP89] O. Butler, D. M. & Pendley, M. H. (1989). **A visualization model based on the mathematics of fiber bundles**. *Comp. in Physics*, 3(5), 45-51.

[HEI07] R. Heinzl: "*Concepts for Scientific Computing*"; PhD Thesis; Institut für Mikroelektronik, TU Wien, 2007;

[LaH05] Robert S. Laramee & Helwig Hauser, **Interactive 3D Flow Visualization Based on Textures and Geometric Primitives**, in NAFEMS World Congress Conference Proceedings, The International Association for the Engineering Analysis Community, May 17-20, 2005, St. Juliens Bay, Malta

[LEG] R. S. Laramee, G. Erlebacher, C. Garth, T. Schafhitzel, H. Theisel, X. Tricoche, T. Weinkauf, D. Weiskopf, **Applications of Texture-Based Flow Visualization**, Engineering Applications of Computational Fluid Mechanics, in publication

[LSH04] R. S. Laramee, J. Schneider & H. Hauser: **Texture-Based Flow Visualization on Isosurfaces from Computational Fluid Dynamics**, Proceedings of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004), May 19-21, 2004, Konstanz, Germany

[MPS05] O. Mallo, R. Peikert, C. Sigg, F. Sadlo, **Illuminated Lines Revisited**, *Proceedings of IEEE Vis 2005, pp. 19-26 (Minneapolis, MN, USA, October 23-28, 2005)*

[PWS06] C. Petz, T. Weinkauf, H. Streckwall, F. Salvatore, B.R. Noack, H.-C. Hege, **Vortex Structures at a Rotating Ship Propeller**, Presented at the 24th Annual Gallery of Fluid Motion exhibit, held at the 59th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Tampa Bay, November 2006

[RA07] S. Roy, and S. Acharya, "Study on Flow and Turbulence Inside a Stirred Tank and Investigation on the Effects of Macroinstability on Trailing Vortex Structures", *ASME International Mechanical Engineering Congress and Exposition*, Seattle, November 2007

[STA98] D. Stalling. "Fast texture-based algorithms for vector field visualization." Dissertation, FU Berlin, Preprint SC-98-58, Konrad-Zuse-Zentrum Berlin (ZIB), December 1998.

[SWH05] D. Stalling and M. Westerhoff and H.-C. Hege, "Amira - an object oriented system for visual data analysis", Visualization Handbook, Christopher R. Johnson and Charles D. Hansen, Academic Press, 2005

[SZH97] D.Stalling, M.Zockler, and H.-C. Hege. **Fast display of illuminated field lines**. In *IEEE Transactions on Visualization and Computer Graphics*, vol.3, pages 118-128, 1997.

[SZH97] D.Stalling, M.Zockler, and H.-C. Hege. **Fast display of illuminated field lines**. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pages 118-128, 1997.

[TRHA07] M. Tyagi, S. Roy, S. Acharya, and A-D Harvey III, "Simulation of laminar and turbulent impeller stirred tanks using immersed boundary method and large eddy simulation technique in multi-block curvilinear geometries", *Chemical Engineering Sciences*, volume 63, issue 5, pages 1351-1363, 2007

[WTH02] T. Weinkauf, H. Theisel, **Curvature Measures of 3D Vector Fields and their Applications**, Journal of WSCG 10(2), WSCG 2002, Plzen, Czech Republic, February 4 - 8, 2002