Evolving Time Surfaces in a Virtual Stirred Tank

Bidur Bohara
Farid Harhad
Department of Computer Science
Louisiana State University
Baton Rouge, LA-70803
bbohar1@tigers.lsu.edu,
fharhad@cct.lsu.edu

Werner Benger
Center for Computation & Technology
Louisiana State University
Baton Rouge, LA-70803
werner@cct.lsu.edu

Nathan Brener
S. Sitharama Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, LA-70803
brener@csc.lsu.edu ,
iyengar@csc.lsu.edu

Marcel Ritter
Department of Computer Science
University of Innsbruck
Technikerstrasse 21a
A-6020 Innsbruck, Austria
marcel.ritter@uibk.ac.at

Kexi Liu, Brygg Ullmer
Center for Computation & Technology
Department of Computer Science
Louisiana State University
Baton Rouge, LA-70803
kliu9@lsu.edu, ullmer@lsu.edu

Nikhil Shetty, Vignesh Natesan,
Carolina Cruz-Neira
Center for Adv. Comp. Studies
University of Louisiana at Lafayette
Lafayette, LA 70504
nikhil.j.shetty@gmail.com

ABSTRACT

The complexity of large scale computational fluid dynamic simulations demand powerful tools to investigate the numerical results. Time surfaces are the natural higher-dimensional extension of time lines, the evolution of a seed line of particles in the flow of a vector field. Adaptive refinement of the evolving surface is mandatory for high quality under reasonable computation times. In contrast to the lower-dimensional time line, there is a new set of refinement criteria that may trigger the refinement of a triangular initial surface, such as based on triangle degeneracy, triangle area, surface curvature etc. In this article we describe the computation of time surfaces for initially spherical surfaces. The evolution of such virtual "bubbles" supports analysis of the mixing quality in a stirred tank CFD simulation. We discuss the performance of various possible refinement algorithms, how to interface alternative software solutions and how to effectively deliver the research to the end-users, involving specially designed hardware representing the algorithmic parameters.

Keywords

visualization, CFD, large data, pathlines, timelines, surface refinement

1. INTRODUCTION

1.1 Motivation

Computational Fluid Dynamics (CFD) is a computationally-based design and analysis technique for the study of fluid flow. CFD can provide high fidelity temporally and spatially resolved numerical data, which can be based on meshes that range from a few million cells to tens of millions of cells. The data from CFD can range to several hundred thousand time steps and be of sizes in order of terabytes.

Therefore, a key challenge here is the ability to easily mine the time dependent CFD data; extract key features of the flow field; display these spatially evolving features in the space-time domain of interest. In this work, we present an interdisciplinary effort to generate and visualize time surfaces of the fluid flow from the time dependent CFD data. The implementation of time surfaces, such as an evolving surface of a sphere, for analyzing the flow field is more relevant in context of a stirred tank system. The integration of surfaces over time generates an evolving surface that can illustrate key flow characteristics such as how matter injected in a stirred tank disperses, and in what regions of the tank is the turbulence high. Such observations are crucial to identifying the best conditions for optimal mixing.

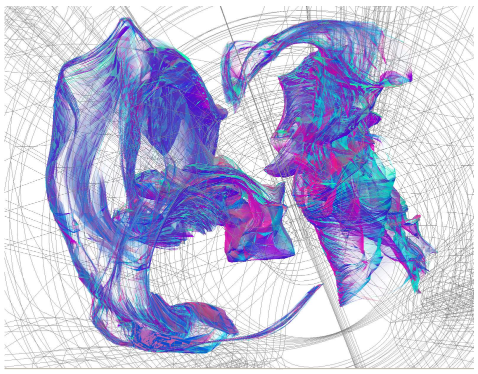

Figure 1: Two evolving spheres visualized just before their mixing in the Stirred Tank simulation system.

The CFD dataset was obtained from a large eddy simulation (LES) of flow inside a stirred tank reactor (STR). The simulation is performed on 200 processors (64 bit 2.33 G Hz Xeon quadcore) where each time-step is calculated in approximately 36 seconds. Stirred tanks are the most commonly used mixing device in chemical and processing industries. Improvements in the design of stirred tanks can translate into several billion dollar annual profit. However, better designs of stirred tanks require detailed understanding of flow and mixing behavior inside the tank. The present study focuses on analyzing the dynamics of mixing inside the tank. Turbulent flow inside the stirred tank was solved numerically using LES to resolve small-scale turbulent fluctuations and the immersed boundary method (IBM) in order to model the rotating impeller blade in the framework of a fixed curvilinear grid representing the tank geometry. The grid is distributed over 2088 blocks and comprised of 3.1 million cells in total. Flow variables like velocity and pressure are defined at the center of each cell and computed for each time step over a total of 5700 time steps representing 25 complete rotations of the impeller. The handling and processing of these voluminous, multi-block, non-uniform curvilinear datasets to generate time surfaces and track set of particles in the fluid flow is the main challenge addressed in this paper.

1.2 Related Work

One of the earliest works related to this problem is the generation of stream surfaces, in particular Hultquist's attempt to generate a triangular mesh representation of streamsurfaces. Hultquist introduced an algorithm that constructs stream surfaces by generating triangular tiles of adjacent streamlines or stream ribbons. In Hultquist's algorithm, tiling is done in a greedy fashion. When forming the next triangle, the shortest leading edge is selected out of the two possible trailing triangles and appended to the ribbon. Each ribbon forming the stream surface is advanced until it is of equivalent length to its neighboring ribbon along the curve they share [13]. Particles are added to the trail of the stream surface by splitting wide ribbons, and particles are removed from the stream surface by merging two narrow (and adjacent) ribbons into one. Note that Hultquist's algorithm was developed for steady flows. Also, advancing the front of the stream surface requires examining all the trailing ribbons.

Along the same lines, Schafhitzel et al [15] adopted the Hultquist criteria to define when particles are removed or added, but they derived a point-based algorithm that is designed for GPU implementation. In addition to rendering a stream surface, they applied line integral convolution to show the flow field patterns along the surface.

Rather than remeshing a stream surface when the surface becomes highly distorted, von Funck et al [23] introduced a new representation of smoke in a flow as a semi transparent surface by adjusting opacity of triangles that get highly distorted and making them fade. Throughout the evolution of the smoke surface, they do not change the mesh, but rather use the optical model of smoke as smoke tends to fade in high divergent areas [23]. However, the authors report that this method does not work well if the seeding structure is a volume structure instead of a line structure.

Core tangibles [21] we use in this paper are physical interaction elements such as Cartouche menus and interaction trays, which serve common roles across a variety of tangible and embedded interfaces. These elements can be integrated to dynamically bind discrete and continuous interactors to various digital behaviors. Many toolkits support low-level tangible user interface design, allowing designers to assemble physical components into hardware prototypes which can be interfaced to software applications using event-based communication. Notable examples include PHidgets [10], Arduino [2], iStuff [1], SmartIts [3] etc. Core tangibles focus on tangible interfaces for visualization, simulation, presentation, and education, often toward collaborative use by scientist end-users [21].

2. MATHEMATICAL BACKGROUND

In the domain of computer graphics one distinguishes four categories of integration lines $q \subset M$ that can be computed from a time-dependent vector field $v \in \mathcal{T}(M)$, mathematically a section of the tangent bundle $T(M)$ on a manifold M describing spacetime: *path lines*, *stream lines*, *streak lines* and *material lines*. Each category represents a different aspect of the vector field:

path lines (also called *trajectories*) follow the evolution of a test particle as it is dragged around by the vector field over time.

stream lines (also called *field lines*) represent the instantaneous direction of the vector field; they are identical to path lines if the vector field is constant over time.

streak lines represent the trace of repeatedly emitted particles from the same location, such as a trail of smoke.

material lines (also called time lines) depict the location of a set of particles, initially positioned along a seed line, under the flow of the vector field.

Each of these lines comes with different characteristics: stream lines and path lines are integration lines that are tangential to the vector field at each point

$$\dot{q} \equiv \frac{d}{ds} q(s) = v(q(s)) \tag{1}$$

Since the underlying differential equation is of first order, the solution is uniquely determined by specifying the initial condition $q(0) = q_0$ by a seed point $q_0 \in M$ in spacetime. Neither stream lines nor path lines can self-intersect (in contrast to e.g. geodesics, which are solutions of a second order differential equation). However, a path line may cross the same spatial location at different times, so the spatial projection of a path line may self-intersect.

In contrast to stream and path lines, streak and material lines are one-dimensional cuts of two-dimensional integration surfaces $S \subset M$, $dim(S) = 2$. This surface is constructed from all integral lines that pass through an event on this initial seed line $q_0(\tau)$:

$$S = \{q : \mathbb{R} \to M, \dot{q}(s) = v(q(s)), q(0) = q_0(\tau)\}$$

The resulting surface contains a natural parametrization $S(s, \tau)$ by the initial seed parameter τ and the integration parameter s. It carries an induced natural coordinate basis of tangential vectors $\{\vec{\partial}_\tau, \vec{\partial}_s\}$, with $\vec{\partial}_s \equiv \dot{q} = v$. For a streak line, the initial seed line $q_0(\tau)$ is timelike as new particles are emitted from the same location over time,

$dq_0(\tau)/dt \neq 0$, for a material line the seed line is spacelike $dq_0(\tau)/dt = 0$, a set of points at the same instant of time. The respective streak/time line is the set of points of the surface $q(t) = S_{t=const.}$ for a constant time. If the integration parameter is chosen to be proportional to the time $s \propto t$, which e.g. is the case when performing Euler steps, then the original seed line parameter τ provides a natural parameter for the resulting lines, i.e. each point along a time line is advanced by the same time difference dt at each integration step.

Refinement of lines by introducing new integration points is mandatory to sustain numerical accuracy of the results. The ideas of the Hultquist algorithm [12] and its improvements by Stalling [17] could be applied also to the spatio-temporal case, however such would result in the requirement to perform timelike interpolation of the vector field. For data sets that are non-equidistant in time such as adaptive mesh refinement data generated from Berger-Oliger schemes[6] finding the right time interval for a given spatial location this becomes non-trivial. For now we refrain from non-equidistant refinement in the temporal direction (such as done in [14]), though this is an option – if not requirement – for future work.

A time surface is the two-dimensional generalization of a time line, a volumetric object in spacetime. The Hultquist algorithm, if applied to a spatio-temporal surface, discusses criteria on refining one edge, whereas here we have a much richer set of possible surface characteristics that may trigger creation or deletion of integration points. Some options are to refine a surface at locations where

- a triangle's edge
- a triangle's area
- a triangle's curvature
- a triangle degeneration ("stretching")

becomes larger than a certain threshold. Section 5.1 reviews our results experimenting with different such criteria.

3. SOLUTION

3.1 Data Model

We use the VISH [4] visualization shell as our implementation platform. It supports the concept of fiber bundles [8] for the data model. The data model consists of seven levels, each of which is comprised of compatible arrays that represent a certain property of the dataset [5]. These levels, which constitute a Bundle, are Slice, Grid, Skeleton, Representation, Field, Fragment and Compound. The Field represents arrays of primitive data types, such as int, double, bool, etc., and the collection of Fields describes the entire Grid. The Grid objects for different time slices are bundled together and are represented as a Bundle. As an example of our implementation, each Field contains values of a property such as coordinates, connectivity information, velocity, etc. The collection of these Fields is a Grid object, and the collection of Grid objects for all time slices is the Bundle of the entire dataset.

The dataset used for visualizing the features of fluid flow contains numerical data for 2088 curvilinear blocks constituting the virtual stirred tank. The input vector field is fragmented and these fragments are the blocks of the Grid. The input dataset for each time slice consists of coordinate location, pressure and fluid velocity for each grid point in the entire 2088 blocks. These properties are stored as Fields in the Grid object for each time slice, and these Grid objects are then combined into a Bundle.

When a multi-block is accessed for the first time, a Uniform-Grid-Mapper is created which is a uniform grid having the same size as a world coordinate aligned bounding box of the multi-block. For each cell of the Uniform-Grid-Mapper a list of curvi linear block cells (indices) is stored which intersect the Uni-Grid-Mapper cell by doing one iteration over all curvilinear grid cells and a fast min/max test. When computing the local multi-block coordinates the corresponding Uni-Grid-Mapper cell is identified first which then selects a small number of curvilinear cells for the Newton iteration. Uni-Grid-Mapper objects are stored in the Grid object of the vector field and can be reused when accessing the same multi-block again later.

3.2 Out of Core Memory Management

The original approach taken while visualizing the features of fluid flow is to keep the entire vector field data in the main memory and integrate over the vector field to extract the features. However, with the necessity of visualizing the time-dependent 3D vector field, the original approach has restrictions, such as the size of the time-dependent data can easily exceed the capacity of main memory of even state of the art workstations. In [24], the authors present the concept of an out-of-core data handling strategy to process the large time dependent dataset by only loading parts of the data at a time and processing it. Two major strategies presented for out-of-core data handling are Block-wise random access and Slice-wise sequential access. The authors emphasize the Slice-wise sequential access strategy for handling the data given in time slices, however, we have implemented both Block-wise access and Slice-wise access of time-dependent data while generating the time surfaces for visualizing the fluid flow.

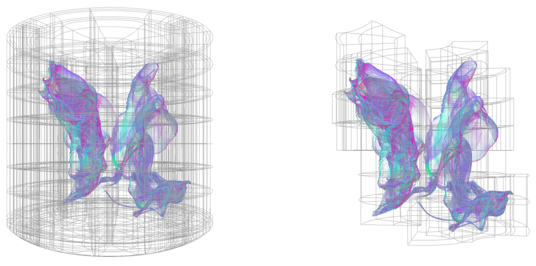

Figure 2: Time surface computed from a vector field given in 2088 fragments (curvilinear blocks) covering the Stirred Tank Grid (left). Only those fragments that affect the evolution of the time surface (right) are actually loaded into memory.

The virtual stirred tank system has 2088 blocks, and each block has vector field data for every time slice. The data for

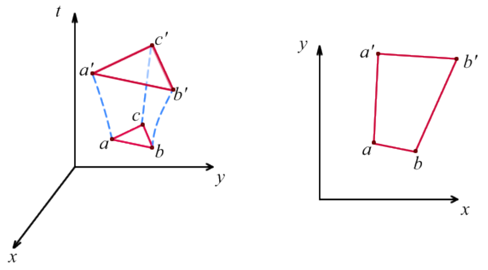

Figure 3: Particle advection of a 2-dimensional element vs. a 1-dimensional element. In our case, our surface element is in 3-dimensional space spanned over time.

each time slice is accessed only once as a Grid object from the input Bundle and processed to generate the time surface at that particular time. The integration of the time surface does not process all the blocks, instead only the blocks that are touched at the given time slice are loaded and processed.

At every time slice two Grid Objects are handled, one containing the input data of the vector field and the other consisting of seed points and connectivity information among the seed points. The connectivity information is used to generate the triangle mesh for surface generation. In the case of no surface refinement, the connectivity information is constant throughout the time slices and is stored once and used multiple times. This conserves the memory and also reduces the memory access. However, with surface refinement the number of points and their connectivity changes over time resulting in an increase in memory usage.

3.3 Particle Seeding and Advection

Our set of particle seeds q_{i,t_0} for $i = 0, ..., n-1$, lie on a spherical sphere. At any given time $t > t_0$, the time surface is represented as as a triangular mesh formed by the particles $q_{i,t}$that have been advected using equation 1. Figure 3 illustrates the difference between our seeding approach versus Hultquist's where we are evolving a surface element (a triangle) over time as opposed to spanning a surface out of a line segment element.

3.4 Triangular Mesh Refinements

As time elapses, the triangular mesh of particles enlarges and twists according to the flow field. To preserve the quality of the mesh, we refine it by adding new particles and advecting them while updating the mesh connectivity. Of the possible refinements criteria mentioned above, we have implemented the following:

edge length , where if the distance between pairwise particles of a triangle are larger than a threshold edge length, we insert a new midpoint and subdivide the triangle accordingly.

triangle area , where if the area of the triangle formed by the new positions of the particle trippet is larger than a threshold area, we insert three midpoints and subdivide the triangle to a new set of four adjacent triangles.

4. ALTERNATIVE APPROACHES

In order to verify and compare our results with other implementations, we also investigate alternative implementations. Paraview [11] is one of the well known and widely used visualization tools in the scientific community. It addresses issues pertaining to the visualization of large scale data-sets using high-performance computing environments. It can be perceived as a framework around the well known Visualization Toolkit (VTK) [16] library. It not only provides a GUI to VTK, but also provides a convenient environment for intuitive visual programming of the visualization pipeline.

Paraview has implicit mechanisms for handling scale, both in terms of data and computation [7]. It achieves this by providing generalized abstractions for parallelization and distribution. Therefore a scientist using Paraview can switch from visualizing smaller data-sets on a his/her desktop to a much larger data-set utilizing a large HPC infrastructure, with minimum effort.

We describe ongoing work and approaches to porting and visualizing the given F5 (fiber-bundle) data-set, as described in 3.1, in Paraview.

4.1 Porting the fiber-bundle (F5) to Paraview

The 500GB fiber-bundle data-set is provided in the F5 format. This format has no native support in Paraview and some form of conversion would be required to utilize the data.

One approach to solve this problem is to use a format converter and separately convert the entire file to a natively supported format. However, this approach causes redundant data and can waste considerable amount of space on the storage disk. An alternative solution is to write a custom reader into Paraview such that the data is read and mapped into internal VTK data-structures. This approach adds an additional computation time into the visualization pipeline and can cause unnecessary slowdown of the visualization process.

An ideal solution would be a combination of the above mentioned approaches such that both space and time optimization can be achieved. Such a solution is possible in our case due to a certain characteristic of the F5 format (explained shortly) and the use of XDMF (eXtensible Data Model and Format) [9] which is supported in Paraview.

An F5 format is characteristically a specific description or organization of the HDF5 data format. All H5 readers and commands which typically work on HDF5 formats also work on F5.

The XDMF data format is an XML format for data generally known as a "light data". It provides light weight descriptions of the "heavy data" which is typically a HDF5 file containing the actual data. A XDMF file can thus be seen as an index into the HDF5 file and is usually much smaller in size, taking very less time to get generated.

Paraview is supplied with the generated XDMF file through which it can access the data in the corresponding HDF5 (or F5) file. No other reader or converter is necessary. An added advantage of this approach is that parallel file readers (if supported) and other parallel algorithms can be used to quickly access and process very large data-sets. We thus leverage on the parallel and distributed framework already provided in Paraview.

4.2 Details of XDMF for F5 fiber-bundle

An XDMF description of the F5 fiber-bundle is shown below.

```
<?xml version="1.0" ?>
<!DOCTYPE Xdmf SYSTEM "Xdmf.dtd" [
<!ENTITY HeavyData "50New.f5">
]>

<Xdmf Version="2.0">
 <Domain>
  <Grid Name="TimeSeries" Type="Temporal">
   <Grid Name="Multiblock" Type="Spatial">
   <Time Value="000000000.0000000000"/>
    <Grid Name="Block00001">
     <Topology TopologyType="3DSMesh"/>
     <Geometry>
       <DataItem Format="HDF">
       &HeavyData;:/f5/path/to/Points/Block00001
       </DataItem>
     </Geometry>
     <Attribute Name="Pressure">
       <DataItem Format="HDF">
       &HeavyData;:/f5/path/to/Pressure/Block00001
       </DataItem>
     </Attribute>
     <Attribute Name="Velocity">
       <DataItem Format="HDF">
       &HeavyData;:/f5/path/to/Velocity/Block00001
       </DataItem>
     </Attribute>
    </Grid>
    <Grid Name="Block00002">
     ...
   </Grid>
  </Grid>
  <Grid Name="TimeSeries" Type="Temporal">
   <Grid Name="Multiblock" Type="Spatial">
   <Time Value="000000001.0000000000"/>
    ....
    ....
   </Grid>
  </Grid>
 </Domain>
</Xdmf>
```

As seen in the description above, the XDMF consists of a collection o f Temporal-Grids which represents each time step. Each Temporal-Grid contains a collection of Spatial-Grids which is a representation of multiblock data. Each multiblock data consists of curvilinear blocks. The data for these blocks are in the HDF5 file specified within DataItems.

As of now, results from the Paraview approach are still pending and subject of further investigation.

5. RESULTS

5.1 Surface Refinement

For the different refinement criteria test cases, we benchmarked our implementation with a 30-timestep subset (85 MB per timestep) of the stirred tank data and on a 64-bit dual core (2GHz each) pentium laptop machine with 4GB of RAM. We advected one sphere for the first 30 timesteps of the simulation. Due to the small size of our test data, we could not notice a difference in time surface meshing quality from the visualization itself, but from the data in tables 1 and 2, we notice a slight performance improvement of the area criteria over the edge length criteria. Though the number of particles is slightly higher in the second case, this suggests that the quality of the surface with the area criterion is better.

threshold	tot points	avg time/slice(sec)	tot time(sec)
0.005	4269	6.480	200.868
0.01	822	1.519	47.1
0.02	258	1.165	36.101

Table 1: Timing Analysis for Edge Length Criteria

threshold	tot points	avg time/slice(sec)	tot time(sec)
0.005	4269	6.864	212.785
0.01	837	1.454	45.08
0.02	258	1.150	35.646

Table 2: Timing Analysis for Triangle Area Criteria

From either tables 1 or 2, picking a threshold too small compared to the characteristic of the triangle being examined, results in maximum refinement, while a large enough threshold leads to no refinement at all.

5.2 Timing Analysis

For the overall integration and refinement of the time surface, we used a larger dataset of size 12GB with 150 timesteps. We ran the implementation on a 64bit quadcore workstation with 64 GB of RAM. We used the edge length criterion with a threshold of 0.01.

time	no. of points	time for slice(sec)	time/point(ms)
0	516	0.4	7.0
50	3468	2.0	5.9
100	15822	7.4	4.8
125	41574	18.8	4.7
150	129939	49.7	4.0

Table 3: Timing Analysis for Threshold=0.01

The listing in the above table is for 12 GB of input data from an initial time of 0 to a final time of 150. Initially the number of points is 516, which increases over time as more points are generated for surface refinement. As the number of points increases, the computation time for the next time slice increases. However, the time per point seems to be slowly decreasing. This may be because more and more points tend to locate in the same block and the data of one block is shared by many points, resulting in less memory access per point.

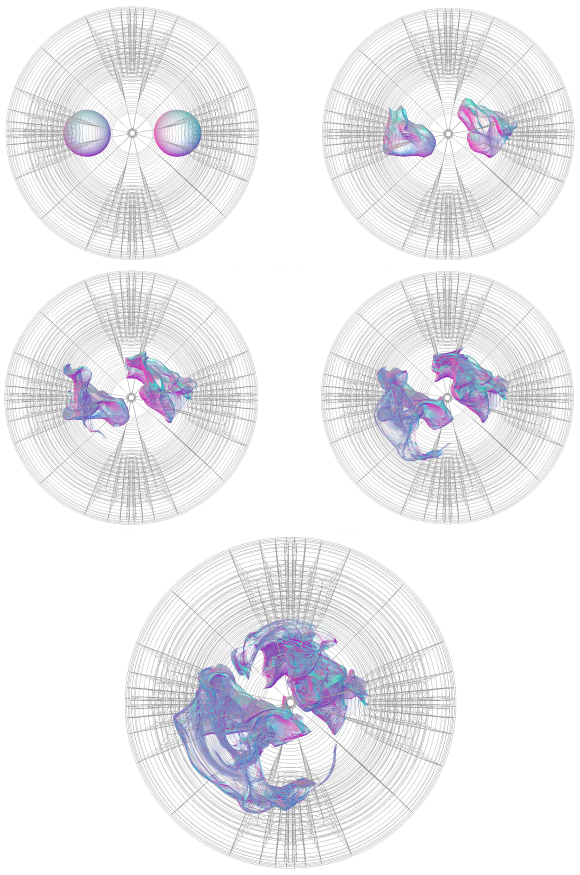

Figure 4: Images showing evolution of two spheres at time slices 0, 50, 100, 125 and 150, respectively from left-top to bottom, as seen top-view of the stirred tank. First image shows the seed spheres, and the last image shows two sphere just before the surfaces are about to mix.

6. DEPLOYMENT TO END USERS

Results of the algorithm can be investigated better if we explore the entire time evolution of the surface interactively, by navigating through space and time. In most visualization environments, the graphical user interface is tightly coupled with the underlying visualization functionality. One feature of VISH is that it decouples the interface from the underlying visualization application. At least in principle, this makes it as easy to couple VISH to a CAVE immersive environment, a web based distributed interface, or physical interaction devices as to the provided traditional 2D graphical use interface. As an example of this, we have based a significant portion of our interaction with the present large dataset stirred tank with "viz tangible" interaction devices. An example of this is pictured in Figure 5. Earlier stages of this work have been described in [22, 20, 19, 18].

An application programming interface (API) is under development which supports coupling our tangibles to VISH and other visualization environments. In this API, when interaction control messages are sent (triggered by physical events, such as RFID entrance/exit or the turning of a knob), they trigger corresponding methods in VISH. We

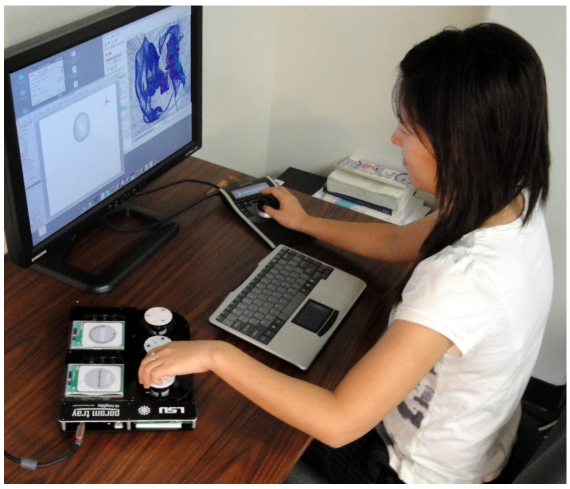

Figure 5: User physically manipulating VISH application through "viz tangibles" interaction devices

use cartouches – RFID-tagged interaction cards [19, 20] – as physical interactors which describe data and operations within the VISH environment. Users can access, explore and manipulate datasets by placing appropriate cartouches on an interaction tray (Figures 5, 6), and making appropriate button presses, wheel rotations, etc.

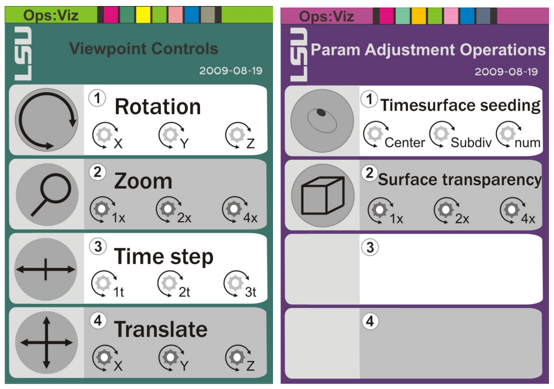

Figure 6: Cartouche cards for viewpoint control and parameter adjustment operations

In our present implementation, we have used two classes of cartouche objects. These are summarized below:

1. *Viewpoint operations:* Specific supported view point controls include rotation, zooming, and translation. In the case of rotation and translation, individual wheels are bounds to the (e.g.) x, y, z axis. In the context of zooming or time step navigation, wheels represent different scales of space and time navigation.
2. *Parameter Adjustment operations:* Our current implementation includes time surface seedings and surface transparency adjustment. For time surface seedings, we steer center of seeds, number of subdivisions, etc.

to parameter wheels. Within surface transparency adjustment, wheels are bounded to different scales of surface transparency.

In future, we hope quantities in high dimensional parameter space such as curvature and torsion of the surface can also be explored effectively with the integration of “viz tangibles” and the API.

7. CONCLUSION

While most of the previous visualization techniques for fluid flow have concentrated on flow streamlines and pathlines, our approach has been directed towards generating the time surfaces of the flow. The interdependencies of integration over a vector field require random access to amounts of data beyond a single workstation's capabilities, while at the same time requiring shared memory for required refinements. This limits available hardware and impacts parallelization efforts. The evolution of a seed surface required refinement of its corresponding triangular mesh to preserve the quality of the time surface over time. From the results we noticed a slight superior quality of the area refinement criterion over the edge length criterion.

8. ACKNOWLEDGMENTS

We thank the VISH development team, among them Georg Ritter, University of Innsbruck, and Hans-Peter Bischof, Rochester Institute of Technology, for their support; Amitava Jana and Sanjay Kodiyalam from Southern University, Baton Rouge for their continued vision for interactive usage in the CAVE VR environment by providing resources to drive further development and optimization of the software environment. This research employed resources of the Center for Computation & Technology at Louisiana State University, 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), NSF MRI-0521559 (Viz Tangibles) and IGERT (NSF Grant DGE-0504507).

9. ADDITIONAL AUTHORS

Additional authors: Sumanta Acharya and Somnath Roy, Department of Mechanical Engineering, at Louisiana State University; `acharya@me.lsu.edu,sroy13@tigers.lsu.edu`

10. REFERENCES

[1] R. Ballagas, M. Ringel, M. Stone, and J. Borchers. iStuff: a physical user interface toolkit for ubiquitous computing environments.

[2] M. Banzi. *Getting Started with Arduino*. Make Books - Imprint of: O'Reilly Media, Sebastopol, CA, 2008.

[3] M. Beigl and H. Gellersen. Smart-its: An embedded platform for smart objects. In *Smart Objects Conference (sOc)*, volume 2003. Citeseer, 2003.

[4] W. Benger, G. Ritter, and R. Heinzl. The Concepts of VISH. In 4^{th} *High-End Visualization Workshop, Obergurgl, Tyrol, Austria, June 18-21, 2007*, pages 26–39. Berlin, Lehmanns Media-LOB.de, 2007.

[5] W. Benger, M. Ritter, S. Acharya, S. Roy, and F. Jijao. Fiberbundle-based visualization of a stir tank fluid. In *WSCG 2009, Plzen*, 2009.

[6] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[7] J. Biddiscombe, B. Geveci, K. Martin, K. Morel, and D. Thompson. Time dependent processing in a parallel pipeline architecture. *IEEE Transactions on Visualization and Computer Graphics*, 13:2007.

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

[9] J. A. Clarke and R. R. Namburu. A distributed computing environment for interdisciplinary applicationsâĂİ, concurrency and computation: Practice and experience vol. 14, grid computing environments special issue. *Currency and Computation: Practice and Experience*, (14):13–15, 2002.

[10] S. Greenberg and C. Fitchett. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 209–218. ACM New York, NY, USA, 2001.

[11] A. HENDERSON. Paraview guide, a parallel visualization application, 2005.

[12] J. P. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *Visualization '92*, pages 171–178. IEEE Computer Society, 1992.

[13] J. P. M. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 171–178, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[14] H. Krishnan, C. Garth, and K. I. Joy. Time and streak surfaces for flow visualization in large time-varying data sets. *Proceedings of IEEE Visualization '09*, Oct. 2009.

[15] T. Schafhitzel, E. Tejada, D. Weiskopf, and T. Ertl. Point-based stream surfaces and path surfaces. In *GI '07: Proceedings of Graphics Interface 2007*, pages 289–296, New York, NY, USA, 2007. ACM.

[16] W. Schroeder, K. Martin, and W. Lorensen. The visualization toolkit: An object oriented approach to 3d graphics, 1996.

[17] D. Stalling. *Fast Texture-Based Algorithms for Vector Field Visualization*. PhD thesis, Free University Berlin, 1998.

[18] C. Toole, B. Ullmer, R. Sankaran, K. Liu, S. Jandhyala, C. W. Branton, and A. Hutanu. Tangible interfaces for manipulating distributed scientific visualization applications. *Submitted to Proc. of TEI'10*, 2010.

[19] B. Ullmer, Z. Dever, R. Sankaran, C. Toole, C. Freeman, B. Casady, C. Wiley, M. Diabi, A. J. Wallace, M. Delatin, B. Tregre, K. Liu, S. Jandhyala, R. Kooima, C. W. Branton, and R. Parker. Cartouche: conventions for tangibles bridging diverse interactive systems. *Submitted to Proc. of TEI'10*, 2010.

[20] B. Ullmer, A. Hutanu, W. Benger, and H.-C. Hege. Emerging tangible interfaces for facilitating collaborative immersive visualizations. *NSF Lake Tahoe Workshop on Collaborative Virtual Reality and Visualization*, 2003.

[21] B. Ullmer, R. Sankaran, S. Jandhyala, B. Tregre, C. Toole, K. Kallakuri, C. Laan, M. Hess, F. Harhad, U. Wiggins, et al. Tangible menus and interaction trays: core tangibles for common physical/digital activities. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 209–212. ACM New York, NY, USA, 2008.

[22] B. Ullmer, R. Sankaran, S. Jandhyala, B. Tregre, C. Toole, K. Kallakuri, C. Laan, M. Hess, F. Harhad, U. Wiggins, and S. Sun. Tangible menus and interaction trays: core tangibles for common physical/digital activities. In *Proc. of TEI '08*, pages 209–212, 2008.

[23] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2008)*, 14(6):1396–1403, November - December 2008.

[24] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Feature flow fields in out-of-core settings. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 51–64. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.