Subsampling-Based Compression and Flow Visualization

Alexy Agranovsky[a,b], David Camp[b], Kenneth I. Joy[a], and Hank Childs[b,c]

[a]University of California, Davis, 1 Shields Avenue, Davis, CA. USA
[b]Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA. USA
[c]University of Oregon, 1585 E 13th Avenue, Eugene, OR. USA

ABSTRACT

As computational capabilities increasingly outpace disk speeds on leading supercomputers, scientists will, in turn, be increasingly unable to save their simulation data at its native resolution. One solution to this problem is to compress these data sets as they are generated and visualize the compressed results afterwards. We explore this approach, specifically subsampling velocity data and the resulting errors for particle advection-based flow visualization. We compare three techniques: random selection of subsamples, selection at regular locations corresponding to multi-resolution reduction, and introduce a novel technique for informed selection of subsamples. Furthermore, we explore an adaptive system which exchanges the subsampling budget over parallel tasks, to ensure that subsampling occurs at the highest rate in the areas that need it most. We perform supercomputing runs to measure the effectiveness of the selection and adaptation techniques. Overall, we find that adaptation is very effective, and, among selection techniques, our informed selection provides the most accurate results, followed by the multi-resolution selection, and with the worst accuracy coming from random subsamples.

Keywords: In Situ, Sub-Sampling, Compression, Adaptive Budget, Flow Visualization

1. INTRODUCTION

To allow supercomputers to become ever larger, hardware architects must make tradeoffs to optimize overall machine cost, computational ability, and power consumption. Increasingly, these architects are selecting disks with I/O rates that are decreasing relative to their machine's computational power. Saying it another way, disk speeds are rising on supercomputers, but they are not rising nearly as fast as their ability to generate data.[1] As leading-edge machines march towards exascale computing (i.e., calculating 10^{18} floating point operations per second), disks will become progressively incapable of keeping pace with the simulation, partly because of the prohibitive financial costs for disks that can maintain sufficient bandwidth and partly because of the prohibitive power costs for moving full resolution simulation data from CPU memory to disk.[2]

These hardware constraints create new challenges for visualization. Specifically, the traditional approach where simulations save their full-resolution data to disk, which visualization software then reads — so-called post-processing — is incongruent with the data movement challenges upcoming in high-performance computing. One solution to this problem is to process the data *in situ*, i.e., having the visualization algorithms run at the same time as the simulation, and producing results without involving the disk. This approach is predicted to play a large role in the future,[3] and there have been many successful examples recently. That said, *in situ* processing works best when end users know what they want to visualize *a priori*. Unfortunately, when users explore data — a very common visualization use case — they do not know what they want to visualize ahead of time. This makes explorative visualization with *in situ* processing problematic to carry out in real-time, since it requires the end user to be actively involved while the simulation runs, likely slowing down the simulation and causing the supercomputer to be used inefficiently.

Researchers have been studying an alternative to real-time explorative visualization with *in situ* processing, which is to use *in situ* processing only to compress the data set, and then writing this compressed data set to the disk, where the user can explore the data in the traditional post-processing setting.[4] This is clearly not ideal for simulation scientists, who would

Further author information: (Send correspondence to A.A.)
A.A.: E-mail: aagranovsky@ucdavis.edu
D.C.: E-mail: dcamp@lbl.gov
K.I.J.: E-mail: kijoy@ucdavis.edu
H.C.: E-mail: hchilds@uoregon.edu

obviously prefer visualizations that reflect the full resolution data. But the constraints upcoming in data movement will not allow for this ideal scenario; simulation scientists will be forced to make compromises. Therefore the focus shifts to the tension between achieving the highest compression rates possible and producing the most accurate answer. Mandating very hire compression rates will often erode accuracy, while mandating very high accuracy will often limit the potential for compression.

An important class of visualization techniques for many scientists is that of flow visualization. Flow visualization encompasses many techniques, many of which depend on particle advection.[5] A particle advection operation seeds a massless particle at an initial location and displaces the particle along a trajectory that is tangent to the velocity field at every point. Particle advection is especially sensitive to compression, since the errors introduced by the compression process can cause the particle to displace to the wrong location. Initial errors will likely be subtle, with the particle trajectory only diverging slightly away from the ground truth. But these errors can accumulate, possibly leading the particle to end up at a vastly different end location. Note that many visualization techniques do not share this problem; with isosurfacing, for example, error from compression will always remain local to those cells that differ from ground truth. That said, scientists are familiar with sensitive results for particle advection — even without compression — since slight changes in seed location can lead to dramatically different trajectories. As a result, many scientists already employ practices to understand the stability of flow visualizations, such as placing a cluster of seeds around a location and checking for consistent behavior.

Data compression techniques transform data to new forms that use fewer bits than its original representation. There are many approaches for compression, some of which introduce no error ("lossless"), and some of which do introduce error ("lossy"). In this work, we explore subsampling-based compression, a lossy technique. With subsampling, vectors are selected at locations throughout the volume to form a basis of the velocity field. The particle advection operation depends on evaluating the velocity at potentially any location in the volume, requiring interpolation from the basis velocity vectors. It is this interpolation that introduces the error in the advection process.

A propitious property of subsampling is that it produces a natural "knob" to balance between compression rate and data integrity. Selecting a basis with more vectors will increase accuracy and have less compression, while a basis with less vectors will decrease accuracy but have higher compression. The research in this paper considers this selection process. Assuming a simulation scientist fixes a specific budget for storage and assuming the simulation scientist decides to use a subsampling-based compression to meet this budget, then the work aims at answering the following question: what are the best methods for choosing a basis that maximizes accuracy?

The contributions of this paper are:

- A novel method for informed selection of basis vectors to minimize error.
- A method for adapting subsampling budget across processors in a parallel run, including a technique for adapting the budget based on vector field complexity.
- A study exploring the effects of subsampling-based compression on the accuracy of particle advection, including a comparison of multiple methods for subsampling and adaptation.

In §3, we present an overview of the technique, which includes a description of our informed picking in §3.2.3. In §4, we present an overview of our study, and, in §5, we describe our results.

2. RELATED WORK

We are not aware of any previous work directly in the area of subsampling-based compression for flow visualization. However, there have been many related efforts, including work on topology-based compression for flow visualization (§2.1), preprocessing data *in situ* for subsequent exploratory visualization (§2.2), general compression techniques visualization (§2.3), and selecting important subregions of vector fields (§2.4).

2.1 Topology-Based Compression for Flow Visualization

There have been several efforts on topology-based compression for flow visualization. Lodha et al. found critical points and similarly provided users a "knob" to control compression.[6] Theisel et al. collapsed critical points and reduced the problem to mesh reduction.[7] Later, Theisel et al. continued their work and provided a threshold to distinguish important features for filtering.[8]

Of all the previous work considered, these efforts are the most similar to our own, especially with respect to methodologies for testing and evaluation. However, none of these works considered the problem of running *in situ*, which creates additional constraints for execution time and memory overhead. Two of the works reported that their techniques took over five minutes to run, and the third work did not report timing information at all. These long durations would not be viable in an *in situ* environment. Further, advances in computing power since these works were published will not change this situation, since they were looking at very small data sets, and would be slower still for the data sets considered in this effort.

Our own work contrasts with these techniques in that it is appropriate for the constraints of the *in situ* environment.

2.2 Using *In Situ* Techniques to Reduce Data for Post-processing

Although *in situ* techniques have been used for some time, they have been primarily used for cases with *a priori* knowledge about what to visualize. There are only a handful of instances of using *in situ* techniques to compress data *in situ* for subsequent explorative post-processing. In one such study, Tikhonova et al. transformed (and compressed) large scale simulation data to enable exploration through volume rendering.[9] In another study, Wang et al. examined flow fields, calculating the uniqueness of each block in a volume and then creating an importance field that characterizes the flow field.[10] Finally, Wang et al. explored a related direction, this time on combustion data, by using importance fields to guide compression in a manner that does not deter from quality rendering.[11]

2.3 General Compression in Visualization

Multi-resolution techniques, especially wavelet-based multiresolution, are well studied in visualization.[12,13] The approach is so popular that the VAPOR visualization tool[14] is designed entirely around the benefits of wavelet compression.

Compression can be more broadly defined to include transforms that reduce data. Techniques such as data subsetting[15–19] and feature identification and tracking[20–23] have also been well studied.

2.4 Informed Selection of the Vector Basis

Our method for informed selection of the basis vectors follows the intuition that vectors in self-similar regions should contribute less samples and those in varied areas should contribute more samples. This intuition reflects the results from several previous works. Chen et al. also consider methods for picking (seed points) that depend on similarity.[24] Xu et al. consider the contribution of each vector from the perspective of an information-theoretic framework.[25] And Chen et al. consider the effects of editing the vector field for Morse decomposition.[26]

3. TECHNIQUE

3.1 Overview

Our method consists of two primary phases. We assume that the budget for how many subsamples can be selected is specified by the simulation scientist prior to the subsampling.

The first phase is to perform vector subsampling. In a parallel setting, we begin this phase by identifying which processors have the most complex vector fields and adapting our overall budget so those processors can store more subsamples (§3.3). We then select the subsamples, using one of three techniques (§3.2). The subsamples we select are the only data written to disk. Although the budget adaptation occurs before the selection, we present the selection first, since concepts key to the informed selection algorithm inform strategies behind budget adaptation.

The second phase is to visualize the data with particle advection techniques, as part of a post-processing visual exploration. Users select particles and our software evaluates their trajectory, i.e., where these particles advect to. We support streamlines and pathlines, but any particle advection-based flow visualization could be supported. We use the fourth order Runge-Kutta (RK4) technique to advance a particle. This technique depends on evaluating the vector field at arbitrary locations; the details of the interpolating the vector field from the subsamples is described in §3.4.

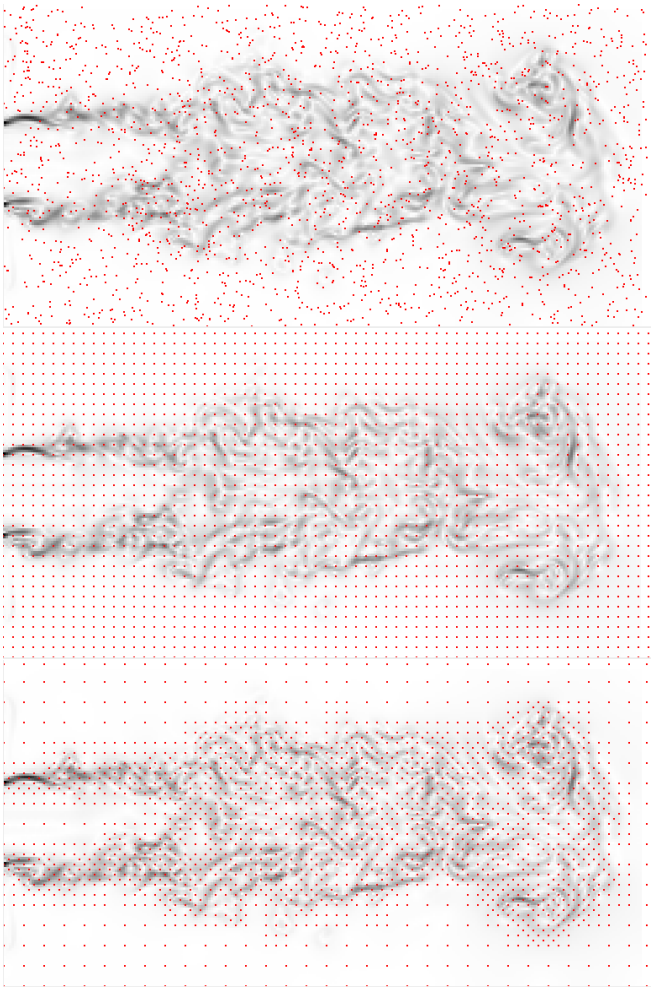

Figure 1. These three images show the placement of subsamples (colored in red) for a two-dimensional steady-state jet data set and 4X compression. The top image is of the *Random* technique (§3.2.1), the middle image is of the *Uniform* technique (§3.2.2), and the bottom image is of our *Informed Selection* technique (§3.2.3). The turbulence of the flow field is colored in grayscale for context, with non-turbulent areas colored white and turbulent regions colored black.

3.2 Selection Techniques

We study three selection techniques: *Random* (§3.2.1), *Uniform* (§3.2.2), and *Informed Selection* (§3.2.3). In all cases, we take the budget specified by the simulation scientist (of how much data can be stored to disk) and translate this to a number of vectors to extract from the volume. An example of the subsamples selected by each of the three techniques is plotted in Figure 1.

3.2.1 Random

The *Random* technique chooses subsamples (i.e., vectors) at random locations throughout the volume. The locations are not constrained to the mesh's grid points, and rarely overlap with these grid points.

3.2.2 Uniform

The *Uniform* technique chooses subsamples at regular locations throughout the volume. In special cases, this technique is similar to multi-resolution sampling. For example, if the specified budget for a two-dimensional data set was to select one subsample for every four grid points, then the resulting subsampling would effectively be at a resolution two times coarser in both dimensions. For this reason, we consider the *Uniform* technique to be a surrogate for comparisons with multi-resolution methods, which is also used for compression (§2.3). However, we note that, for multi-resolution methods,

the values for coarse refinement levels are often selected to represent all the values in the finer grid, where *Uniform* simply chooses one of the values from the finer grids. We believe that the difference in error between *Uniform* and a true multi-resolution method is negligible.

3.2.3 Informed Selection

The *Informed Selection* technique chooses subsamples by taking cues from the underlying flow field. The algorithm targets areas experiencing turbulence and rotation, increasing its sampling budget in those regions.

The algorithm has two primary phases: it first calculates the deformation at each point in space (*"Detecting Non-Laminar Flow"*), and then chooses vectors based on assessments of this deformation (*"Choosing Vectors"*).

Detecting Non-Laminar Flow: the goal of this phase is to assess the amount of turbulence and rotation throughout the volume, specifically by creating a scalar value at each point to denote the flow metrics at its location.

To analyze local changes within the flow field, we introduce second-order tensors, or matrices, which hold the scalar values acting on the vector-spaces. These values are responsible for the linear deformations exhibited along all axes. A 3×3 matrix/tensor $\mathbf{A} \in \mathbb{R}^{3\times3}$ is a linear mapping between vectors $\mathbf{v}$ and $\mathbf{w} \in \mathbb{R}^3$. Within the vector field, we calculate the so called *velocity gradient tensors* $\nabla\mathbf{v}$ of a flow field $f : \mathbb{R}^3 \to \mathbb{R}^3$ which describes a local linearized rate of change in velocity:

$$\nabla\mathbf{v} = \begin{pmatrix} \frac{\delta v_x}{\delta x} & \frac{\delta v_x}{\delta y} & \frac{\delta v_x}{\delta z} \\ \frac{\delta v_y}{\delta x} & \frac{\delta v_y}{\delta y} & \frac{\delta v_y}{\delta z} \\ \frac{\delta v_z}{\delta x} & \frac{\delta v_z}{\delta y} & \frac{\delta v_z}{\delta z} \end{pmatrix}$$

where $\nabla\mathbf{v}$ reveals information on how the velocity is changing in space. The characteristics of this tensor provides important deformation descriptors such as magnitude and direction.

To analyze the effects of the instantaneous change in velocity in the region, we consider the volumetric neighborhood within it, represented by a coordinate frame set using the identity matrix, $\mathbf{I}$. As a volume, this coordinate frame represents the three major axes of a sphere. Mapping the velocity gradient onto this volume, $(\mathbf{I} + \nabla\mathbf{v})$, the neighborhood is deformed according to the effects of the flow field. This would be analogous to deforming a spherical volume in the directions of the instantaneous changes in velocity.

To quantify the spatial changes given by the velocity gradient tensor, we take a closer look at the *eigenvectors* and *eigenvalues* of the deformed volume, which give the maximal and minimal directions and magnitude, providing principle components of the deformation. When looking for turbulence, the deformation may be categorized by calculating the *fractional anisotropy* (FA), which approximates how line-like a neighborhood becomes after a distortion. The FA is a scalar value between zero and one that describes the degree of anisotropy during the deformation process, giving an idea as to how line-like a neighborhood has become after distortion. A fractional anisotropy of zero would be a perfect sphere, while an FA of one would cause the ellipse to degenerate into a line. The equation for calculating the fractional anisotropy is

$$FA(\mathbf{T}) = \sqrt{\frac{3}{2}} \frac{\sqrt{(\lambda_1 - tr(\mathbf{T}))^2 + (\lambda_2 - tr(\mathbf{T}))^2 + (\lambda_3 - tr(\mathbf{T}))^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \tag{1}$$

where $tr(\mathbf{T})$ is the normalized trace of the accumulated deformation gradient tensor and the λ's are the eigenvalues of the same tensor.

To examine local changes to the flow field due to rotation, we take a look back at the velocity gradient tensor. Decomposing ∇v into its anti-symmetric component describes the deformation due to rotation, which because the rotational tensor has only three independent elements, may be expressed as a vector

$$\Omega = \begin{pmatrix} \Omega_{23} \\ \Omega_{13} \\ \Omega_{12} \end{pmatrix} = \begin{pmatrix} \frac{\delta v_z}{\delta y} - \frac{\delta v_y}{\delta z} \\ \frac{\delta v_x}{\delta z} - \frac{\delta v_z}{\delta x} \\ \frac{\delta v_y}{\delta x} - \frac{\delta v_x}{\delta y} \end{pmatrix} \tag{2}$$

which is commonly referred to as the *vorticity* or the tendency of particles in a fluid to rotate. The length of this vector gives the angular velocity of the rotation.

The scaling provided by the fractional anisotropy and the magnitude of the vorticity allows for the identification of velocity vectors residing in regions of non-laminar flow. Given a constraint on the number of vectors that may be chosen, the FA and vorticity guides the vector selection in such a way that samples more densely in areas of increased fluid flow effects.

Choosing Vectors All velocity vectors now have a measure of turbulence and vorticity, but, before vectors can be chosen according to these metrics, coverage must be guaranteed throughout the data set, even in laminar regions. Therefore, from the overall budget B, a fraction is allocated towards uniformly subsampling the entire space. This fraction is defined as one uniform resolution coarser than that needed to maximally cover the entire domain given the budget constraint. The following equation is used to calculate the aforementioned maximum grid resolution:

$$R_M = ceil\left(\frac{\log_2 \frac{Dim_x \times Dim_y \times Dim_z}{B}}{k}\right) \tag{3}$$

where k is the number of dimensions and Dim_x, Dim_y, and Dim_z are the sizes of each respective dimension. The amount of velocity vectors needed to uniformly cover the space using our selection method is calculated using R_M:

$$FR = \frac{Dim_x}{R_M+1} \times \frac{Dim_y}{R_M+1} \times \frac{Dim_z}{R_M+1} \tag{4}$$

where the resolution grid one compression level higher, R_C, is used to calculate the fraction of the budget associated with general space coverage. The remaining number of velocity vectors, $B - FR$ will be distributed using the turbulence and rotation metrics described earlier.

The continuation of the informed selection algorithm involves choosing velocity vectors in areas of increased fluid flow effect. The process begins by examining the fractional anisotropy and vorticity of all vectors within the cells of grid R_C. For both metrics, if the average within the cell is higher than that of the average fractional anisotropy or vorticity across all vectors, then the cell is refined one level of resolution higher creating cells akin in size to those of R_M.

The formulation of grid R_C assures that at this stage of the selection process, velocity vectors are still available from the constrained budget. The refined cells are sorted in order by highest fractional anisotropy and vorticity value and a velocity vector is chosen in the middle of the cell, going down both lists from highest to lowest, until the budget is exhausted. If a budget still remains, vectors with high turbulence and/or high rotation are chosen regardless of position as needed. At the conclusion of the informed selection algorithm, areas of non-laminar flow are sampled densely while non-turbulent areas are sampled sparsely yet uniformly to ensure coverage.

3.3 Adaptive Budget Selection

When working on a distributed system, the overall budget B must be dispersed among all tasks, as it limits the total amount of velocity vectors chosen. The simplest approach would be to split the budget evenly, giving each task an equal allowance of vectors to choose. However, depending on the area covered by an individual task, evenly splitting the budget may result in over-sampling in areas of laminar flow and under-sampling in areas of increased turbulence and rotation. Therefore, an adaptive budget may be preferred to limit the over-sampling and increase selection availability in areas of interest.

The insight of fluid flow effects gained through the Informed Selection technique may also be used to vary the budget between tasks. During the selection process, cells are refined one resolution higher according to turbulence and vorticity. Any remaining budget is then focused on the center points of these cells and any other velocity vectors in areas of high

deformation. This aforementioned remaining budget can be evenly or adaptively spread among the tasks, using the number of cells refined within each task as a starting point. However, if no cells have been refined, then the task will receive only enough of the budget to allow for the general sub-sampling of the space used to evenly sample the area.

Before the budget can be distributed between tasks that have had cell refinements, it must be adjusted to compensate for the number of already required vectors. This required budget B_{req} includes the velocities chosen to generally sample the entire data set and those that make up the refined cells within each task. An additional estimation must be made before B_{rem}, the amount of budget remaining for adaptive selection, is calculated. Recall that the step of the Informed Selection algorithm after initial cell refinement is to choose the velocity vector at the center of the refined cells. If the number of refined cells over the entire data set, CR_{total}, is greater than $B - B_{req}$, then there is not enough budget left to put a vector at the center of each cell. Therefore, the remaining budget becomes $B_{rem} = B - B_{req}$ and the adaptive budget per task is calculated as a fraction of the number of refined cells that task owns,

$$B^i = B_{rem} * (\frac{CR^i}{CR_{total}}) + B^i_{req} \tag{5}$$

where B^i is the budget allocated for task i, CR^i is the number of refined cells in task i, and B^i is the budget task i requires.

If, however, there is enough budget remaining to select a velocity vector for each refined cell, then the remaining budget becomes $B_{rem} = B - (B_{req} + CR_{total})$ and the adaptive budget per task is to be primarily based on fluid flow effects. Now that it is assured that every task will have enough of the budget to fill the refined cells, the remaining velocity vectors within each will be chosen in areas of high rotation and high turbulence. To better distinguish large values of fractional anisotropy and vorticity among the tasks, the average FA and $VORT$ in task i are cubed and a summation is formed from all tasks. The average of the two metrics will determine the amount of budget received:

$$B^i = B_{rem} * \left[0.5 * \frac{FA^i}{FA_{total}} + \frac{VORT^i}{VORT_{total}}\right] + B^i_{req} + CR^i \tag{6}$$

where FA^i and $VORT^i$ are the average fractional anisotropy and vorticity cubed, and FA_{tota} and $VORT_{tota}$ are the cubed amounts summed over all tasks. The individual budget also includes the amount task i requires plus the number needed to choose a vector at the center of each refined cell.

The motivation behind two different adaptive schemes is to be able to target areas of interest while maintaining a general covering of the those same areas. If the remaining budget is small, it is more advantageous to cover a larger area. However, if this type of coverage is available with a large budget, the focus turns to better capturing high valued deformations within the flow field. This adaptation of the budget can be applied to the Informed Selection method, as well as the Random method, increasing the utility of both. While it may be applied to the Uniform method as well, the uniformity constraint may lead to a severe under utilization of the allotted budget and is therefore not recommended.

3.4 Interpolation

Accurate interpolation is critical for the success of the technique. Subsampling the vector field already introduces error; if the interpolation method introduces undue error, it could erode the efficacy of the sampling. Further, if interpolation problems are severe, then they would prevent acceptable accuracy even if there was no compression (i.e., if the subsampling selected all the vectors in the data set as the basis). We considered multiple interpolation techniques:

- Shepard's Method - A form of inverse distance weighting interpolation, Shepard's method analyzes all velocity vectors in the global space and weights individual vector information according to its distance from the particle position.
- Moving Least Squares Interpolation - Commonly referred to as MLS, Moving Least Squares builds a local interpolation neighborhood around the particle position, offering continuous reconstruction.
- Barycentric Coordinate Interpolation - Forming a convex hull around the particle, this method interpolates the velocity vector values that form the bounding triangle or tetrahedron in 2D and 3D respectively.

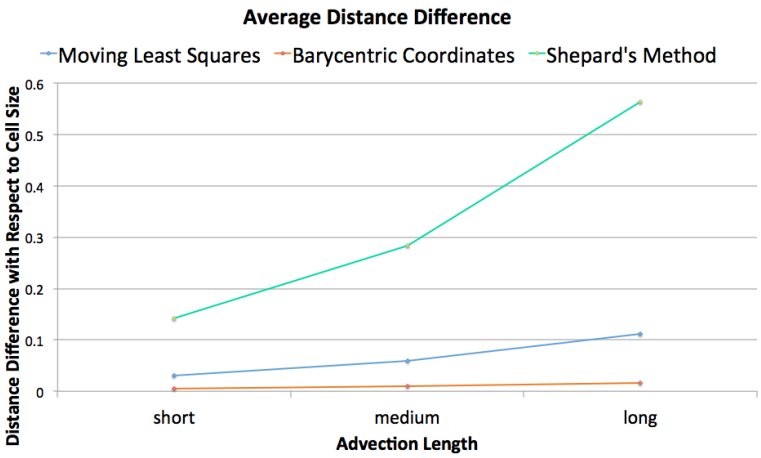

Figure 2. This figure shows an accuracy comparison between the three interpolation methods: Shepard's Method, Moving Least Squares interpolation, and interpolation over barycentric coordinates. The accuracy measurement considers the average distance between particle positions along advected path-lines in comparison to a trajectory advected using all available information. The average distance is then shown with respect to the cell size of the base data resolution.

To further examine the varying interpolation strategies, all methods were given an identical field of velocity vectors, chosen by our informed picking technique with a budget totaling a quarter of the vectors provided for the data set. As a test of interpolation accuracy, each method was used on the same starting particle positions within a flow field, where the particles were advected for an identical length of time: short, medium, and long. Each of the approximated pathlines was then compared point-wise to a ground truth trajectory computed using traditional bilinear interpolation over the full resolution of the data set. The final extracted metric is the average distance between all interpolated particles and ground truth particles, divided by the cell size of the base data resolution, graphed in Figure 2. From the comparison chart, we conclude that barycentric coordinate interpolation is the top choice among the three methods, providing the most accurate results. The lower number of velocity vectors used in this method minimizes the smoothing that occurs with MLS and Shepard's interpolation. For the purposes of quickly finding a bounding tetrahedron, we have used the Computational Geometry Algorithms Library (CGAL)[27] to perform a Delaunay triangulation on the field of velocity vectors prior to interpolation.

As a result of this pre-study, we opted to use barycentric coordinate interpolation, as it contributes the least error of the interpolation methods we studied.

4. STUDY OVERVIEW

4.1 Configurations

Our study was designed to provide coverage over a variety of configurations. We varied three factors:

1. Reduction Factor (3 options)
2. Data Set (4 options)
3. Selection Techniques (5 options)

We ran the cross-product, meaning $3 \times 4 \times 5 = 60$ tests overall. The variants for each factor are discussed below.

4.1.1 Reduction Factor

We considered three reduction factors: 8X, 64X, and 512X.

4.1.2 Data Set

We considered four simulation data sets: a star exploding, thermal hydraulics, flow around a tokamak, and jet flow in a box. All data sets were on rectilinear grids, with the jet flow's size at $128 \times 256 \times 128$, and the rest with a size of 256^3.

4.1.3 Selection Techniques

We use the three selection techniques described in §3.2: *Random* (§3.2.1), *Uniform* (§3.2.2), and *Informed Selection* (§3.2.3). Again, an example of the subsamples selected by each of the three techniques is plotted in Figure 1. We also employ adaptive budgeting (§3.3) to two of the schemes, to create *Informed Selection (adaptive)* and *Random (adaptive)* variants. Thus the total number of selection techniques is give.

4.2 Runtime Environments

We ran this on 64 processors on NERSC's Hopper machine. Each data set was decomposed evenly over processors, meaning each processor operated on a 64^3 piece of the overall data set.

4.3 Measurements

To measure accuracy, we placed a seed in every cell of the data set. For each seed, we measured the distance traveled by the particle and the final location, both for the compressed vector field and for the original vector field. We also measured the memory and execution time for the compression operations, to understand impacts in *in situ* environments.

4.4 Error Evaluation

Our error metric, E, measures how closely the end positions match when advecting a seed point using both the original vector field and the compressed vector field. Seeing as error accumulates throughout the advection process, the end positions are good candidates for comparison. The distance between the end positions is normalized by the distance traveled by the particle. The motivation here is that users will tolerate more error in end position if the particle has traveled further. If a particle travels 100 units, and the end points are 1 apart, then users will likely find such results useful. For this case, our E would be $1 \div 100 = 0.01$. However, if the particles travel only 1 unit, and the end points are 0.5 apart, then the users would likely deem the result misleading, since, proportionally, the result was quite far off. For this case, our E would be $0.5 \div 1 = 0.5$.

Formally, let:

- S denote a seed location,
- D denote the duration to advect,
- V denote the original vector field,
- $\overline{V}$ denote the vector field after a compression process,
- $Advect(a,b,c)$ denote a function that returns the position resulting from advecting a seed a for duration b using vector field c,
- $Pathlength(a,b,c)$ denote a function that returns the total length traveled from advecting a seed a for duration b using vector field c, and
- $Dist(a,b)$ denote a function that returns the Euclidean distance between points a and b.

Then we define our error metric, E, as:

$$E(S,D,\overline{V},V) = \frac{Dist(Advect(S,D,\overline{V}),Advect(S,D,V))}{Pathlength(S,D,V)} \tag{7}$$

The focus on end position has the potential to overlook the importance of the entire trajectory. For example, if the trajectory from the compressed data ends at the same location, but takes a different route to get there, then end users likely would object. However, our study looks at multiple time durations. So if the trajectories happen to coincide after a given duration (but not during the time leading up), it will still be penalized when the shorter durations are considered.

$PathLength(S,D,V)$ is always non-negative, but combinations of S, D, and V that return 0 — corresponding to a zero-length path, which will occur when a seed is placed at a 0-velocity location — are problematic and we omit such points from analysis. It is also possible to have (S, D, V, $\overline{V}$) tuples that cause E to be greater than 1. For example, if V and $\overline{V}$ had totally divergent directions at S, then the particles would travel in opposite directions and the distance between their end points could be greater than the length of the trajectory for S from V. These cases are rare (approximately one in ten thousand) and we cap the E value for such cases at 1.

Therefore, E has a range from 0 to 1. 0 corresponds to an exact match of end points and 1 corresponds to the end points being very far away. Further, a compression method can be evaluated by looking at statistics of E. For our purposes, the best compression method is the one that has E values that skew more towards 0 than the other compression methods.

5. RESULTS

5.1 Comparing Error Across Experiments

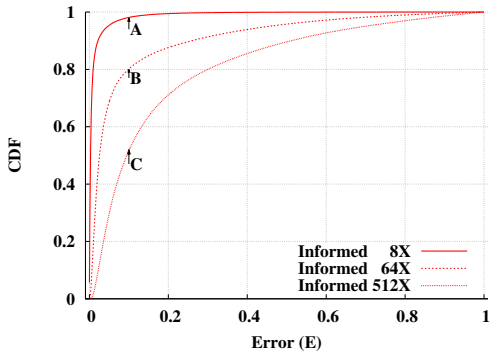

Figure 3. This sample chart illustrates the focus of the error metric E on analyzing accuracy across the compression levels. A vertical cross section of a compression curve defines what percentage of particles are within a desired accuracy range. In this figure, labels A, B, and, C mark a 10% distance error ($E = 0.1$) with 98% of the particles for 8X compression, 80% for 64X, and 53% for 512X being within the range.

We used the error metric E (§4.4) to assess accuracy across the various selection techniques. We now refer the reader to Figure 3, which assists in explaining how we measure error. This figure plots the results for only three compression levels, and using only the *Informed Selection* method. The x-axis is over our error metric, E, which represents an error percentage with values ranging from 0 to 1. The y-axis is a cumulative distribution function (CDF) of the particles. This tells us what proportion of the particles have a given value of E or less. Saying it another way, by drawing a vertical line from any point on the x-axis, the intersection with the curve shows what percentage of particles are within that accuracy range. The labels A, B, and C in Figure 3 are points on each of the different levels of compression, 8X, 64X, and 512X, representing a 10% error. Point A tells us that 98% of the particles are within a 10% error when using 8X compression, while point B shows 80% for 64X, and point C marks 53% for 512X. We can see in Figure 3 that higher compression rates lead to a lower amount of particles within a specified error bound, which is an expected result as we have less data to draw from during particle advection.

5.2 Analyzing Accuracy Achieved In a Variety of Configurations

The results of our study — all 60 tests — are plotted in Figure 4. Each figure of the image represents a different data set, measuring accuracy for advection. Within each chart, we compare the various picking techniques at varying levels of compression.

Comparison of Average Error Distance for all Compressions	Astro	Thermal	Jet	Fusion	Average
Informed (A) is X more accurate than Informed (N-A)	24.98%	11.43%	27.89%	21.43%	21.43%
Informed (A) is X more accurate than Uniform (N-A)	41.85%	34.80%	36.01%	53.78%	41.61%
Uniform (N-A) is X more accurate than Random (A)	63.69%	14.63%	55.19%	-11.41%	30.52%
Random (A) is X more accurate than Random (N-A)	18.22%	7.81%	18.32%	6.37%	12.68%

Table 1. Comparative Average Distance between methods for the data sets. Informed Selection (Adaptive) is more accurate than all other methods. Overall, it performs 41% better than Uniform and 21% better than Informed Selection (Non-Adaptive). While Random (Adaptive) does better than Random (Non-Adaptive), overall it performs 30% worse than Uniform.

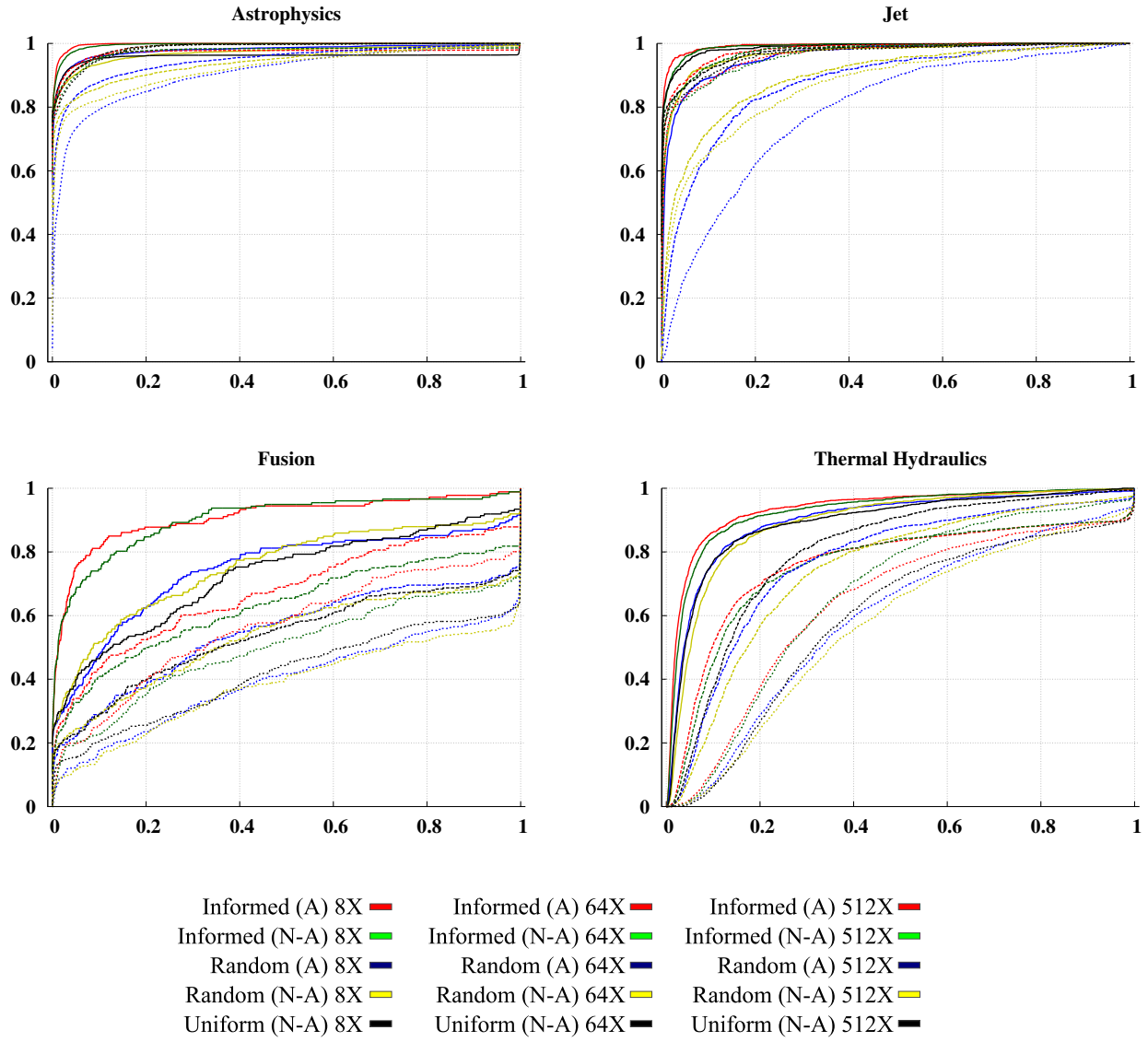

Figure 4. Four charts showing the accuracy achieved for our four data sets and three compression levels. The X axis is the error metric E, discussed in §4.4 and the Y axis is the cumulative distribution function (CDF) of particles. Discussion of how to interpret these charts can be found in Figure 3 and §5.1. The charts are laid out in a $2x2$ matrix, with each focusing on a separate data set. Each chart contains fifteen curves, corresponding to the cross-product of three compression ratios and five picking varieties. The colors are based on the picking method and the line styles on the compression ratio. These charts are in response to the research question posed in the Introduction, regarding which picking methods are best for maximizing accuracy. We envision that a simulation scientist would study tables like these when deciding what picking method to use and what compression level to use. The observations we make from these charts are discussed in §5.2.

The last column of Table 1 shows the general pattern for the performance of the three major picking techniques. On average, the Informed Selection method performs 41% more accurately than the Uniform method, which out performs Random by 30%. The trends for the individual data sets, mostly follow the averages from Table 1, and exemplified in Figure 4. As compression increases, a general pattern forms where the Informed Selection method performs more and more like Uniform as it attempts to broadly cover more space with an increasingly limited budget. On the other hand, Random performs much worse as compression increases because it may utilize its limited budget to sample densely in

certain areas and sparsely in others. However, one notable exception is with the Fusion data set, where Random out performs Uniform as the intensity of the flow field requires highly concentrated seeding for accurate measures and the arbitrary dense areas that form with Random sampling significantly improve its performance.

Selection	Jet	Met 0.5273	Fusion	Met 1.0508	Astro	Met 0.91758	Thermal	Met 1.87002
	512x	8x	512x	8x	512x	8x	512x	8x
Informed (A)	0.04834	0.061199	0.98186	0.99693	0.23477	0.22993	1.21052	1.23929
Informed (N-A)	0.052076	0.052367	0.9906	0.989923	0.29122	0.26009	1.2497	1.2613
Uniform (N-A)	0.000946	0.001105	0.004065	0.005037	0.006104	0.006788	0.00402	0.004597
Random (A)	0.038909	0.054269	0.97899	0.980653	0.242241	0.271331	1.2263	1.23411
Random (N-A)	0.001045	0.002594	0.00427	0.010482	0.0064234	0.010521	0.004176	0.010521

Table 2. Execution time for picking vectors with our five subsampling techniques in various configurations. The amount of picking time is proportional to the number of vectors to pick for *Random* and *Uniform*. *Informed Selection* has a sorting step that prevents a linear speedup.

Continuing our analysis, we elaborate on the differences between the adaptive and non-adaptive techniques. In all tests, the adaptive version of the selection method in question performs more accurately than the non-adaptive version. When working on a distributed system, the adaptive budget attempts to compensate for the inability of one part of the data to freely communicate with another. Cases may occur where one task holds a section of data where the flow is particularly laminar while another encapsulates highly turbulent and rotational fluid flow effects. The adaptive budget allows for less vector field samples to be taken in the laminar case, giving a greater budget for sampling the deformation occurring the later task. With the adaptive budgeting consistently outperforming the non-adaptive budgeting, we conclude that these type of cases occur frequently enough to advocate for an adaptive budget on distributed systems. Table 3 shows the magnitude of the adaptation in budget.

Test	Maximum	Average	Minimum
Jet, 8X	45203	8192	1088
Jet, 64X	5845	1024	128
Jet, 512X	4090	128	16
Star, 8X	306835	32768	4096
Star, 64X	35306	4096	512
Star, 512X	4473	512	64
Fusion, 8X	75881	32768	7683
Fusion, 64X	8509	4096	1218
Fusion, 512X	878	512	180
TH, 8X	59727	32768	4096
TH, 64X	8063	4096	512
TH, 512X	1007	512	64

Table 3. This table informs the extent that the budget is adapted across processors. For a given test (e.g., "Jet, 8X"), the maximum budget over all processors is listed in the maximum column, and the minimum budget over all processors is listed in the minimum column. Since the budget does not change based on data complexity, the average column represents the budget for each processor if there were no adaptation.

5.3 Suitability for *In Situ* Processing on Leading-Edge Supercomputers

Although our subsampling compression techniques do not occur *in situ* as part of a significantly large-scale simulation, we can still collect evidence about their suitability from the 64 task test. Specifically, we can measure:

- the memory overhead for our techniques, since memory will be a precious commodity on future high end supercomputers, and
- the execution time, since the simulation will require the compression technique to finish within some time budget, so it can continue advancing in time.

There is no memory used for *Random* and *Uniform*. For *Informed Selection*, we store an additional array (for fractional anisotropy and vorticity), and this array is proportional to the size of the mesh. The cost of calculating the fractional anisotropy and vorticity is shown in the top row of Table 2. Although the data sets are similar in size, the computation cost varies according to the eigenvalue and eigenvector calculations required which can be skewed depending on the linear system (calculated using the CGAL math library). However, the time to do the selection of vectors is smaller in comparison to the calculation of the fluid flow metrics. While lower, the *Informed Selection* has a sorting step that prevents a linear speedup and whose time to completion also heavily depends on the data. Additionally, two AllReduce communication calls are made between the tasks. The first call calculates the average fractional anisotropy and average vorticity and the second communicates the required budget and total number of refined cells across the data set. In total, 2 $float$ and 2 *int* values are communicated between the tasks.

Summarizing the suitability for *in situ* processing, the *Informed Section* method requires more memory and more execution time, especially during initialization. However, we believe the amounts are still within acceptable limits and that the selection technique is still viable, especially in light of the low amount of communication required to create an adaptive budgeting scheme which can therefore be applied to the *Informed Section* method, greatly increasing accuracy measures.

6. CONCLUSION AND FUTURE WORK

This work explores the direction of subsampling-based compression for particle advection-based flow visualization, in the context of *in situ*-fueled exploration, and illuminates tradeoffs between accuracy and compression. We feel the primary contributions of our study are three-fold: (i) the results that compare the error involved with varied configurations (144 tests overall), (ii) a novel technique for subsample selection, and (iii) exploration of of adapting the reduction budget over a distributed-memory environment.

This work suggests many interesting future directions. First, our algorithm for informed selection is (mostly) better than the other methods we studied, but it may not be optimal; studying improvements to this algorithm — both in accuracy and in execution time and memory footprint — would be interesting and beneficial to simulation scientists. Second, users often know *a priori* the most interesting regions to do flow analysis; exploring controls for prioritizing where the compression occurs could significantly improve results. Finally, additional analysis on which regions are best and worst represented would benefit users, and such analysis could be combined with uncertainty visualization techniques in novel ways.

6.1 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.

REFERENCES

[1] Childs, H., "Architectural Challenges and Solutions for Petascale Postprocessing," in [*SciDAC 2007*], *Journal of Physics Conference Series* **78**, 012012 (2007).

[2] Dongarra, J., Beckman, P., et al., "The International Exascale Software Roadmap," *International Journal of High Performance Computer Applications* **25**(1) (2011).

[3] Ahern, S. et al., "Scientific Discovery at the Exascale," tech. rep., Report for the DOE ASCR 2011 Workshop on Exascale Data Management, Analysis, and Visualization (July 2011).

[4] Childs, H., Ma, K.-L., Yu, H., Whitlock, B., Meredith, J., Favre, J., Klasky, S., Podhorszki, N., Schwan, K., Wolf, M., Parashar, M., and Zhang, F., "In Situ Processing," in [*High Performance Visualization—Enabling Extreme-Scale Scientific Insight*], 171–198 (Oct. 2012).

[5] McLoughlin, T., Laramee, R. S., Peikert, R., Post, F. H., and Chen, M., "Over Two Decades of Integration-Based, Geometric Flow Visualization," in [*EG 2009 - State of the Art Reports*], Pauly, M. and Greiner, G., eds., 73–92, Eurographics Association (April 2009).

[6] Lodha, S., Renteria, J., and Roskin, K., "Topology preserving compression of 2d vector fields," in [*Visualization 2000. Proceedings*], 343–350 (2000).

[7] Theisel, H., Rössl, C., and Seidel, H.-P., "Compression of 2d vector fields under guaranteed topology preservation," *Computer Graphics Forum* **22**(3), 333–342 (2003).

[8] Theisel, H., Rossl, C., and Seidel, H. P., "Combining topological simplification and topology preserving compression for 2d vector fields," in [*Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*], 419–423 (2003).

[9] Tikhonova, A., Yu, H., Correa, C. D., Chen, J. H., and Ma, K.-L., "A preview and exploratory technique for large-scale scientific simulations," in [*EGPGV*], Kuhlen, T., Pajarola, R., and Zhou, K., eds., 111–120, Eurographics Association (2011).

[10] Wang, C., Yu, H., and Ma, K.-L., "Importance-driven time-varying data visualization," *IEEE Trans. Vis. Comput. Graph.* **14**(6), 1547–1554 (2008).

[11] Wang, C., Yu, H., and Ma, K.-L., "Application-driven compression for visualizing large-scale time-varying data," *IEEE Computer Graphics and Applications* **30**(1), 59–69 (2010).

[12] Clyne, J., "Progressive Data Access for Regular Grids," in [*High Performance Visualization—Enabling Extreme-Scale Scientific Insight*], 152–170 (Oct. 2012).

[13] Gross, M. H., Lippert, L., and Staadt, O. G., "Compression methods for visualization," *Future Generation Computer Systems* **15**(1), 11–29 (1999).

[14] Clyne, J., Mininni, P., Norton, A., and Rast, M., "Interactive desktop analysis of high resolution simulations: application to turbulent plume dynamics and current sheet formation," *New Journal of Physics* **9**, 301 (Aug. 2007).

[15] Gosink, L., Anderson, J. C., Bethel, E. W., and Joy, K. I., "Variable Interactions in Query-Driven Visualization," *IEEE Transactions on Visualization and Computer Graphics (Proceedings of Visualization 2007)* **13**, 1400–1407 (October 2007).

[16] Gosink, L. J., Garth, C., Anderson, J. C., Bethel, E. W., and Joy, K. I., "An Application of Multivariate Statistical Analysis for Query-Driven Visualization," *IEEE Transactions on Visualization and Computer Graphics* **17**(3), 264–275 (2011).

[17] Rübel, O., Prabhat, Wu, K., Childs, H., Meredith, J., Geddes, C. G. R., Cormier-Michel, E., Ahern, S., Weber, G. H., Messmer, P., Hagen, H., Hamann, B., and Bethel, E. W., "High Performance Multivariate Visual Data Exploration for Extemely Large Data," in [*Supercomputing 2008 (SC08)*], (November 2008).

[18] Stockinger, K., Bethel, E. W., Campbell, S., Dart, E., and Wu, K., "Detecting Distributed Scans Using High-Performance Query-Driven Visualization," in [*SC '06: Proceedings of the 2006 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*], IEEE Computer Society Press (October 2006).

[19] Stockinger, K., Shalf, J., Wu, K., and Bethel, E. W., "Query-Driven Visualization of Large Data Sets," in [*Proceedings of IEEE Visualization 2005*], 167–174, IEEE Computer Society Press (October 2005).

[20] Ji, G. and Shen, H.-W., "Efficient Isosurface Tracking Using Precomputed Correspondence Table," in [*Proceedings of Symposium on Visualization (VisSym) '04*], 283–292, Eurographics Association (2004).

[21] Ji, G., Shen, H.-W., and Wegner, R., "Volume Tracking Using Higher Dimensional Isocontouring," in [*Proceedings of IEEE Visualization '03*], 209–216, IEEE Computer Society (2003).

[22] Silver, D. and Wang, X., "Tracking and Visualizing Turbulent 3D Features," *IEEE Transactions on Visualization and Computer Graphics (TVCG)* **3**(2), 129–141 (1997).

[23] Silver, D. and Wang, X., "Tracking Scalar Features in Unstructured Datasets," in [*Proceedings of IEEE Visualization '98*], 79–86, IEEE Computer Society Press (1998).

[24] Chen, Y., Cohen, J. D., and Krolik, J. H., "Similarity-guided streamline placement with error evaluation," *Visualization and Computer Graphics, IEEE Transactions on* **13**(6), 1448–1455 (2007).

[25] Xu, L., Lee, T.-Y., and Shen, H.-W., "An information-theoretic framework for flow visualization," *Visualization and Computer Graphics, IEEE Transactions on* **16**(6), 1216–1224 (2010).

[26] Chen, G., Mischaikow, K., Laramee, R. S., Pilarczyk, P., and Zhang, E., "Vector field editing and periodic orbit extraction using morse decomposition," *IEEE Trans. Vis. Comput. Graph.* **13**(4), 769–785 (2007).

[27] "CGAL, Computational Geometry Algorithms Library." http://www.cgal.org.