# Uncertainty Modeling and Error Reduction for Pathline Computation in Time-varying Flow Fields

Chun-Ming Chen Ayan Biswas Han-Wei Shen

The Ohio State University *

## Abstract

When the spatial and temporal resolutions of a time-varying simulation become very high, it is not possible to process or store data from every time step due to the high computation and storage cost. Although using uniformly down-sampled data for visualization is a common practice, important information in the un-stored data can be lost. Currently, linear interpolation is a popular method used to approximate data between the stored time steps. For pathline computation, however, errors from the interpolated velocity in the time dimension can accumulate quickly and make the trajectories rather unreliable. To inform the scientist the error involved in the visualization, it is important to quantify and display the uncertainty, and more importantly, to reduce the error whenever possible. In this paper, we present an algorithm to model temporal interpolation error, and an error reduction scheme to improve the data accuracy for temporally down-sampled data. We show that it is possible to compute polynomial regression and measure the interpolation errors incrementally with one sequential scan of the time-varying flow field. We also show empirically that when the data sequence is fitted with least-squares regression, the errors can be approximated with a Gaussian distribution. With the end positions of particle traces stored, we show that our error modeling scheme can better estimate the intermediate particle trajectories between the stored time steps based on a maximum likelihood method that utilizes forward and backward particle traces.

## 1 Introduction

Fluid flow analysis plays an important role in many scientific applications, such as aerodynamics, climate modeling and medicine. To understand time-varying flow fields, displaying pathlines is a common way to reveal complex flow features. Pathlines are the trajectories of particles moving in time-varying flows. They are also a fundamental tool for deriving other visualization and analysis techniques, such as path surfaces, streaklines, and FTLE [10]. With the advances in high-performance computer architectures, flow simulations can now produce high-resolution flow fields with millions of spatial grid points and thousands of time steps or more. However, machines for post analysis and visualization usually have much smaller storage space and can only handle a small fraction of the original simulation output. A common practice to reduce the data size while keeping the original spatial resolution is to reduce the temporal sampling rate, e.g., once every tens or hundreds of time steps [16, 21]. Then the data values at time steps in between are approximated by linear interpolation. However, this inevitably introduces errors into the analysis and visualization results. Although the error at an individual point may be small, pathline computation, which integrates the interpolated vectors, will accumulate errors quickly, and thus the flow lines will become rather unreliable. While errors are inevitable, quantification and visualization of uncertainty is imperative. To visualize the uncertainty of flow fields, probabilistic pathline tracing has been proposed [14, 15, 17], where the flow field uncertainty is modeled in probability distributions. While most of the uncertainty analysis algorithms assume that flow field uncertainty comes from measurements or ensemble runs, little focus has been on the analysis of interpolation errors caused by down-sampling time-varying flow fields.

In this paper, we present a storage and I/O-efficient algorithm to model the interpolation errors for flow fields down-sampled in time. For each down-sampled value sequence we collect and store the distribution of the interpolation errors, which is later used to model the uncertainty of the interpolated values. To minimize the storage overhead, we observe that if a data sequence is fitted with least-squares regression, the error distribution can be approximated with a Gaussian distribution where its standard deviation is the standard error of the regression. Based on this idea, we propose a higher-order polynomial interpolation scheme and show that by storing only two extra parameters for each down-sampled value sequence, the interpolation error can be reduced while the error distribution needed for uncertain pathline visualization being approximated. Moreover, we show that our regression and error modeling method can be computed *incrementally*, by only one sequential scan of the flow data with memory requirement invariant to the number of time steps in the sampling interval. This incremental computation reduces the I/O cost and memory footprint of the error modeling procedure and is suitable for data streaming scenarios where data are available only for a short period of time and cannot be accessed afterwards. For example, it is suitable for *in situ* analysis of simulation output, where data are available in core only before being replaced by data generated in the next time step. With the error distribution, we further show that more accurate particle trajectories can be estimated from just the particle end positions at the sampled time steps. To achieve this, for each grid point $P1$ at the beginning of a sampling time interval, we perform pathline tracing but only store the particle end position $P2$ at the end of each time interval to save storage. To obtain a more accurate pathline within this time interval without touching the data in between, we propose a forward-and-backward trace intersection technique that computes two uncertain pathlines from $P1$ and $P2$ respectively in the opposite directions toward each other, and then merge the two uncertain traces by the maximum likelihood principle. This refinement technique is shown to generate more accurate pathlines than conventional approaches, where only the down-sampled flow fields with our error model and the final particle positions per sampling interval are available.

The contributions of this paper are threefold. (1) We present a storage-efficient error modeling method for interpolating down-sampled time-varying flow fields. (2) We show our error model can be computed incrementally to achieve efficient I/O and memory usage. (3) We provide a probability based method to refine the intermediate particle trajectories using our error model.

This paper is organized as follows: In Section 2 we review the related works. Section 3 provides an overview of our error modeling procedure. Section 4 describes our data regression and error modeling method, followed by the incremental computation algo-

*e-mail: {chenchu,biswas,hwshen}@cse.ohio-state.edu

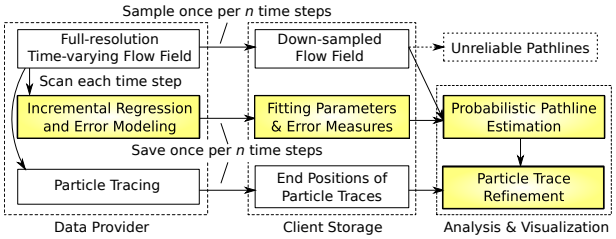


Figure 1: Overview of our system. White blocks: existing data down-sampling procedures. Color thick blocks: proposed error modeling method for uncertain pathline computation and refinement.

rithm in Section 5 and the pathline refinement algorithm in Section 6. Experimental results are shown in Section 7 and discussed in Section 8. Finally, we conclude the paper in Section 9.

## 2 Related Works

Data reduction while providing uncertainty measurements addresses the problem of the inevitable information loss through lossy compressions. A traditional approach is to store the error measures like error bounds, peak signal-to-noise ratio (PSNR) or root mean square error (RMSE) [3, 16]. To provide statistical information about the lost data, Thompson *et al.* [20] introduced Hixels to represent local region distributions in histograms. Chaudhuri *et al.* [6] constructed multi-resolution histograms for large-scale scalar and vector data. Liu *et al.* [13] fitted local distributions by Gaussian mixture models (GMM) to further reduce the storage size for GPU rendering. In this paper we model the uncertainty by the distribution of errors from polynomial fitting in the temporal dimension. Among data compression techniques, Ameer [1] investigated polynomial fitting models for lossy image compression and showed that the linear polynomial model produces less error than JPEG when a high compression rate is desired. Recently, Fout and Ma [8] presented a lossless polynomial-based predict-coding method for floating-point value sequences.

Uncertainty analysis and visualization have received considerable attention for more than a decade [12]. Lately Brodie *et al.* [3] provided a broad review including discussions about the sources of uncertainty and their visualization with different data types. For uncertain particle tracing, Luo *et al.* [15] and Love *et al.* [14] modeled the uncertain velocities in PDFs and performed distribution-based particle integration based on Euler's equation. After each integration step, a representative particle location, e.g., the mean location, is extracted as the starting position for the next step. This approach approximates the actual pathline distribution with reduced computation complexity. In topology analysis of uncertain flow fields, Otto *et al.* [17] constructed flow field topology using a Monte Carlo approach, which updates a particle density field describing the uncertain particle location. Schneider *et al.* [19] generated stochastic flow maps and measured the statistical flow separation analogous to FTLE in deterministic flow fields. The idea of our forward and backward trace intersection method can be related to estimation refinement methods incorporating information from the both ends of a time interval, such as optical flow computation in video analysis. In vector field analysis, recently Chen *et al.* [7] used forward and backward mappings to compute the approximate image for Morse decomposition.

## 3 System Overview

In this section we provide an overview of our incremental error analysis and pathline refinement method, as shown in Figure 1. The first row of the figure shows a common procedure of pathline computation from down-sampled flow fields. The time-varying flow field is first sampled at a fixed time interval, referred to as the *sampling interval*, denoted as $n$. After data down-sampling, only the data at the two ends of the sampling intervals are available for pathline computation, but the result will be unreliable, since the information in the unsaved time steps is lost and there is no indication of the error from linear interpolation.

To provide error measurement and control, we propose an error reduction and modeling scheme for down-sampled flow fields. As shown in the second row of Figure 1, we iteratively load each time step of the flow field and incrementally compute polynomial regression and its error measure over the time series of data. At the end of each sampling interval, i.e., every $n$ time steps, we save the resulting polynomial parameters and the error measures to describe the uncertainty of the unsaved $(n-1)$ time steps of the data.

Since errors are accumulated quickly when computing pathlines from down-sampled flow fields, to improve the accuracy of the pathlines, we compute the end position of the particle originated from each grid point at the beginning of the sampling interval after traveling $n$ time steps. This is similar to computing a flow map [10] for every $n$ time steps. Storing flow maps in longer time steps has been used to improve the pathline accuracy and provide general information about the flow structure like FTLE [5, 11]. Given the flow maps, we can utilize the error distribution computed above to refine the pathlines between the saved time steps. This component of our algorithm is illustrated in the last row of Figure 1, detailed in Section 6.

## 4 Error Modeling and Reduction Method

In this section, we describe a storage-efficient method to model the uncertainty of data down-sampled in time, where the uncertainty is quantified by the distribution of temporal interpolation errors.

### 4.1 Terminology

We first define the terms that will be used in the rest of the paper. Assume that the time-varying data at each step $i$, $i = 0,1,2...$ has a time stamp $t_i \in \mathbb{R}$. Given a grid point at a spatial coordinate $\mathbf{x}$, we form a sequence from a time-varying scalar field $\mathbf{F}$:

$$y_i = \mathbf{F}(\mathbf{x}, t_i) \in \mathbb{R}, i = 0,1,2,... \quad (1)$$

Note that for a vector field in the flow field, we treat each vector component independently as separate scalar fields.

The time-varying flow field is sampled once every $n$ steps, i.e., only data at steps $0, n, 2n, ...,$ are stored. Thus interpolation is used to approximate $y_i$ as $\hat{y}(t_i)$ at step $i$. The interpolation error is defined as the difference between the interpolated value and the true value, i.e., $y_i - \hat{y}(t_i)$. Below we explain our error modeling method for a time sequence at a single grid point within the first sampling time interval $[t_0, t_n]$. The same routine is applied to every grid point in every sampling interval.

### 4.2 Least Squares Estimation and Gaussian Errors

When modeling the distribution of a group of samples, Gaussian distribution is usually considered to provide a compact approximation. However, it is not always true that the samples are from a Gaussian distribution. Since our goal is to model the distribution of interpolation errors, one can alter the interpolation function, or fitting function, in a way that the error distribution is closer to a desired model. To do this, it is known that the least squares estimators optimize the function parameters by modeling the errors as a Gaussian distribution, and thus the function parameter estimation is equivalently a maximum likelihood estimation [2, 18]. With a regression function $f$, a data sequence $(x_i, y_i), i = 0,1,...n$, can be modeled as:

$$y_i \simeq f(x_i) + E, \quad (2)$$

where $E$ denotes a random variable representing the error that can take positive or negative values. The least squares estimator approximates $f$ by modeling $E$ as a Gaussian distribution with a zero mean and an unknown variance, which is independent to the input $x_i$. Using the same model as in Equation 2, we fit the input data sequence with polynomial regression as the estimator and approximate the errors as a Gaussian distribution. In Section 7.2, we empirically validate this assumption with several fluid flow simulation datasets with statistical hypothesis tests.

With the errors of a least-squares regression modeled as a Gaussian distribution with zero mean, we only need to estimate and store its variance. Based on the least squares fitting, the unbiased estimator of the variance is derived [9] as

$$s^2 = \sum_{i=0}^{n} \frac{(y_i - f(t_i))^2}{(n+1)-K}, \tag{3}$$

where $K$ is the number of parameters used in the regression. Note $s$ is referred to as the *standard error* of the regression.

### 4.3 Quadratic Bezier Curve Fitting

To apply polynomial regression to our problem, it is necessary to determine an appropriate polynomial model to describe the data in the unsaved time steps. To deal with the challenge of large data sets, our goal is to find a polynomial model that requires the minimal storage space but with reasonable accuracy. Because in our framework the data in both ends of a sampling interval are stored, we enforce the polynomial function to always go through those end points. By doing so, the degrees of freedom are reduced by two for the polynomial and hence we have two less parameters to store. So, instead of using a general polynomial function, we choose the Bezier curve function, which is a form of polynomial with the end points fixed. Equation 4 defines a quadratic Bezier curve with two end points $\mathbf{P}_0$ and $\mathbf{P}_2$, and a control point $\mathbf{P}_1$:

$$\mathbf{B}(r) = (1-r)[(1-r)\mathbf{P}_0 + r\mathbf{P}_1] + r[(1-r)\mathbf{P}_1 + r\mathbf{P}_2], \tag{4}$$

where $r \in [0,1]$. The equation is also applicable to vector inputs. $\mathbf{B}(r)$ can be interpreted as an interpolation between the two end points $\mathbf{P}_0$ and $\mathbf{P}_2$.

The Bezier curve fitting also guarantees $C^0$-continuity for values on both ends of the sampling interval, which is not always true for a general polynomial since each sampling interval is fitted separately. Although Bezier curve fitting produces slightly higher error than general polynomial fitting of the same order, this drawback is relatively minor compared to the benefit stated above.

To select an appropriate polynomial order, we empirically choose the quadratic form, since for each input sequence it only requires one extra control point, hence less storage overhead, given that the two end points are always stored. We apply the quadratic Bezier curve fitting and model the Gaussian error for each value sequence within the sampling interval. As a result, for each sampling interval we store two extra values per grid point, the control point of the quadratic Bezier curve and the standard error.

Finally, based on Equation 2, for each grid point $\mathbf{x}$ at time $t \in [0,n]$ within a sampling time interval, we estimate the value $V$ with Gaussian error by:

$$V_{\mathbf{x},t} \simeq \mathbf{B}_{\mathbf{x}}(t/n) + \mathrm{N}(0, s_{\mathbf{x}}^2), \tag{5}$$

where $\mathbf{B}_{\mathbf{x}}(t/n)$ denotes the Bezier curve function as in Equation 4 and $s_{\mathbf{x}}$ denotes the standard error corresponding to $\mathbf{x}$.

## 5 Incremental Error Modeling Algorithm

With the regression function defined by Bezier curves and the error modeled as Gaussian distributions, we show that the computation can be both I/O and memory-efficient for the data provider to analyze the original large data. We define our incremental algorithm with the following two constraints:

1. The algorithm reads the data in sequential time steps in only one scan. This reduces the I/O overhead of the error modeling procedure. It is also beneficial for analyzing streamed data where data are available in a sequential order.

2. The memory usage in the entire run should be invariant to the number of time steps in a sampling interval. Thus the algorithm is scalable to any arbitrary sampling rate. We assume that the memory can only cache a few time steps of the data.

Data regression and error distribution computation usually require two separate runs with the data, since the individual error, which is the difference between the estimated and the true value, cannot be computed before the regression function is determined. However, we show that the regression parameters and its standard error required in our error modeling can be computed together incrementally. As an illustration we first show that the RMSE of linear interpolation can be computed incrementally, which will then be used to derive the algorithm for our problem.

### 5.1 RMSE from Linear Interpolation

Given two temporal points $(t_0, y_0)$ and $(t_n, y_n)$, linear interpolation estimates the intermediate value $\hat{y}(t)$ at time $t \in \mathbb{R}, t_0 \le t \le t_n$ by

$$\hat{y}(t) = \frac{(t_n - t)\cdot y_0 + (t - t_0)\cdot y_n}{t_n - t_0}. \tag{6}$$

We will use $\hat{y}$ as the estimation to the input sequence. Thus the root-mean-square error (RMSE) of $\hat{y}$ is defined as

$$RMSE(\hat{y}) = \sqrt{\frac{1}{n+1}\sum_{i=0}^{n}(y_i - \hat{y}(t_i))^2}. \tag{7}$$

During the incremental computation, since the interpolated values $\hat{y}(t_i), 0 < i < n$ are unknown until the value $y_n$ in the end point of the interpolation interval becomes available at step $n$, we cannot compute each error $(y_i - \hat{y}(t_i))$ when $y_i$ is just produced. An intuitive approach is to store all values of $y_0 \cdots y_n$ before computing *RMSE*, but it violates our goal of using constant storage. However, with an arrangement of Equation 7, RMSE can be computed by incrementally accumulating some summations related to $y_i$ at each time step. That is, by expanding Equation 7 and grouping each summation, we obtain:

$$RMSE(\hat{y}) = \sqrt{\frac{1}{n+1}\left(\sum_{i=0}^{n} y_i^2 - 2\sum_{i=0}^{n}(y_i \cdot \hat{y}(t_i)) + \sum_{i=0}^{n}\hat{y}(t_i)^2\right)}. \tag{8}$$

In Equation 8, the first term $\sum y_i^2$ can be incrementally accumulated at each time step; the last term can be summed at step $n$, since each $\hat{y}(t_i)^2$ can be computed using Equation 6, which only involves $y_0$ and $y_n$. The middle term needs further derivation using Equation 6, and by grouping each summation we get:

$$\sum_{i=0}^{n}(y_i \cdot \hat{y}(t_i)) = \frac{t_n \cdot y_0 - t_0 \cdot y_n}{t_n - t_0}\sum_{i=0}^{n} y_i + \frac{y_n - y_0}{t_n - t_0}\sum(y_i \cdot t_i). \tag{9}$$

As shown in Equation 8 and 9, by incrementally accumulating $\sum y_i$, $\sum(y_i \cdot t_i)$ and $\sum y_i^2$ at each time step, together with $y_0$ and $y_n$ available at the last time step, the RMSE for linear interpolation can be sequentially computed. This is an incremental computation with one scan of the sequence, and it only requires constant memory for storing the above summations during the iterations, regardless of the length of the input sequence.

### 5.2 Incremental Quadratic Bezier Curve Fitting

In this section we provide the algorithm for incremental quadratic Bezier curve fitting and the standard error computation. In general, least squares regression optimizes the function parameters with minimum sum of squared errors to the data values:

$$\text{argmin}_f \sum_{i=0}^{n} (y_i - f(x_i))^2. \tag{10}$$

To optimize $f$ is to solve the equation when the derivative of the above function equals to zero. For detailed derivation of solving polynomial regression the interested reader is referred to [18].

To optimize a quadratic Bezier curve, given an input sequence of time-value pairs $\{(t_i, y_i) | i = 0, 1, 2, ..., n\}$, where each $t_i$ is normalized into the range $[0,1]$, i.e., $t_i = \frac{i}{n}$, we define two end points $p_0 = y_0$ and $p_2 = y_n$, as the end points of the Bezier curve. One can derive the equation for computing the control point $p_1$ as follows:

$$p_1 = \frac{\sum(y_i t_i) - \sum(y_i t_i^2) - p_0 \sum((1-t_i)^3 t_i) - p_2 \sum((1-t_i) t_i^3)}{2\sum((1-t_i)^2 \cdot t_i^2)}, \tag{11}$$

where $\sum(\cdot)$ is short for $\sum_{i=0}^{n}(\cdot)$. Equation 11 shows that we only have to preserve $p_0$ and accumulate the two partial sums $\sum y_i \cdot t_i$ and $\sum y_i \cdot t_i^2$ during the incremental computation. The rest summations can be pre-computed since they are not related to the data value $y_i$.

To compute the standard error for the results of quadratic Bezier curve fitting, we apply Equation 3 and assign $K = 1$ for one unknown $p_1$. We observe that the computation in Equation 3 resembles RMSE in Equation 7 with a different constant in the denominator (now it is $n$ for $n+1$ data inputs). Therefore similar to the derived equation in Equation 9, we need to obtain the summations $\sum y_i^2$, $\sum(y_i \cdot \hat{y}(t_i))$ and $\sum \hat{y}(t_i)^2$ online, where $\hat{y}(t_i)$ is now estimated by the Bezier curve. Among these three summations, the first term has been summed up incrementally; the last term is summed by computing each $\hat{y}(t_i)$ from Equation 4, where $p_0$, $p_1$ and $p_2$ are known at the end of the interval. The second term is derived as:

$$\sum y_i \cdot \hat{y}(t_i) = p_0 \sum y_i + 2(p_1 - p_0) \sum (y_i \cdot t_i) + (p_0 - 2p_1 + p_2) \sum (y_i \cdot t_i^2). \tag{12}$$

Therefore, the standard error can also be computed incrementally by collecting an additional summation $\sum y_i$.

We summarize our incremental error modeling algorithm in Algorithm 1. Note that the same procedure is applied on each grid point at each iteration. In the end of each sampling interval the corresponding control points and the standard errors are generated and stored. As a result the extra storage required for each sampling interval is equal to twice of the data size of one time step, which is still relatively small compared to the original data size. What we gain is the error distribution with more accurate interpolation.

**Algorithm 1** Incremental error modeling algorithm

1: Initialize *ysum*, *ytsum*, *yt2sum* and *y2sum* to 0, each representing $\sum y_i$, $\sum y_i \cdot t_i$, $\sum y_i \cdot t_i^2$ and $\sum y_i^2$ , respectively.
2: **for** $i = 0$ **to** $n$ **do** {For each time step in a sampling interval}
3: Read data value $y_i$
4: Let $t \leftarrow i/n$
5: $ysum \leftarrow ysum + y_i$
6: $ytsum \leftarrow ytsum + y_i \times t$
7: $yt2sum \leftarrow yt2sum + y_i \times t^2$
8: $y2sum \leftarrow y2sum + y_i^2$
9: **end for**
10: Compute the control point $p_1$ with Equation 11
11: Compute the standard error $s$ with Equation 3, 4 and 12
12: **return** Bezier-curve control point $p_1$ and $s$.

## 6 Uncertain Pathline Estimation and Refinement

With the errors modeled as Gaussian distributions for the temporally down-sampled flow fields, in this section we present a probability based particle tracing algorithm to refine particle traces computed between the pre-stored time steps.

As we are processing data of the incoming time steps and computing the true particle traces incrementally using the full-resolution flow field, it is not possible to store the trace positions at every time step for every possible seed location. To reduce the storage overhead, without having to identify important pathlines through detailed analysis, one common solution is to store the particle traces only at certain sample time steps. The disadvantage of doing this, however, is that important details of the pathlines in the intermediate time steps are lost. In this section we show that with the error distributions computed from the temporally down-sampled flow field, as described previously, it is possible to identify the uncertainty and more importantly, to compute more accurate pathlines. In the following, without loss of generality, we assume that pre-computed particle traces are sampled only at every $n$ time steps, where $n$ is the sampling interval used for the flow fields.

### 6.1 Probabilistic Particle Integration

We first review a general probabilistic particle integration method given an uncertain velocity field. To model a static 3D velocity field with uncertainty, we use a 3D random variable $\mathbf{V_x}$ to describe the probability distribution of the velocity at location $\mathbf{x}$ in the data domain D. That is, $P[\mathbf{V_x} \subset [< u, v, w >, < u+du, v+dv, w+dw >]]$ denotes the probability of an uncertain vector at location $\mathbf{x}$, where its vector values are within the range $< u, u+du > \times < v, v+dv > \times < w, w+dw >$, where $(u, v, w) \in \mathbb{R}^3$. For a time-varying uncertain flow field at time $t$, we use the notation $\mathbf{V_{x,t}}$.

To model an uncertain particle position, we define a 3D random variable $\mathbf{X}_t$ as a particle density distribution at time $t$. To integrate the particle distribution, one can loop through all possible locations $(x, y, z) \in$ D and compute the probability of a particle moving to location $(x, y, z)$ from the previous time step using Euler's integration method with $\mathbf{V}$. This convolution computation is involved with two nested loops of all possible locations. For more details the interested reader is referred to Otto *et al.*'s work [17].

### 6.2 Intermediate Trajectory Estimation

Given the particle positions on the same trace at the beginning and end time steps of a time interval, we show that with a temporally approximated vector field and the knowledge of errors, we can better estimate the particle trajectory in between. We observe that by utilizing a probabilistic particle tracing method, we can obtain two independently approximated particle traces within the same time interval–forward tracing from the starting position and backward tracing from the end position. These two estimations have a smaller error close to the particle's initial position and a larger uncertainty in the end because errors are accumulated. To get a more accurate particle trace, we can combine these two approximated pathlines based on the co-occurrences of possible particle locations from both traces. Probabilistically, this is done by intersecting the two distributions:

$$\mathbf{X}_t^{\text{Intersect}} = \mathbf{X}_t^{\text{Fw}} \bigcap \mathbf{X}_t^{\text{Bw}}, \tag{13}$$

where $\mathbf{X}_t^{Fw}$ and $\mathbf{X}_t^{Bw}$ denote the respective forward and backward particle distributions at time $t$. Since both traces are independently estimated and thus the corresponding random variables are independent, the intersection can be computed by multiplying the two PDFs. Note that after the multiplication, normalization of the resulting probability distribution is required, so that the integration of the probabilities remains to be one.

Figure 2(a) illustrates the intersection of two Gaussian distributions. As we can see, the maximum likelihood of the intersection is

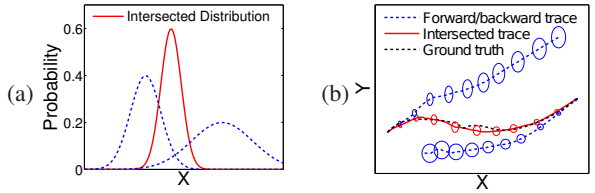


Figure 2: (a): Illustration of intersection of two Gaussian distributions. (b): Illustration of our forward-backward trace intersection method. Each ellipse represents the uncertainty of the corresponding trace at a sampled time step modeled as a 2D Gaussian, where the radius is proportional to the standard deviation.

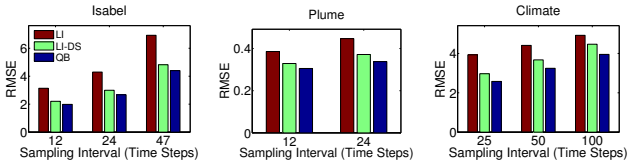


Figure 3: RMSE of different interpolation methods to the ground truth. Red: linear interpolation. Green: linear interpolation with a double sampling rate. Blue: quadratic Bezier curve interpolation.

closer to the distribution on the left, which has less standard deviation, or less uncertainty. In addition, the standard deviation of the intersected distribution is reduced and becomes less uncertain.

### 6.3 Pathline Integration Using the Gaussian Model

Since the flow field is modeled with errors as a Gaussian distribution, we present an efficient method to avoid the costly convolution computation for probabilistic particle tracing, by approximating the particle distribution at each time step with a Gaussian distribution.

A multi-dimensional Gaussian distribution $X$ is defined by two parameters, the mean vector $\mu$ and the covariance matrix $\Sigma$. The PDF is typically denoted by $X \sim \mathrm{N}(\mu,\Sigma)$. Assuming $X \sim \mathrm{N}(\mu_X,\Sigma_X)$ and $Y \sim \mathrm{N}(\mu_Y,\Sigma_Y)$, the linear combination of these two Gaussian distributions is

$$aX + bY \sim \mathrm{N}(a\mu_X + b\mu_Y, a^2\Sigma_X + b^2\Sigma_Y), a,b \in \mathbb{R}, \qquad (14)$$

which is another Gaussian distribution. We use this property to derive our Gaussian-based particle tracing method as the following.

Initially a deterministic seeding position $\mathbf{x}_0$ is given, i.e., $\mathbf{X}_{t_0} \sim \mathrm{N}(\mathbf{x}_0, \mathbf{0})$. To obtain the velocity at an arbitrary spatiotemporal position in a 4D domain, typically linear interpolation with the neighboring 16 grid points on a Cartesian space is performed. Since we use the quadratic Bezier curve as the interpolant for the time dimension, for each neighboring grid point we first perform temporal interpolation using Equation 4 and model the Gaussian error with the stored standard error $s$. Together we form a multivariate version of Equation 5 for the interpolated vector on a grid point $\mathbf{x}$ at time $t$:

$$\mathbf{V}_{\mathbf{x},t} \simeq \mathbf{B}_{\mathbf{x}}(t) + \mathrm{N}(\mathbf{0}, diag(\mathbf{s}_{\mathbf{x}}{}^2)). \qquad (15)$$

Here $\mathbf{B}_{\mathbf{x}}(t)$ and $\mathbf{s}_{\mathbf{x}}$ are 3D vectors, and $diag(\mathbf{s}_{\mathbf{x}}{}^2)$ is a diagonal matrix with individually squared elements of $\mathbf{s}_{\mathbf{x}}$ on the diagonal. With the error modeled as a Gaussian distribution, the interpolated velocity at time step $t$ on each grid point is also Gaussian. Therefore to obtain $\mathbf{V}$ at an arbitrary position at time step $t$, we compute tri-linear interpolation from the neighboring grid points using Equation 14.

To speed up the uncertain pathline estimation, we approximate the probabilistic particle integration by assuming that the probable particle locations are in a small local region, so that the velocities in this region are similar to that at the center of the particle distribution $\mu_{\mathbf{X}_t}$. This idea is similar to the uncertain pathline visualization method used by Luo *et al.* [15], where the velocity distribution at the representative particle location is applied at each integration step. With this idea we approximate the particle integration as the following equation, which resembles Eular's method:

$$\mathbf{X}_{t+\Delta t} \simeq \mathbf{X}_t + \mathbf{V}_{\mu_{\mathbf{X}_t},t} \cdot \Delta t, \qquad (16)$$

where $\Delta t$ is the integration step size. This computation involves another linear combination of the two Gaussian distributions, position $\mathbf{X}$ and the displacement as the velocity $\mathbf{V}$. By this integration method, $\mathbf{X}_i$ is always in a Gaussian distribution, which avoids the double-loop convolution computation and large memory requirement for storing the particle distribution described in Section 6.1. The computation of backward particle tracing is similar, by assigning a negative $\Delta t$ in Equation 16.

Finally, to intersect two uncertain pathlines, we intersect the two random variables $\mathbf{X}_t^{Fw}$ and $\mathbf{X}_t^{Bw}$ at each corresponding time step. The following intersection equation for Gaussian distributions is derived [4]. Let $\mathrm{N}(\mu_c,\Sigma_c) = \alpha \mathrm{N}(\mu_1,\Sigma_1) \cdot \mathrm{N}(\mu_2,\Sigma_2)$, where $\alpha$ is the normalizing constant, we get:

$$\mu_c = (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2) \qquad (17)$$

$$\Sigma_c = (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1} \qquad (18)$$

As shown in Equation 17, the new mean $\mu_c$ is a linear combination of the two input mean values, weighted inversely by their respective variance. Therefore for each intermediate time step, the combined mean will be pulled to the one that is more certain, which is as desired for our particle path refinement.

Figure 2(b) illustrates our forward-backward intersection method applied on the test dataset *Isabel*. Given the start and end positions of a particle trace, we perform Gaussian-based particle tracing respectively and obtain two uncertain traces, as shown by the blue dashed lines. By intersecting these two compliment traces at each corresponding time step, we obtain the trace in red, which is very similar to the true pathline as the black dashed line.

## 7 Experimental Results

In this section we show the experimental results of our error modeling of the flow field and the pathline refinement methods. Three time-varying datasets were used in our experiments: *Isabel*, *Plume* and *Climate*. *Isabel* is a simulation of hurricane Isabel from the West Atlantic region in 2003. The dimensions are $500 \times 500 \times 100$ with 48 time steps. *Plume* is a simulation of the thermal activity on the surface of the Sun, with dimensions of $252 \times 252 \times 1024$ with 29 time steps. *Climate* is a climate simulation over the Indian and Pacific Ocean. The dimensions are $2699 \times 599 \times 50$ with 101 time steps. Since different datasets have different number of time steps, we chose 12, 24 and 47 time steps as the sampling intervals for *Isabel*, 12 and 24 for *Plume*, and 25, 50 and 100 for *Climate*.

### 7.1 Comparison of Interpolation Methods

Since we perform polynomial regression on the flow field data, to see how much errors are reduced, we compared the RMSE measures with our Bezier-curve interpolation and linear interpolation. Since in our method, each quadratic Bezier curve requires to store an extra control point, to perform a fair comparison we also compared the linear interpolation of data sampled with a double sampling rate; i.e., one extra time step of data in the middle of each sampling interval are stored and interpolated. Figure 3 shows the average RMSE measures with each sampling interval of different length. In each case, we can see that the quadratic Bezier curve interpolation (blue bars) can reduce the error the most given the same amount of storage space.

Table 1: Percentage of samples where the errors of the estimation are within the 1.96 estimated standard deviation. (a): Test on the sequences of vector fields fitted by quadratic Bezier curves. (b): Test on the estimated trace locations using our forward-backward intersection method to the true locations.

| Dataset | Isabel | | | Plume | | Climate | | |
|---|---|---|---|---|---|---|---|---|
| Sampling Interval | 12 | 24 | 47 | 12 | 24 | 25 | 50 | 100 |
| (a) Vector field | 95.9 | 95.5 | 95.0 | 95.7 | 96.1 | 95.4 | 95.5 | 95.2 |
| (b) Particle trace | 94.3 | 86.9 | 83.0 | 72.9 | 78.8 | 89.6 | 75.7 | 64.8 |

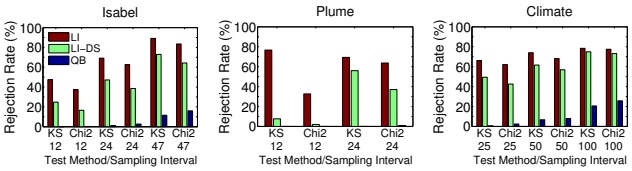


Figure 4: Validation of Gaussian errors using Kolmogorov-Smirnov test (KS) and Chi-square goodness-of-fit test (Chi2). The percentage of samples rejected by each test is shown. Red: linear interpolation. Green: linear interpolation with a double sampling rate. Blue: quadratic Bezier curve interpolation.

Table 2: Comparison of the file sizes in each test case. The extra storage includes the error measures and the pathline end positions from each grid point and each sampling interval.

| Dataset | *Isabel* | | | *Plume* | | *Climate* | | |
|---|---|---|---|---|---|---|---|---|
| Original File Size / Time steps | 14.4 GB / 48 | | | 22.6 GB/29 | | 98.0 GB / 101 | | |
| Sampling Time Interval | 12 | 24 | 47 | 12 | 24 | 25 | 50 | 100 |
| Down-sampled Time Steps | 5 | 3 | 2 | 3 | 2 | 5 | 3 | 2 |
| Down-sampled Flow Field Size (GB) | 1.5 | 0.9 | 0.6 | 2.3 | 1.6 | 4.9 | 2.9 | 1.9 |
| Total Extra Storage Size (GB) | 3.6 | 1.8 | 0.9 | 4.7 | 2.3 | 11.6 | 5.8 | 2.9 |

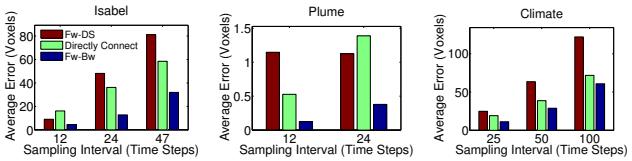


Figure 5: Average error of estimated trajectories to the true pathline. Red: Directly connect. Green: Forward tracing on flow fields with a double sampling rate. Blue: Our forward-backward intersection method.

### 7.2 Validation of Gaussian-Distributed Errors

Now we validate our hypothesis that the errors from our quadratic Bezier curve fitting can be approximated as a Gaussian distribution. We first computed errors for each sampled data sequence, where an error sequence was formed by the differences between the true values and the approximated values. The sequence was then tested with two statistical tools, the Chi-square goodness-of-fit test and the Kolmogorov-Smirnov test against our hypothesis that the test sequence is a Gaussian with zero mean and the estimated standard error as the standard deviation. The former test is widely known while the latter is more robust and preferred nowadays. We used the threshold value (Alpha) 0.05 in both tests. That is, if the computed $p$-value is less than the threshold, our hypothesis is rejected. More than 10,000 randomly sampled data sequences were tested for each case and the rejection rate was recorded. For comparison, we also tested the normality of error distributions from using the original linear interpolation with and without the double sampling rate.

As shown in Figure 4, when the sampled sequences were approximated by the quadratic Bezier curve fitting (in blue bars), with 12 time steps as the sampling interval, less than 1% of the samples were rejected against our normality hypothesis by either of the test methods. With 24 or 25 time steps chosen as the sampling interval, less than 5% of the sequences were rejected. The percentage increased up to 17% with the sampling interval 47 or 50 time steps. The higher rejection percentage was 26% by the Chi-square goodness-of-fit test with the sampling interval 100 steps in the Climate dataset. On the other hand, in all cases much more samples were rejected in the normality tests when the linear interpolation with or without a double sampling rate was used. The results show that compared to linear interpolation, the errors of Bezier curve fitting can be much better approximated by a Gaussian distribution with computable standard deviation.

Statistically, a Gaussian distribution has the property that 95% of the samples fall within the range approximately $\pm 1.96$ standard deviations of the mean. We examine whether the estimated standard errors can provide this confidence interval. From the above sampled sequences approximated by the quadratic Bezier curve fitting, we counted the number of samples where the true error was within $\pm 1.96$ standard deviation using the estimated standard error. As shown in row (a) of Table 1, the percentage of the sequences passing our test was larger than 95% in all the test cases. This test also shows the estimated standard errors can be used to provide the confidence interval.

### 7.3 Performance and Storage Requirement

We now evaluate the performance of our incremental error modeling algorithm. The test was performed on a Linux machine with Intel Core i7-2600 Quad-core CPU, 16 GB of RAM. Since we apply error modeling on the value sequence at each grid point individually, there is no dependency among different grid points and thus the computation can be easily parallelized. As presented in Algorithm 1, the incremental error modeling computation has two major tasks: first, it accumulates partial sums for each time step, denoted as ADD operation; Second, with the accumulated summations, it computes Bezier fitting and the standard error at the end of each sampling interval, denoted as FIT operation. We measured the operation time for each test case with four threads used. As expected, the computation time of each operation was invariant to the length of the sampling interval. The average ADD time was 168 ms, 420 ms and 282 ms respectively for *Isabel*, *Plume* and *climate* dataset; the average FIT time was 204 ms, 505 ms and 325 ms, respectively. The differences of above timings from various time steps and sampling intervals per dataset were within 2% of the mean.

The resulting storage for each test case is shown in Table 2. Given the sampling time interval $n$, the number of sampling intervals $I$ in a total $N$ time steps will be $I = \lfloor N/n \rfloor$. Thus $I+1$ samples will be stored including the first and the last time steps. Since at the end of each sampling interval our error model requires to store the Bezier control point and the standard error for each grid point, if a time step of the flow field is in size of $S$, the total storage required is $(2I \cdot S)$. Furthermore, to obtain more accurate pathline, we store the flow map for each sampling interval and thus incurs additional $(I \cdot S)$ of storage. In total the extra storage in use is $(3I \cdot S)$.

### 7.4 Comparison of Intermediate Trajectory Estimation

With stored flow maps, i.e., end positions sampled from the true pathlines (computed from the full-resolution flow field), we estimate the particle traces in between, using three different methods: 1. **Regular forward tracing with a double sampling rate**: This is the conventional approach to compute pathlines using fourth-order Runge-Kutta integration method (RK4), where the sampled flow fields are linearly interpolated in the time dimension. Double sampling rate was used to further reduce the flow field error.

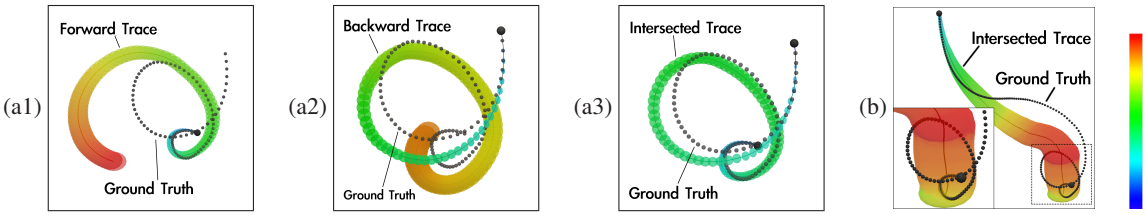


Figure 6: Uncertain forward (a1) and backward (a2) traces from two ends of a true pathline in the Isabel dataset within the sampling interval of 12 time steps. (a3): Our pathline refinement method by intersecting the previous forward and backward traces. (b): Our forward-backward intersected trace with the sampling interval of 24 time steps. Including for the rest of the figures, the color coding on all the estimated traces represents the estimated standard deviation (blue: lower error, and red: higher error). The 1.96 standard deviation is used as the error range.

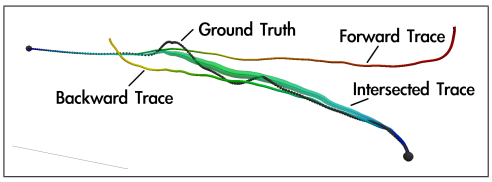


Figure 7: Test on the Plume dataset with the sampling interval of 24 time steps.

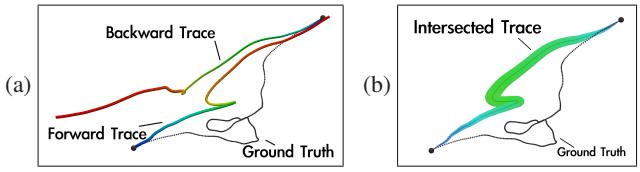


Figure 8: Forward and backward traces (a) and our intersected trace (b) in the Climate dataset with the sampling interval of 50 time steps.

2. **Directly connect**: When particle end positions are given, we connect the beginning and end positions with linear interpolation.
3. **Forward-backward pathline intersection method**: This is our method incorporating the stored flow field with error modeling and the particle end positions.

We compared the accuracy of the above three estimated intermediate traces with true pathlines traced from the full resolution data using RK4 method as the ground truth. For each time step we collected the distance of each estimated particle location to the true location and measured the error by averaging the collected distances from each time step of the trajectory. As shown in Figure 5, our method produced better estimations than the other two methods in all cases with $15\% - 50\%$ less error in average.

### 7.5 Visual Analysis of Pathline Uncertainty

Finally, we visually inspect the trajectories computed from our forward-backward pathline intersection method with the estimated uncertainty measures. Since we model the uncertain particle locations in Gaussian distributions, at each integration step a pair of mean and covariance matrix is generated. To show the uncertainty of the trajectory, for each time step we plot an ellipsoid with the principal axes. To assign the magnitudes for the principal axes, we use 1.96 standard deviations in each corresponding dimension, by assuming that there is a 95% of probability for the true particle location to be enclosed by the ellipsoid. The estimated standard deviations are also used for color coding of both the traces and the ellipsoids.

Figure 6(a1)-(a2) show the respective particle trajectories from two end points of a flow trace going in the opposite directions. As can be seen in Figure 6(a1), where the dotted line is the ground truth, the forward trace initially follows the true trajectory well but then drifts off in the middle of the integration (the line becomes thicker, representing more uncertainty); in Figure 6(a2), the backward trace presents the vortex structure better, but also becomes less accurate as the integration proceeds. The intersection of the above two traces, as shown in Figure 6(a3), is closer to the ground truth than any of the two individual traces and can better preserve the vortex-shaped path. In Figure 6(b), we show the trajectory estimation with the same initial position as Figure 6(a) in the bottom, but using the sampling interval of 24 time steps. In this case the vortex structure is barely seen in the intersected pathline, but the error range is larger, indicating higher uncertainty in the region.

Figure 7 shows an example of the Plume dataset, where a particle moves from left to right and becomes more turbulent in the middle. The forward trace drifts off a lot near the end of the integration, while the backward trace stops in the middle of the true trace. The intersected trace in between is more similar to the true pathline.

Finally, Figure 8 shows a case in the Climate dataset using 50 time steps as the sampling interval. In this case with a larger sampling interval, both the forward and backward traces drift off quickly after the first few integration steps and thus the intersected trace in Figure 8(b) is also not closer to the true pathline. However, the uncertainty of the final trace is shown by its large width, which can help the user interpret the result without being totally misled.

## 8 Discussion

The experimental results in Section 7.1 show that our Bezier curve model can reduce the interpolation error effectively, compared to that using linear interpolation in the same storage. Moreover, the normality tests in Section 7.2 indicate the distribution of errors from Bezier curve fitting is close to be Gaussian, although the number of rejections in the normality tests increased as the sampling intervals became larger. In this case we may still provide a 95% confidence interval with the computed standard errors, as shown in Table 1(a). Whether the normality hypothesis and the confidence interval can be applied to larger sampling intervals for certain data sets requires more investigation and is part of our future work.

The performance of our incremental error modeling algorithm is efficient, as shown in Section 7.3. One limitation is that our incremental algorithm, as listed in Algorithm 1, requires caching four partial sums and the first value $y0$ for each grid point. This means the memory requirement, although remaining constant with respect to different sampling time interval $n$, is not negligible when the size of the flow field is large. We may alleviate this problem by fitting a small spatial region of data values together to reduce the total number of sequences to fit, which is also part of the future work.

The extra storage cost in our framework is close to be three times of the down-sampled data, which consists of three different types of information: Bezier control point reduces the flow field interpolation error and allows the error to be modeled as Gaussian, described by the standard error of the fitting results. The particle end positions, or the flow maps, can avoid error accumulation during uncertain particle tracing. Although one can store flow maps in higher

sampling rates or design an algorithm to adaptively sample pathline trajectories to preserve flow features, our error modeling scheme with forward-backward pathline intersection can further be used to refine the intermediate trajectories whenever the time-varying data are temporally down-sampled, a common practice in the field.

As shown in Section 7.4 and 7.5, our forward and backward pathline intersection algorithm can incorporate particle end positions and the error-modeled flow field to generate more accurate pathline trajectories, with point-wise error estimation indicating the uncertainty. Therefore, although the error of the reconstructed trajectories in the middle portion can still be large especially when passing through turbulent areas, our error estimation of reconstructed pathlines serves as an indication of data reliability, which provides a hint for the scientists to refine the sampling frequency for the interested temporal ranges whenever possible. Finally, although we model the uncertain pathline trajectories in Gaussian distributions, the estimated error range bounded by the 1.96 standard deviation does not always enclose the true path, especially when the integration step is too large. Table 1 row (b) lists the confidence level of the estimated error range, which represents the percentage of trajectory vertices whose errors to the true particle locations were within the corresponding ellipsoid range using the 1.96 standard deviation. We can see that the confidence levels were less than 95% and would drop as the sampling interval increased. On the other hand, since our error modeling for vector fields still provides error estimation in 95% confidence intervals as shown in Table 1(a), we hypothesize that the smaller confidence level in the estimated particle distribution is due to the approximated Gaussian-based particle integration method. When the uncertain particle locations scatter more, the assumption of all the particles in the distribution moving together around the center of the distribution will not hold. In such a case Monte-Carlo based particle integration methods such as the work by Otto *et al.* [17] can be adopted, though with higher storage and computation costs. On the other hand, our Gaussian based pathline estimation still provides an effective qualitative measurement of errors for large sampling intervals.

## 9 Conclusion and Future Work

In this paper we present an error modeling and reduction method for large scale time-varying flow visualization, when it is not possible to process or store data from every time step due to the high computation and storage cost. We show that the errors can be modeled in Gaussian distributions when the data are fitted by Bezier curves with least squares regression, which can be incrementally computed with minimal memory footprint. We then show the applicability of the error model for pathline visualization using existing uncertain pathline computation techniques. With flow maps stored at the end of each sampling intervals, we propose a forward-backward pathline intersection method that can reduce the error of the intermediate pathlines more effectively. Finally we show that our approximated pathlines can reveal more details of the flow structure like vortices while displaying the uncertainty within the sampling intervals. The target applications of our algorithm can be any vector field or pahtline analysis tasks on down-sampled flow fields with the requirement of error indication.

There are many avenues for future work. As discussed in Section 8, while in this work we use a Gaussian-based particle integration method for uncertain pathline computation, we expect more accurate pathline estimation using more sophisticated algorithms. More incremental regression and error modeling methods can be explored. For example, whether we can model errors of all vector components in a joint distribution requires more investigation. While we empirically show that our datasets can be modeled by Bezier-curve functions with Gaussian errors, we expect some datasets may not fit our model. Therefore, we will continue to refine our model, probably with the aid of domain specific knowledge.


## Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. This work was supported in part by NSF grants IIS-1250752, IIS-1065025, and US Department of Energy grants DE-SC0007444, DE-DC0012495, program manager Lucy Nowell.


## References


[1] S. Ameer. *Investigating polynomial fitting schemes for image compression*. PhD thesis, University of Waterloo, 2009.

[2] W. Bischoff, H. Cremers, and W. Fieger. Normal distribution assumption and least squares estimation function in the model of polynomial regression. *Journal of Multivariate Analysis*, 36(1):1–17, 1991.

[3] K. Brodlie, R. AllendesOsorio, and A. Lopes. A review of uncertainty in data visualization. In *Expanding the Frontiers of Visual Analytics and Visualization*, pages 81–109. Springer London, 2012.

[4] P. Bromiley. Products and convolutions of gaussian probability density functions. Technical report, Tina-Vision, 2014.

[5] S. L. Brunton and C. W. Rowley. Fast computation of finite-time Lyapunov exponent fields for unsteady flows. *Chaos*, 20(1):17503, Mar. 2010.

[6] A. Chaudhuri, T.-Y. Lee, B. Zhou, C. Wang, T. Xu, H.-W. Shen, T. Peterka, and Y.-J. Chiang. Scalable computation of distributions from large scale data sets. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pages 113–120, 2012.

[7] G. Chen, K. Mischaikow, R. S. Laramee, and E. Zhang. Efficient morse decompositions of vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):848–862, 2008.

[8] N. Fout and K.-L. Ma. An adaptive prediction-based approach to lossless compression of floating-point volume data. *IEEE Transactions on Visualization and Computer Graphics*, 18:2295–2304, 2012.

[9] W. Greene. The least squares estimator. In *Econometric Analysis*. Prentice Hall, seventh edition, 2011.

[10] G. Haller and T. Sapsis. Lagrangian coherent structures and the smallest finite-time Lyapunov exponent. *Chaos*, 21(2):023115, 2011.

[11] M. Hlawatsch, F. Sadlo, and D. Weiskopf. Hierarchical line integration. *IEEE transactions on visualization and computer graphics*, 17(8):1148–63, Aug. 2011.

[12] C. Johnson and A. Sanderson. A next step: visualizing errors and uncertainty. *IEEE Computer Graphics and Applications*, 23(5):6–10, 2003.

[13] S. Liu, J. Levine, P. Bremer, and V. Pascucci. Gaussian mixture model based volume visualization. *2012 IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pages 73–77, 2012.

[14] A. Love, A. Pang, and D. Kao. Visualizing spatial multivalue data. *Computer Graphics and Applications, IEEE*, 25(3):69–79, 2005.

[15] A. Luo, D. Kao, and A. Pang. Visualizing spatial distribution data sets. In *Proceedings of the Symposium on Data Visualisation 2003*, VISSYM '03, pages 29–38. Eurographics Association, 2003.

[16] K.-L. Ma. In situ visualization at extreme scale: challenges and opportunities. *Computer Graphics and Applications, IEEE*, 29(6):14–19, 2009.

[17] M. Otto, T. Germer, and H. Theisel. Uncertain topology of 3d vector fields. In *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, pages 67–74, 2011.

[18] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. Least squares as a maximum likelihood estimator. In *Numerical Recipes In C: The Art Of Scientific Computing*, number i, chapter 15.1, pages 657–661. Cambridge University Press, 2nd edition, 1992.

[19] D. Schneider, J. Fuhrmann, W. Reich, and G. Scheuermann. A variance based ftle-like method for unsteady uncertain vector fields. In *Topological Methods in Data Analysis and Visualization II*, pages 255–268. Springer Berlin Heidelberg, 2012.

[20] D. Thompson, J. Levine, J. Bennett, P.-T. Bremer, A. Gyulassy, V. Pascucci, and P. Pebay. Analysis of large-scale scalar data using hixels. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pages 23–30, 2011.

[21] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma. In situ visualization for large-scale combustion simulations. *IEEE computer graphics and applications*, 30(3):45–57, 2010.