A Scientific Data Representation Through Particle Flow Based Linear Interpolation

Yu Pan, Feiyu Zhu, Hongfeng Yu
Department of Computer Science and Engineering
University of Nebraska-Lincoln

{ypan, fzhu, yu}@cse.unl.edu

***Abstract*—Scientists often desire interactive visual analytics services to efficiently and effectively study their large-scale scientific data generated from simulations or observations. However, as the volume of scientific data is growing exponentially, it becomes increasingly difficult to achieve this goal for a typical interactive visual analytics system nowadays. The bottlenecks in visual analytics processes manifest in fetching time series data in a continuous manner. Since the changes in scientific datasets over a period of time are usually small and continuous, it is possible to learn an *optical flow* based representation of such dynamics. Therefore, the intermediate time steps of data can be efficiently inferred at run time. However, the existing optical flow determination methods cannot be directly applied to scientific datasets due to the highly complex non-rigid transformations in the feature space of scientific datasets. In this paper, we present a new method, named *particle flow*, that can capture the inherently complex dynamics of scientific datasets. We can effectively reconstruct any intermediate frames by interpolating the starting and ending frames using the resulting particle flow. We have also demonstrated that our approach can be effectively applied in data reduction for scientific datasets. Extensive experiments are conducted to show the accuracy and the efficiency of our approach over existing methods.**

***Index Terms*—scientific data, visualization, data representation**

I. Introduction

Large-scale scientific data visualization has become increasingly important for scientific studies in different fields. However, analyzing large-scale scientific data is still challenging, mainly due to the resources that need to accommodate the sheer size of data. Data reduction is recognized as a feasible solution for visualizing and analyzing large scientific data. Example approaches of data reduction include sampling, compression, and data transformation. For a time-varying scientific dataset, a common strategy is to compress the data in the time domain for data reduction. Data at a certain time point can be treated as a frame. If two key frames can be used to predict the frames between them, we may only need to store the key frames, which can greatly reduce the size of data.

Interpolation methods for time-varying data have been of great interests to researchers. In the field of video compression, for interpolation of time-varying data (e.g., video frames), *optical flow* has been invented to guide the synthesis of frames [6]. This method is often called motion compensation, in which I-frames are the key frames and P-frames are the intermediate frames generated on the fly from I-frames. A typical video compression standard is MPEG, which is a blocked-oriented motion-compensation standard utilizing optical flow between frames. A better quality of optical flow can result in a more accurate interpolation of frames [5]. However, optical flow has not been fully exploited in scientific data visualization yet. The main reason is that traditional optical flow based methods are usually used to track the movement of rigid bodies. In a simulation-generated or sensor-observed scientific dataset, objects or features (such as the movement of fluid) are usually volatile and cannot be described by a rigid body, and thereby the calculation of optical flow becomes a very challenging problem.

In this paper, we present a novel method to generate a modified optical flow within a time range of a scientific dataset. We call this modified method *particle flow*. Given the starting and ending frames of a time range, we can reconstruct any intermediate frame between them using the starting and ending frames, as well as the resulting particle flow. The main contributions of our paper are as follows:

1) Our method does not rely on any feature description and comparison as in traditional optical flow based methods, and can be adapted to complex transformations of features across data frames. Therefore, our method is suitable for capturing and retaining feature dynamics in scientific datasets, and facilitates us to effectively compress the original large dataset.
2) We have conducted an extensive experimental study to demonstrate the effectiveness of our approach and the higher-quality reconstruction results compared to the existing methods. We have also explored the main parameters and provided guidance for a practical usage of our approach.

II. Related Work

Researchers have developed many compressed data representations for large-scale scientific datasets. Thompson et al. [16] used hixels as a compact and information-rich representation of large-scale data. Liu et al. [11] used per voxel Gaussian mixture models (GMMs) as a data representation to reduce the amount of data needed, while still enabling real-time data access and rendering. Discrete Wavelet Transform (DWT) has been used to reduce the data into a limited number of coefficients and has been used for volume rendering [14]. Chaudhuri et al. [2] reduced raw data into a compressed structure using an encoding technique. Chen et al. [3] improved data accuracy of temporally down-sampled data and modeled the error with a Gaussian distribution. Distribution-based representations have been used to alleviate artifacts and were able to provide a confidence measure in visualization tasks [17]. Most of these methods consider the spatial representations of volume data, without considering the possible temporal compression. For temporal compression, a difference based method can be used to compress the data by taking advantage of the similarity of temporally neighboring data [13]. Although this method can reduce the size of data, the differences still need to be stored.

Traditional frame interpolation methods contain two steps, motion estimation and pixel synthesis. Motion is commonly represented using optical flow. In the field of computer vision, optical flow is an effective way to reduce data temporally using data at two time points to predict the data between these time points. The quality of flow-based interpolation depends on the accuracy of the flow, which is often challenged by large and fast motions. There are many optical flow techniques available. The Gunner Farnebäck method approximates the neighborhood of frames by polynomials [4]. The Lucas-Kanade method [12] regards image patches and an affine model for the flow field. The Horn-Schunck method [7] optimizes a function based on residuals from the brightness constancy constraint and a particular regularization term representing the expected smoothness of flow field. FlowNet is capable of finding optical flow estimation using a layer that correlates feature vectors at different image locations [5], [8]. PWCNet uses warped features of images to construct a cost volume, which is processed by a neural network to find the optical flow [15]. Zhou et al. have proposed a method that employs neural networks to warp input pixels to create a novel view. Their method can warp individual input frames and blend them together to produce a frame between the input ones. The deep voxel flow algorithm has been used to address the problem of synthesizing new video frames, either between frames or behind frames. SuperSloMo [9] uses a bi-directional optical flow. The flows are linearly combined to approximate the intermediate optical flow. However, optical flow has not been fully exploited for scientific applications.

III. Particle Flow Based Representation

We first introduce the background of optical flow (Section III-A). Then, we formalize the problem (Section III-B) and introduce the framework of our model (Section III-C). We present the assumptions of our model (Section III-D), and use a modified optical flow for linear interpolation (Section III-E). We detail the objective function used to train our model (Section III-F).

A. Background

1) A Brief Introduction to Optical Flow: Optical flow expresses patterns of motion observed in a visual scene (e.g., video) and can be represented as a vector field where each point has a direction indicating the motion of this point. For each adjacent pair in a series of continuously changing images or frames, the general goal of the construction of optical flow is to compute the velocity of each point. This velocity information typically is obtained by estimating the movement of feature objects across frames. Traditional approaches to determine optical flow include phase correlation, block-based methods, differential methods, and discrete optimization methods based on prior definitions of features. Recently, learning-based methods become increasingly popular and some of them have outperformed traditional methods, in particular when features are not well defined [6].

Once we obtain optical flow, it can facilitate us to conduct a set of computer vision operations, such as motion detection, object segmentation and tracking, distance detection, and so on. Optical flow can also be used in video compression, in which the intermediate video frames are inferred based on the key frames and their optical flows. This application is closely related to our research and it is interesting to explore the possibility of applying optical flow based methods to derive compressed scientific data representations.

2) Research Challenges: An intuition is that we can extend optical flow to express the dynamics of scientific data, given the data similarity between videos (e.g., a series of 2D images) and scientific data (e.g., a series of 3D volumes). However, this is a non-trivial task.

For traditional optical flow methods, they perform optimally when a feature is close to a rigid body, where the feature may move dynamically but its attributes (e.g., shape and texture) are relatively stable. This is most applicable for computer vision applications. However, in a scientific dataset, features can be dramatically changed in their attribute space, and typically are not rigid bodies. Therefore, if we directly apply traditional methods, the resulting optical flow cannot meaningfully capture the underlying dynamics of features. For example, the Gunner Farnebäck method [4] uses a quadratic polynomial to approximate the neighborhood of each pixel, and uses polynomial expansion to bring in the term of optical displacement (i.e., optical flow) along each direction. The resulting optical flow is subject to the minimization of an objective function. The basic assumption of Gunner Farneback's algorithm is that the feature in a neighborhood is stable across frames and thus can be expressed by the same quadratic function. Again, this assumption may not apply to scientific datasets because of typical non-rigid transformations between two data frames. This can prevent Gunner Farnebäck's algorithm from managing to match corresponding points.

Recent learning-based optical flow methods train end-to-end deep models to output optical flows using the input of two images. One example is PWC-Net [15] that first derives an image pyramid using Convolutional Neural Networks (CNN), and warps the second image to the first one for each level of images pairs in the image pyramid. A cost volume layer is then applied to calculate the dissimilarity of the first image and the warped version of the second image. Finally, an

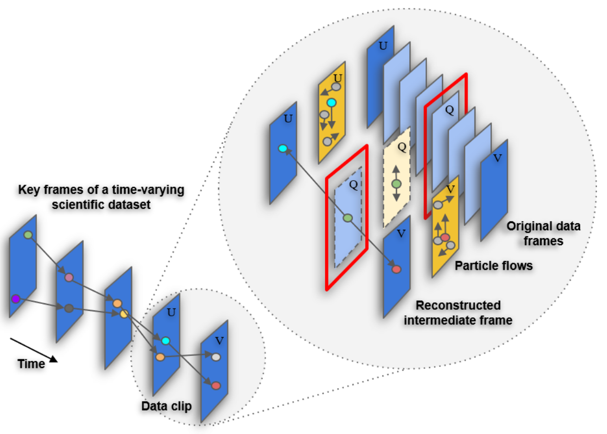

Key frames of a time-varying scientific dataset
Time
Data clip
U
V
Q
Original data frames
Particle flows
Reconstructed intermediate frame

Fig. 1: An illustration of our framework. Planes U and V represent the starting and ending frames of a data clip of a time-varying scientific dataset. We aim to learn the particle flows (i.e., a modified version of optical flows) for both Planes U and V (in dark yellow). The particle flow for an intermediate frame Q (in light yellow) is estimated based on the flows on Planes U and V. Then, the values on Plane Q (in light blue with dashed line) are estimated as a linear interpolation of the corresponding values in Planes U and V. Finally, we compare the similarity of the reconstructed values and original ones on Plane Q (in red windows). The learned particle flows are subject to minimizing the similarity measurement.

optical flow estimator layer is applied to output the final result for optical flow. The learning based methods mainly have two disadvantages. First, they still rely on some forms of feature comparison (e.g., the cost volume layer in PWC-Net) to determine optical flows. Thus, they are most suitable for those scenarios with rigid transformations. Second, as supervised learning methods, they require a large amount of optical flows as background knowledge, which may not be feasible for scientific datasets.

We develop a modified version of optical flow, named *particle flow*, which can exploit the advantages but reduce the limitations of original optical flow approaches for scientific applications. In our experimental study, we adopt Gunner Farnebäck's algorithm and PWC-Net as baseline to obtain the optical flows and compare their performance with our method. For comparison, we also bring in a naive fade-in and fade-out method that linearly interpolates key data frames to generate intermediate frames. Our particle flow solution can achieve results superior to the existing methods for scientific datasets.

B. Problem Formalization

Given a series of n data frames $\{d_0, d_1, ..., d_t, ..., d_n\}$, our goal is to learn a concise representation of this series, a tuple (D, F), where $D = \{d_0, d_n\}$ and $F = \{f_{0\to n}, f_{n\to 0}\}$. Here, d_0 and d_n are the starting and ending frames, and $f_{0\to n}$ and $f_{n\to 0}$ are the particle flows between the starting and ending frames, respectively.

C. The Framework of Our Model

Our model can be considered as a representation learning framework. For the representation of scientific data clips (i.e., a series of time frames), there are two complementary components:

1) Compression: obtain an appropriate optical flow based representation.
2) Decompression: reconstruct the original data clip using the representation.

We can plug-in existing optical flow methods for the compression component and conduct a linear interpolation along the time axis to infer intermediate data frames for decompression, which, however, is less optimal given the limitations of existing optical flow methods for scientific datasets. Alternatively, in this work, we holistically combine the compression and decompression components, and directly consider the target particle flows as trainable parameters that can be learned through model optimization, which can facilitate us to capture and represent the dynamics of scientific datasets. Figure 1 illustrates the framework of our model.

D. Assumptions of Our Model

For numerous types of scientific datatsets, such as those in combustion science, plasma physics, and atmospheric science, whether they are synthesized through simulations or collected from sensors, we can consider moving particles as the cause of the dynamics of these datasets. The spatial distributions and temporal dynamics of scientific datasets are the reflection of the motion of one or several kinds of particles that influence various attributes of systems. With this idea in mind, we can model a system as flows of particles in which each particle has a different trajectory. We can even go further by imagining some sort of abstract particles that fill the space. Here, we call particles as movable pixels (or voxels for 3D volume). We can think of the attributes of a dataset labeled on each pixel such that we consider the attributes as the function of pixel, instead of the function of time and space. Borrowed from the terminology in the computer vision community, we name the trajectories of these movable pixels as particle flow, which we aim to learn from scientific datasets in this work. Particle flow can be treated as a modified version of optical flow. However, particle flow is different from optical flow in that particle flow estimates the movement of particles at the pixel level instead of estimating the movement of rigid objects at a higher level. We believe particle flow is better suited for scientific datasets where the movement is much more dynamic.

Since the time span of the data we try to learn the representation from is relatively small, we can make the following assumptions:

1) All the movable pixels in the starting frame still exist in the ending frame.
2) The trajectories of these movable pixels within a small time span are linear, and thereby can be expressed as optical flow, which is a vector field.

3) The optical flow field changes smoothly with respect to the spatial coordinates.
4) The optical flow field changes linearly within a small spatial area.
5) The attributes on each movable pixel will not change drastically within a small time span and also change linearly within the time span.

Based on these assumptions, we can learn a much simpler representation of the original data clips. The resulting representation consists of only the starting and ending frames and the optical flows between the starting and ending frames.

E. Particle Flow Based Linear Interpolation

Once we get this representation consisting of the starting frame d_0, the ending frame d_n, the forward flow $f_{0\to t}$, and the backward flow $f_{t\to 0}$, we can synthesize all the intermediate data frames using linear interpolation at runtime. For a data frame d_t, we can get an estimate $\hat{d}_t$ as follows:

$$\hat{d}_t = (1-\frac{t}{n}) \odot warp(d_0, f_{t\to 0}) + \frac{t}{n} \odot warp(d_n, f_{t\to n}), \quad (1)$$

where $warp(\cdot,\cdot)$ is a warping operation of the first augment given the particle flow as the second argument. We use bi-linear interpolation when a warped pixel falls in between the discrete pixels. The $\odot$ is an element-wise multiplication that conducts a linear interpolation of the warped frames from both directions based on the time t. This linear operation is in accordance with our Assumption 4 in Section III-D.

Assume that we have the particle flows $f_{0\to n}$ and $f_{n\to 0}$ between the starting and ending frames, to get the particle flows for an intermediate frame at t, $f_{t\to 0}$ and $f_{t\to n}$, we need a way to estimate them. Given our assumption that the particle flow field is smooth and changes linearly within a small spatial area, for each pixel in an intermediate frame, we can estimate its flow based on the corresponding flow at the same position in the starting and ending frames:

$$\hat{f}_{t\to 0} = -w_t \frac{t}{n} \odot f_{0\to n} + (1-w_t)\frac{t}{n} \odot f_{n\to 0} \quad (2)$$

$$\hat{f}_{t\to n} = w_t(1-\frac{t}{n}) \odot f_{0\to n} + (1-w_t)(1-\frac{t}{n}) \odot f_{n\to 0} \quad (3)$$

where w_t is the mixture weight of flows for both directions:

$$w_t = \frac{f_{n\to t} \odot (1-\frac{t}{n})}{f_{n\to t} \odot (1-\frac{t}{n}) + f_{0\to t} \odot \frac{t}{n}}. \quad (4)$$

Here, we use the flows from two positions as reference points and the mixture weight indicates that the impact of the flows reduces as the distance between the reference point and the considered point increases. Figure 2 illustrates our ideas on estimating the flows for intermediate frames.

F. Particle Flow Learning

We aim to learn the forward flow $f_{0\to t}$ and the backward flow $f_{t\to 0}$ from the original series of n data frames $\{d_0, d_1, ..., d_t, ..., d_n\}$. We transfer the learning process to solving an optimization problem, where we define a set of objective

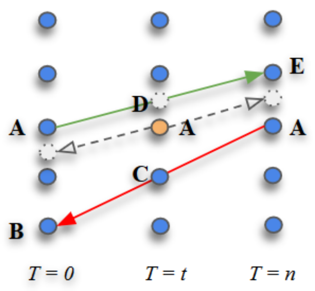


Fig. 2: An illustration of intermediate particle flow estimation. The circles in solid line represent the pixels at integer positions and the circles in dashed line represent virtual pixels at non-integer positions. The bi-directional particle flows of the yellow pixel **A** in the intermediate frame at $T = t$ is estimated based on the particle flows of **A**s in the frames at $T = 0$ and $T = n$. The particle flow of **A** in the frame at $T = 0$ points to **E** in the frame $T = n$ through a virtual point **D** in the frame at $T = t$. The particle flow of **A** in the frame at $T = n$ points to **B** in the frame at $T = 0$ through **C** in the frame at $T = t$. We first calculate the bi-directional particle flows for **C** and **D**, and estimate the particle flows for **A** in the frame at $T = t$ by linearly interpolating them based on their distances to **A**. Again, the estimated particle flows for **A** point to virtual pixels in the frame at $T = 0$ and $T = n$, which can be further estimated using bilinear interpolation.

functions according to our assumptions in Section III-D and minimize these functions with respect to the forward and backward flows.

1) Objective Functions: Based on our discussion in Section III-E, we can easily determine that one term of our objective function is the *reconstruction loss*:

$$\mathscr{L}_r = \frac{1}{m-1}\sum_{t=1}^{m-1}\frac{1}{N}\|d_t - \hat{d}_t\|_2 \quad (5)$$

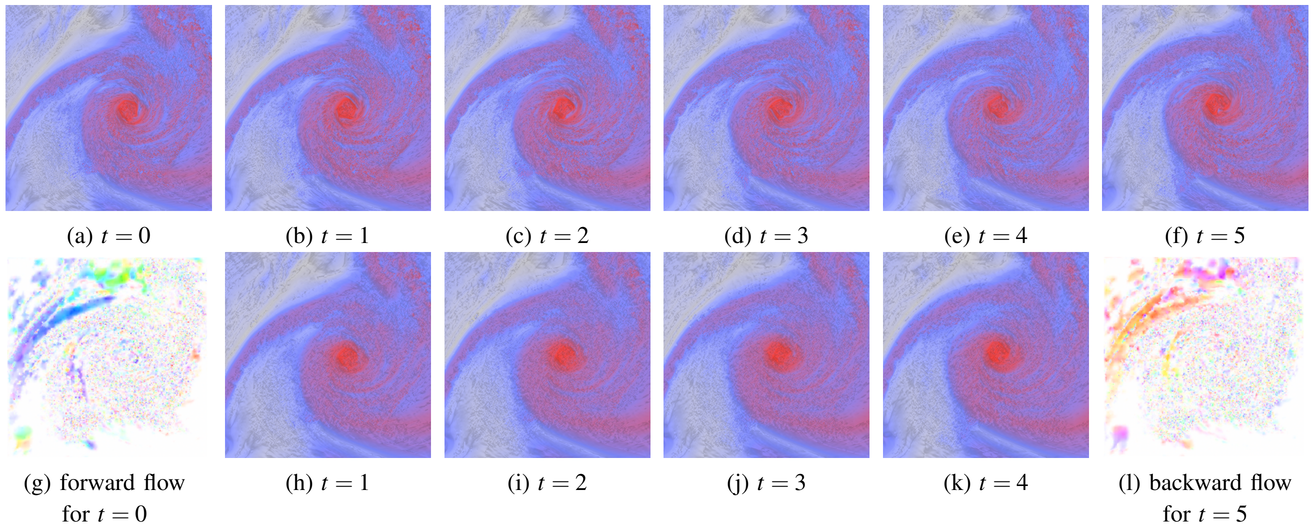
where m is the number of all the intermediate data frames, N is the number of pixels in each data frame, and $\|\cdot\|_2$ is L^2 norm. $\mathscr{L}_r$ denotes the average difference between reconstructed data frames and original data frames.

Since we assume the particle flows of both directions are smooth, we also add a *smoothness loss*:

$$\mathscr{L}_s = \|\Delta f_{0\to n}\|_1 + \|\Delta f_{n\to 0}\|_1 \quad (6)$$

where we compute the Laplacian of both forward and backward flows to measure their smoothness. A smaller Laplacian value indicates a smaller local variation of a flow. We neglect the direction of Laplacian, and add their L^1 norm together to measure the combined smoothness of both flow directions.

In addition, as we assume the trajectories of the movable pixels are linear, this means for each of the movable pixels in the intermediate frames, the flows in both directions, $f_{t\to 0}$ and $f_{t\to n}$, should be opposite. Here, we make the angle between the


(a) $t=0$ (b) $t=1$ (c) $t=2$ (d) $t=3$ (e) $t=4$ (f) $t=5$

(g) forward flow for $t=0$ (h) $t=1$ (i) $t=2$ (j) $t=3$ (k) $t=4$ (l) backward flow for $t=5$

Fig. 3: Original and reconstructed data frames for the Isabel hurricane dataset. In the first row, the images a to f show the original data frames from $t=0$ to $t=5$. In the second row, the images g and l show the particle flows for $t=0$ and $t=5$, and the images h to k show the reconstructed intermediate data frames. In each image of data frame, higher vapor values correspond to red or purple regions, and lower vapor values correspond to blue or gray regions. In each image of particle flow, a vector value is mapped to a unique pseudo color. The reconstructed frames are visually close to the original frames.

two vectors as near to 180° as possible and introduce *trajectory loss*:

$$\mathscr{L}_t = -\frac{dot(f_{t\to n}, f_{t\to 0})}{\|f_{0\to n}\|_2 \|f_{n\to 0}\|_2} \tag{7}$$

where the numerator computes the pixel-wise dot product of the flows that is divided by the pixel-wise L^2 norm of the flows from both directions. This gives the cosine of the angle of the two vectors. An angle closer to 180° leads to a smaller $\mathscr{L}_t$.

At last, as we assume the attributes on each movable pixel will not change drastically, we warp the original starting frame d_0 to estimate the end frame using $f_{n\to 0}$, and also warp the original ending frame d_n to estimate the starting time point using $f_{0\to n}$. Then, we compare the estimated frames with the original ones, which we name as *attribute loss*:

$$\mathscr{L}_a = \frac{1}{N}(\|d_0 - warp(d_n, f_{0\to n})\|_2 + \|d_n - warp(d_0, f_{n\to 0})\|_2) \tag{8}$$

where N is the total number of pixels in each data frame.

Thus, our final objective function is:

$$\mathscr{L} = \mathscr{L}_r + \lambda_s \mathscr{L}_s + \lambda_t \mathscr{L}_t + \lambda_a \mathscr{L}_a \tag{9}$$

where λ_s, λ_t, and λ_a are hyperparameters that control the quota of each term in the objective function. We empirically set $\lambda_s = 0.6$, $\lambda_t = 0.1$, and $\lambda_a = 0.1$, which provides appropriate results in our work.

2) Optimization: Our optimization model sets the forward and backward particle flows $f_{0\to n}$ and $f_{n\to 0}$ as model parameters, and optimizes the objective function $\mathscr{L}$ to learn the particle flows. The resulting $f_{0\to n}$ and $f_{n\to 0}$, together with the first and last frames d_0 and d_n, will be used as a concise representation of the entire set of data frames $\{d_0, d_1, ..., d_t, ..., d_n\}$.

To solve this optimization problem, we use the conventional gradient descent method. We initialize $f^0{}_{0\to n}$ and $f^0{}_{n\to 0}$ using random vector values, and then iteratively refine them. Specifically, in each iteration i, we have

$$f^i{}_{0\to n} = f^{i-1}_{0\to n} - \gamma \frac{\partial \mathscr{L}}{\partial f^{i-1}_{0\to n}}, \tag{10a}$$

$$f^i{}_{n\to 0} = f^{i-1}_{n\to 0} - \gamma \frac{\partial \mathscr{L}}{\partial f^{i-1}_{n\to 0}}, \tag{10b}$$

where γ is a learning rate controlling the convergence rate. We stop the learning process when the refinement is less than a specified threshold. In our implementation, we use the built-in gradient descent functionality of TensorFlow [1] to carry out this optimization process.

IV. Experimental Results

In this section, we first introduce the datasets used in our experiments and the implementation details in Section IV-A, and conduct an extensive experimental study on the effectiveness of our model in Section IV-B. We show the experimental results for the efficiency of our model in Section IV-C, and explore the impacts of several key factors of our model in Section IV-D.

A. Datasets and Implementation Details

We have tested our algorithm on two datasets, an Isabel hurricane dataset and a vortex dataset. The Isabel hurricane dataset was generated by the National Center for Atmospheric Research (NCAR) and made available through the IEEE Visualization 2004 Contest. It has a $500 \times 500 \times 100$ spatial resolution and 48 time steps in total. We use the variable vapor of this dataset in our test. The data value ranges between 0 and 0.02368. The vortex dataset is generated from a pseudo-spectral simulation of vortex structures. It has a $128 \times 128 \times 128$ spatial resolution and 100 time steps in total. To facilitate our comparison between our method and the existing methods, for the vortex dataset, we use a 2D spatial slice at a fixed position from each time step to obtain a 2D data series in our study. The data value ranges between 0 and 7.5.

We learn the particle flows using Python 3.6 and Tensorflow 1.2 [1] on a single machine that has an Intel Core i7-6700K CPU, an NVIDIA GeForce GTX 980 Ti GPU, and 16GB DDR4 memory. For each dataset, we choose a different length of data clip to test our method. For each data clip, we optimize our model using Adam optimizer [10] with a learning rate γ of 0.001 for 1000 epochs. For the resulting representation (i.e. the starting and the ending frames, and the forward and backward flows), we further compress them using Run-Length Encoding (RLE) to obtain a higher compression ratio.

For each dataset, since the majority of the variable values fall near 0 that is the minimal value, it is difficult to learn an optical flow directly from the raw data. We can think of those small values as noises affecting the accuracy of our learning of meaningful dynamics. Hence, we first preprocess the datasets by clipping the values within specified lower and upper bounds. The experiment results show this preprocessing procedure can enhance the effectiveness of our method. We will also discuss the impact of different choices of the lower and upper bounds on the results in Section IV-D3. Otherwise, if not specified explicitly, we validate our methods using 8-frame data clips for the vortex dataset with a lower bound 0.5 and an upper bound 7.5, and using 6-frame clips for the Isabel hurricane dataset with a lower bound 0.001 and an upper bound 0.016.

Figure 3 shows a demonstration of our method on the Isabel hurricane data. Given an original sequence of data frames in Figure 3(a) to (f), we learn the forward and backward particle flows, as shown in Figure 3(g) and (l). We reconstruct the intermediate frames, Figure 3(h) to (k), using the starting and ending frames, as well as the particle flows. We can perceive that the reconstructed frames are visually close to the original frames in Figure 3.

B. Effectiveness of Our Method

We qualitatively and qualitatively study the feasibility and effectiveness of our method. Our experiments show that our method can learn a reasonable representation for a data clip.

1) Reconstruction Accuracy: We first compare the reconstruction accuracy of our method with the existing optical flow approaches including the PWC-Net method [15] and the Gunner Farnebäck method [4]. We also consider a naive fade-in and fade-out method that constructs an intermediate frame by a simple linear interpolation of the starting and ending frames.

We use three different metrics to measure the similarity between reconstructed frames and original frames:

- Peak Signal-to-Noise Ratio (PSNR):

$$PSNR = 20 \cdot log_{10} \frac{MAX}{\sqrt{MSE}},$$

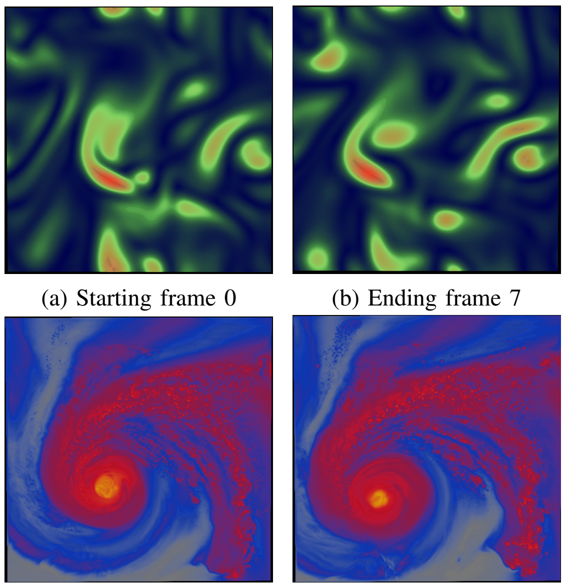
TABLE I: Comparison of the reconstruction accuracy.

The vortex dataset

Method	PSNR	SSIM	NMSE
PWC-Net	24.86	0.77	0.22
Gunner Farnebäck	22.27	0.70	0.29
Fade-In-Fade-Out	24.36	0.74	0.23
Our Method	33.40	0.96	0.07

The Isabel dataset

Method	PSNR	SSIM	NMSE
PWC-Net	26.83	0.71	0.08
Gunner Farnebäck	26.57	0.66	0.09
Fade-In-Fade-Out	25.44	0.57	0.11
Our Method	31.11	0.82	0.05


(a) Starting frame 0 (b) Ending frame 7

(c) Starting frame 0 (d) Ending frame 5

Fig. 4: The first row shows the original starting and ending frames, 0 and 7, of the vortex dataset. The second row shows the original starting and ending frames, 0 and 5, of the Isabel dataset.

where *MAX* represents the maximum value in the original frame and *MSE* indicates Mean Square Error between the original and reconstructed frames. PSNR measures the normalized pixel-wise differences with respect to the maximum value.

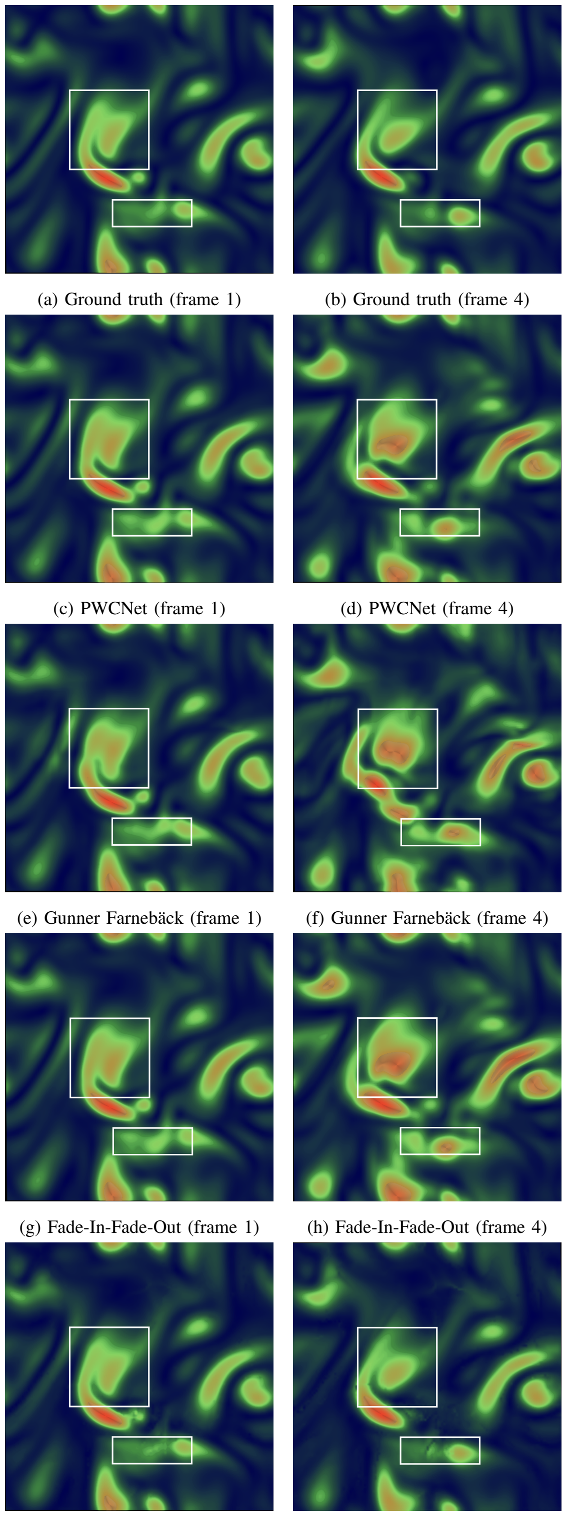
- Structural Similarity Index (SSIM) [18]:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

where x represents the original frame, y represents the reconstructed frame, μ_x and μ_y indicate the mean values of x and y, σ_x and σ_y indicate their standard deviations, and c_1 and c_2 are two variables used to stabilize the division with a weak denominator. SSIM measures the human perceived difference between x and y.

- Normalized Mean Square Error (NMSE):

$$NMSE = \frac{MSE}{\mu_x\mu_y},$$



(a) Ground truth (frame 1) (b) Ground truth (frame 4)

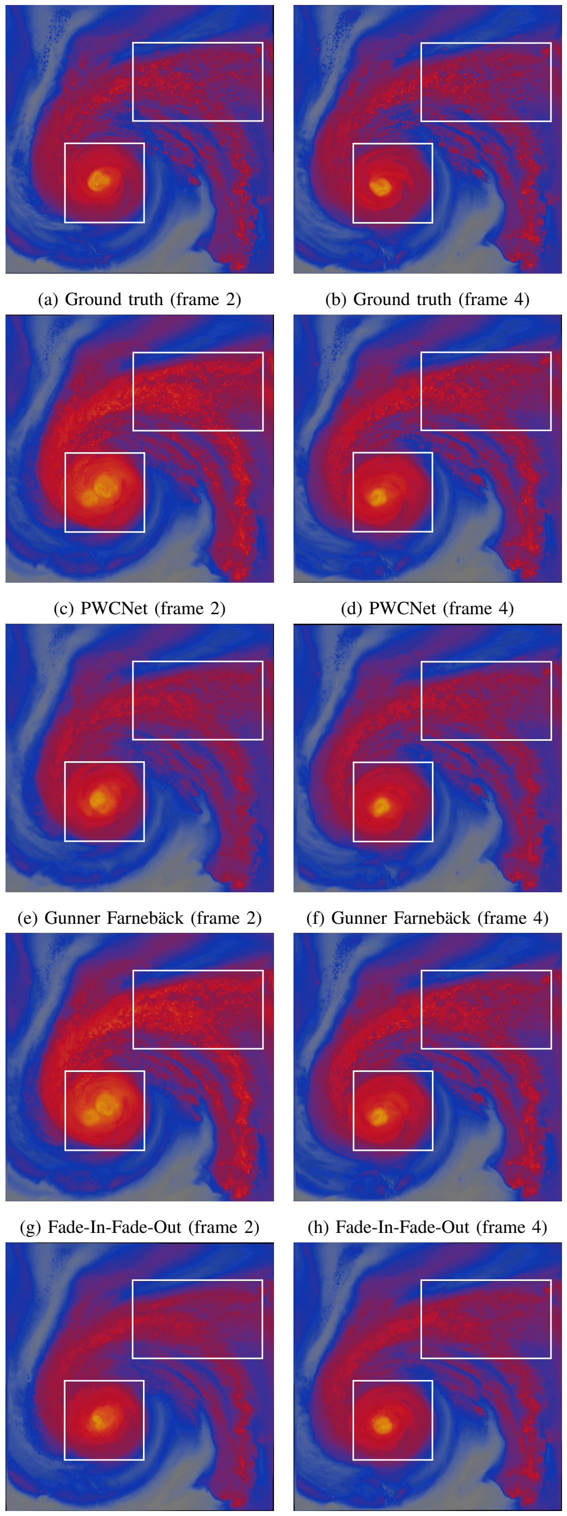
(c) PWCNet (frame 1) (d) PWCNet (frame 4)

(e) Gunner Farnebäck (frame 1) (f) Gunner Farnebäck (frame 4)

(g) Fade-In-Fade-Out (frame 1) (h) Fade-In-Fade-Out (frame 4)

(i) Our method (frame 1) (j) Our method (frame 4)

Fig. 5: The first row shows the original frames 1 and 4 of the vortex dataset. The rest of the rows show the reconstructions of the frames 1 and 4 based on the starting and ending frames using different methods.



(a) Ground truth (frame 2) (b) Ground truth (frame 4)

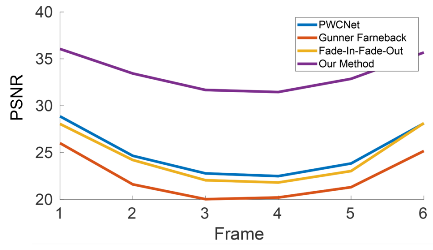
(c) PWCNet (frame 2) (d) PWCNet (frame 4)

(e) Gunner Farnebäck (frame 2) (f) Gunner Farnebäck (frame 4)

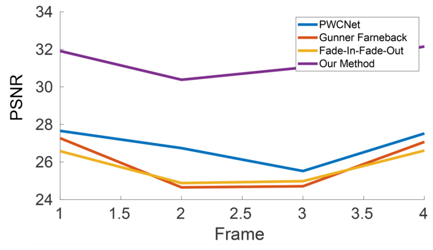
(g) Fade-In-Fade-Out (frame 2) (h) Fade-In-Fade-Out (frame 4)

(i) Our method (frame 2) (j) Our method (frame 4)

Fig. 6: The first row shows the original frames 2 and 4 of the Isabel dataset. The rest of the rows show the reconstructions of the frames 2 and 4 based on the starting and ending frames using different methods.



Method	Starting	1	2	3	4	5	6	Ending
PWC-Net	-	28.87	24.66	22.78	22.49	23.84	28.12	-
Gunner Farneback	-	26.01	21.61	20.03	20.21	21.31	25.17	-
Fade-In-Fade-Out	-	28.06	24.21	22.05	21.81	23.03	28.14	-
Our Method	-	36.06	33.44	31.68	31.46	32.87	35.68	-



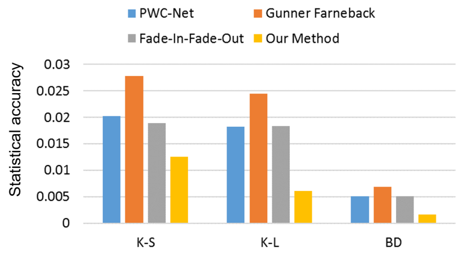
Method	Starting	1	2	3	4	Ending
PWC-Net	-	27.66	26.74	25.52	27.52	-
Gunner Farneback	-	27.27	24.65	24.71	27.07	-
Fade-In-Fade-Out	-	26.58	24.88	24.98	26.61	-
Our Method	-	31.91	30.38	31.03	32.15	-

Fig. 7: Comparison of the PSNR on the vortex dataset (top) and the Isabel dataset (bottom) by frames.

where x represents the original frame, y represents the reconstructed frame, and μ_x and μ_y indicate the mean values of x and y. NMSE measures the normalized pixel-wise differences with respect to the mean value of the original and reconstructed frames.

Table I demonstrates the reconstruction accuracy for both datasets. For each dataset and each method, we reconstruct each intermediate frame within a data clip using the starting and ending frames, compute each metric for each reconstructed frame, and calculate their average. For PSNR or SSIM, the higher a value is, the higher the accuracy is, and contrary for MSE. We can see that our method outperforms all the baseline methods in terms of all the metrics.

Figure 4 shows the original starting and ending frames for the vortex and Isabel hurricane datasets using volume rendering. Figures 5 and 6 show a qualitative comparison of different methods on the reconstructed frames of both datasets. The boxes highlight a few selected differences between the ground truth and the reconstructions of different methods. We can clearly see that our method can reconstruct intermediate frames that are most visually similar to the ground truth. For the vortex dataset, all methods can obtain appropriate reconstructions of the frame 1 that is close to the starting frame 0. However, for the frame 4 that is in the middle of the data clip, the existing methods tend to generate a reconstruction that is a mixture of the starting and the ending frames. This is because these methods mainly compare the features in the starting and ending frames to generate the optical flows that might not capture highly intermittent transient features. Our method makes use



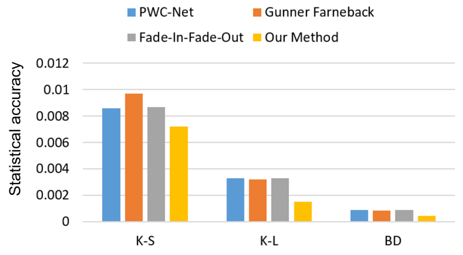


Fig. 8: Comparison of the statistical accuracy of reconstruction on the vortex dataset (top) and the Isabel dataset (bottom).

of the intermediate frames, and thereby can generate a more accurate optical flow and the corresponding reconstructions. The similar results can be also perceived in Figure 6.

The quality of reconstruction also depends on where an intermediate frame is. We conduct experiments to demonstrate the reconstruction accuracy as a function of time steps. Figure 7 shows the PSNR results for both datasets. We can see that the more a frame is close to one end, the more accurate the reconstruction result is. This is because of the fact that our linear trajectory assumption in Section III-D is not perfect, and the pixels on an intermediate frame far from both ends tend to deviate from the linear trajectory. However, we can see even the frames with the lowest reconstruction accuracy using our method significantly outperform those frames with the highest reconstruction accuracy using other baseline methods.

2) Statistical Accuracy: From the data analyst's perspective, the statistical features are equally important. Here we study the statistical accuracy of our method. Specifically, we conduct experiments to show how well the distribution of values is recovered for each time frame in terms of Kolmogorov-Smirnov (K-S) distance, Kullback-Leibler (K-L) divergence, and Bhattacharyya Distance (BD). All the metrics indicate higher statistical accuracy when they are smaller. The possible best value for K-S and K-L is 0, meaning the original distribution and the reconstructed one are the same.

We compare our method with other baseline methods. Figure 8 illustrates the comparison results. We can see our method outperforms all the baseline methods with respect to all the metrics and can achieve a high distribution similarity for each dataset. For example, for the vortex dataset, the K-S value of our method outperforms PWC-Net, Gunner Farnebäck, and Fade-In-Fade-Out by 63%, 120%, and 50%, respectively. Similar comparisons can be also obtained for the other metrics. For both datasets, the K-S and K-L values of our method are closer to 0, indicating the reconstructed frames of our method is very similar to the original ones.

C. *Efficiency of Our Method*

One of our objectives is to get a concise representation of a data clip, which can be considered as a compression version of the original one. As our method is a lossy compression method, so intuitively, the higher the compression ratio is, the less information will be kept. Here we compare the reconstruction accuracy under different compression ratio. Table II illustrates this effect as the length of the data clip increases.

TABLE II: Reconstruction accuracy of the vortex dataset and the Isabel dataset under different compression ratio.

The vortex dataset

Clip length	Compression ratio	PSNR	SSIM	MSE
4	25.3%	43.636	0.998	0.015
5	21.1%	41.087	0.991	0.020
6	18.1%	38.215	0.981	0.038
7	15.8%	35.091	0.965	0.060
8	14.1%	33.401	0.956	0.075
9	12.7%	31.640	0.940	0.096
10	11.5%	30.456	0.925	0.111

The Isabel dataset

Clip length	Compression ratio	PSNR	SSIM	MSE
4	25.3%	33.036	0.894	0.041
5	21.1%	31.625	0.847	0.051
6	18.1%	31.117	0.824	0.055
7	15.8%	30.432	0.793	0.060
8	14.1%	29.386	0.749	0.068
9	12.7%	28.78	0.728	0.073
10	11.5%	28.16	0.710	0.080

Typically, a PSNR value above 30 indicates a reasonable reconstruction result. For the vortex dataset, the reconstruction of our method is closer to the original when the compression ratio is above 11.5%. For the Isabel dataset, to obtain reasonable reconstructions, the compression ratio is ideally above 15.8% for our method.

We also compare the compression ratio between our method and the Run-Length Encoding (RLE) method that is a conventional data reduction method. We use RLE to directly compress each frame within a specified clip length.

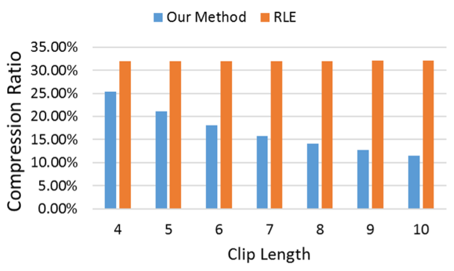


Fig. 9: Comparison of the compression ratio between our method and RLE on the vortex dataset.

Figure 9 shows a comparison of the compression ratio between our method and RLE on the vortex dataset. We can see that the compression ratio of our method decreases dramatically as the clip length increases. In contrast, RLE keeps almost the same compression ratio with increasing clip length. Also, our method has a much better compression ratio than RLE without losing much accuracy.

D. *Parameter Analysis*

In this section, we further study the impact of various factors on the final results, which will be helpful for us to understand some of the attributes of our model.

1) Impact of Temporal Resolution: For a data clip with a fixed time span (i.e., with the same starting and ending frames), we test different temporal resolutions for the intermediate data frames. That is we learn the particle flows using different numbers of intermediate data frames, and compare their reconstruction accuracy. These selected intermediate data frames are evenly distributed within the time span.

Table III shows the result using the vortex dataset for a clip length of 8, which agrees with our intuition: the more intermediate frames we use during learning, the better the quality of the representation is. For the lowest resolution, we use only the starting frame and the ending frame for learning and thus our objective function has no $\mathscr{L}_r$ term.

TABLE III: Impact of temporal resolution on reconstruction accuracy for the vortex dataset.

# of Intermediate Frames	PSNR	SSIM	MSE
0	24.804	0.770	0.206
1	30.172	0.833	0.099
2	32.012	0.941	0.085
3	33.006	0.951	0.079
4	33.401	0.956	0.075

For the vortex data, Table III shows that our method needs at least two intermediate supporting frames to gain appropriate reconstruction accuracy (i.e., PSNR is larger than 30). When using more intermediate frames, the reconstruction accuracy keeps at a higher rate. This means there are more dynamics embedded in the intermediate frames without which we can not fully recover the whole sequence.

2) Impact of Spatial Resolution: We can reduce the spatial resolution of the optical flows and get a more concise representation. When we need to use the optical flows, we can just upsample them to the original resolution by bilinear interpolation. This is a common practice employed in existing optical flow applications [15]. We are also interested in the impact of different spatial resolutions for our particle flow method. Table IV illustrate the reconstruction accuracy for different choices of resolution using our method.

TABLE IV: Impact of spatial resolution on reconstruction accuracy for the Isabel dataset.

Downsampling rate	PSNR	SSIM	MSE
1	33.505	0.956	0.074
3	31.037	0.907	0.099
5	30.04	0.848	0.111

We downsample our particle flow by 1-5 times and then upsample it using bilinear interpolation method for reconstructing the clips. Generally, we achieve a lower reconstruction error rate with a finer resolution of the optical flow. This result matches our expectation and shows this problem cannot be simply considered as the traditional optical flow problem as the particles have much more complicated dynamics than rigid moving body and can only be learned independently.

3) Impact of Data Threshold: Finally, we study the impact of different choices of lower and upper bounds for data preprocessing. As introduced in Section IV-A, our implementation clips the data values within specified lower and upper bounds. This is important because the quality of representation is sensitive to these thresholds. Table V shows the PSNR reconstruction accuracy for different combinations of lower and upper bounds.

TABLE V: Impact of data threshold on reconstruction PSNR for the vortex dataset.

		upper bound					
		5.0	5.5	6.0	6.5	7.0	7.5
lower bound	0.0	32.717	32.284	32.581	32.815	32.934	33.039
	0.5	31.884	32.303	32.648	32.862	32.992	33.401
	1.0	31.884	32.303	32.648	32.862	32.992	33.401
	1.5	30.472	30.868	30.979	31.17	31.257	31.388
	2.0	29.947	30.144	30.263	30.428	30.435	30.538
	2.5	29.150	29.240	29.400	29.381	29.485	29.501
	3.0	28.475	28.607	28.679	28.739	28.882	28.783
	3.5	28.475	28.607	28.679	28.739	28.882	28.783
	4.0	27.071	27.288	27.422	27.584	27.667	27.760

The general trend is that the PSNR increases while the range of data (i.e., the difference between the lower and upper bounds) increases, which is reflected by the larger PSNR values close to the top-right of Table V. However, for a small lower bound, this trend reverses. Specifically, we can see that the lower bound 0.5 outperforms the lower bound 0 for nearly all values of the upper bounds. Thus, the maximum PSNR occurs when the lower bound is 0.5 and the upper bound is 7.5 for the vortex dataset, as shown in Table V. This is probably because the values near 0 cannot provide useful information about the dynamics of the system and should be considered as noises.

V. Conclusion

In this paper, we have shown a new method to estimate optical flows for the interpolation of time-varying scientific data. Our approach can predict the intermediate frames with higher accuracy and efficiency. We have compared our methods with several existing optical flow and data reduction methods and showed that our method can achieve superior performance in term of accuracy and compression ratio. The results also show the validity of our assumptions on the model of time-varying scientific data.

In the future, we would like to improve our model to tackle larger scale multivariate scientific datasets that have more complex dynamics. We would like to further investigate the impact of the parameters, such as temporal and spatial resolutions and data threshold, and address data noises to improve the reconstruction accuracy of particle flow. We also plan to extend our approach to interpolate data in the spatial domain to further increase the data reduction ratio, while maintaining data essentials in compressed representations.

VI. Acknowledgment

This research has been sponsored by the National Science Foundation through grants ICER-1541043 and IIS-1423487.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016.

[2] A. Chaudhuri, T.-Y. Lee, H.-W. Shen, and T. Peterka. Efficient range distribution query in large-scale scientific data. In *Large-Scale Data Analysis and Visualization (LDAV), 2013 IEEE Symposium on*, pages 125–126. IEEE, 2013.

[3] C.-M. Chen, A. Biswas, and H.-W. Shen. Uncertainty modeling and error reduction for pathline computation in time-varying flow fields. In *Visualization Symposium (PacificVis), 2015 IEEE Pacific*, pages 215–222. IEEE, 2015.

[4] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[5] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. Van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[6] D. Fortun, P. Bouthemy, and C. Kervrann. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 134:1–21, 2015.

[7] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[8] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2462–2470, 2017.

[9] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[12] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981.

[13] K.-L. Ma and H.-W. Shen. Compression and accelerated rendering of time-varying volume data. In *Proceedings of the 2000 International Computer Symposium-Workshop on Computer graphics and virtual reality*, pages 82–89, 2000.

[14] M. B. Rodríguez, E. Gobbetti, J. A. I. Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. K. Suter. A survey of compressed GPU-based direct volume rendering. In *Eurographics (STARs)*, pages 117–136, 2013.

[15] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

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

[17] K.-C. Wang, K. Lu, T.-H. Wei, N. Shareef, and H.-W. Shen. Statistical visualization and analysis of large data using a value-based spatial distribution. In *Pacific Visualization Symposium (PacificVis), 2017 IEEE*, pages 161–170. IEEE, 2017.

[18] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.