Computers, Environment and Urban Systems 77 (2019) 101352

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/ceus

Capturing the dance of the earth: PolarGlobe: Real-time scientific visualization of vector field data to support climate science

Sizhe Wang[a,b], Wenwen Li[a,*]

[a] School of Geographical Sciences and Urban Planning, Arizona State University, United States
[b] School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, United States

ARTICLE INFO

Keywords:
Streamline visualization
Virtual globe
Climate change
Big data
PolarGlobe

ABSTRACT

This paper introduces a streamline visualization technique that empowers PolarGlobe, an interactive, virtual globe-based, multi-dimensional scientific visualization tool to facilitate the observation and visual inspection of changes in the climate in real time. Specifically, this technique achieves effective visualization of vector-based earth science data through an automated data processing pipeline which integrates novel strategies including random seeding, finer-granularity parallelization and real-time rendering. The random seeding strategy allows for a vivid visual effect and an interactive framerate regardless of the spatial resolution in the raw dataset. The visualization algorithm is designed to be naturally parallelizable by partitioning the rendering tasks of unsteady vector field into multiple subtasks such that high-performance rendering can be realized. The platform is capable of taking either irregular or regular gridded data as input, and through the proposed data (re)projection pipeline, an automatic transformation of spatially enabled scientific data from the original data projection to the 3D globe-based virtual space is achieved. A series of experiments was conducted to identify the best configuration of rendering parameters to achieve the optimal rendering performance and visual effect. The results demonstrated the scalability and capability of the proposed PolarGlobe system to visualize big and unsteady vector flow data across different spatial and temporal scales. PolarGlobe implements former Vice President Al Gore's vision of a digital earth that enables scientists and citizens across the world to interactively study our planet. We expect the methods and techniques presented in this work to contribute significantly to both the scientific visualization and climate science communities.

1. Introduction

Scientific visualization focuses on novel ways of presenting data to allow salient features to be clearly displayed and hidden patterns to be easily inspected (Hansen & Johnson, 2011). An effective visualization strategy facilitates information comprehension at orders of magnitude faster than reading through the raw numbers and text in the data. According to the NIH (National Institute of Health)/NSF (National Science Foundation) Visualization Research Challenges Report (Johnson et al., 2005), visualization plays a critical role in scientific discovery, security, and competitiveness, and the insights it provides will help to "discover new theories, techniques, and methods, and improve the daily life of the general public." Today, visualization has been leveraged in many science domains to foster knowledge generation across traditional disciplinary boundaries (Keena, Etman, Draper, Pinheiro, & Dyson, 2016).

In climate science, scientists have relied intensively on visualization tools to understand extreme weather events (Wang, Li, Wang, & Johnson, 2018), analyze atmospheric processes (Helbig et al., 2014), communicate results with the general public (Dyer & Amburn, 2010; Johansson et al., 2017), and support informed decision making (Li, Shao, Wang, Zhou, & Wu, 2016). In the past decade, the entire science domain for that matter was ushered to the era of big data (Lynch, 2008). Advances in data acquisition techniques, such as environmental sensor networks, satellite images, and numerical simulation models, have resulted in the generation of massive amounts of climate data at an unprecedented speed, resolution, and complexity (Rautenhaus et al., 2018). For instance, the climate change data collected by the National Aeronautics and Space Administration (NASA) is expected to reach the size of 350 PB (Skytland, 2012), which is equivalent to 70 years of the total letters delivered by the United States Postal Service. The US National Centers for Environmental Prediction (NCEP)'s Global Forecast System (GFS) is generating weather forecasting data at a rate of over 240 GB per day (Han & Pan, 2011). The Geostationary Operational Environmental Satellite system captures the meteorology data of

* Corresponding author.
E-mail address: wenwen@asu.edu (W. Li).

https://doi.org/10.1016/j.compenvurbsys.2019.101352
Received 10 April 2019; Received in revised form 17 May 2019
Available online 19 June 2019

continental US in near- real time (every 5 min; GEOS-R, 2018). These voluminous, multi-dimensional, and multi-variate climate data pose significant challenges to traditional visualization tools in terms of dealing and effectively using these big data (Schnase et al., 2017).

Many key climate variables that capture motions in the atmosphere, such as wind direction and speed, can be categorized as vector field data. Enabling the effective visualization of such data plays a critical role in tracking and analyzing extreme weather conditions, such as tropical storms and typhoons (Seckel, 2018), as well as in identifying similar flow patterns at different geographical regions and across different spatial and temporal scales (Ingram & Chu, 1987). At the same time, scientific visualization serves as an important teaching tool to increase the next generation's interest in science and engineering (Dyer & Amburn, 2010). Compared with scalar data, such as temperature, vector field data are more challenging to display, as they contain not only magnitude values but also directions at a specific point in space. As some vector field data are unsteady or change with time, this poses another significant challenge for their visualization.

Several climate visualization platforms, such as Paraview (Ayachit, Geveci, Moreland, Patchett, & Ahrens, 2012), CDAT (Climate Data Analysis Tool; Santos et al., 2013), and the Integrated Data Viewer (IDV), have provided support to the visualization of vector field data. Specifically, Paraview is an open-source and multi-platform scientific data visualization software that allows users to quickly build up a visualization pipeline for analyzing large datasets. CDAT is a similar tool with a focus on visualizing climate data through its interactive command line interface. IDV is a software developed by Unidata and provides support to the visualization of geoscience data from multi-source, including remote sensing imagery, gridded data, surface observations, radar data, through a unified interface. These tools facilitate data visualization in a layer-based (2D) or volume-oriented (3D) setting. Some support the overlay of vector field data on top of a base map (satellite image of the area) or a scalar image (temperature); different ways of presenting vector field data (a more detailed methodological review) can be found in Section 2).

However, these climate visualization platforms suffer from different deficiencies. First, existing tools (i.e. Paraview) that offer advanced vector field visualization has limited capability to examine climate phenomena in a spatial context. Most tools only provide direct visualization of the data on a blank canvas. The spatial context—where on earth the phenomena being investigated are occurring are missing. A 2D map is sometimes is provided, but much distortion is generated when 3D phenomena are projected into a 2D context. Second, support for the visualization of unsteady vector field data, or data with real-time characteristics and that are continuously changing over time, is lacking. Third, an efficient algorithm is needed to realize data rendering. Many existing solutions rely completely on either hardware acceleration or software rendering. A combination and balance of these two strategies are important, especially when moving visualization from a desktop environment to a web-based environment. Fourth, tools such as CDAT lack a user-friendly graphic interface. Most visualization functions need to be invoked through a command line tool programmatically. The deep learning curve of such tools makes them difficult to use by non-experts. Fifth, most existing tools remain as standalone applications; there is limited support to virtual communities and physically distributed groups.

In recent years, open source libraries, such as Openlayers (2019) and Cesium (2019) have become popular in developing web-based visualization systems. They provide rich mapping features and interfaces, such as rendering of images, gridded and vector data on base maps or 3D globes. Both libraries are built upon open, easy-to-extend architectures. Several interesting web applications, such as Earth Wind Map (Beccario, 2019) and Windy.com are built upon these open-source libraries. Both of these applications support the visualization of 2D steady vector field data (such as wind) generated from numerical simulation models, but the lack of support to high dimensional or time-series data limits their utilities in analyzing complex climate phenomena.

To address the above challenges, this study will introduce our research in developing a vector field visualization technique based on streamline visualization. This method will take time-series data into account and provide an effective and efficient rendering strategy for multi-dimensional, real-time climate data. We also integrated it into PolarGlobe, a web-scale virtual globe platform that builds on and extends Cesium to allow multi-dimensional visualization and visual analytics of big climate data (Li & Wang, 2017; Wang, Li, & Wang, 2017). The remainder of the paper is organized as follows: Section 2 reviews recent literature, Section 3 introduces the proposed streamline visualization approach, and Section 4 compares and conducts a series of experiments to balance performance and visual effect. Section 5 presents the application of the proposed approach to enable real-time monitoring of hurricane Florence and other critical movements in the atmosphere and the ocean, and Section 6 concludes the work and presents future research directions.

2. Related work

Vector field data visualization methods can generally be categorized into the following four classes: direct visualization, dense texture-based visualization, geometric visualization, and feature-based visualization (Laramee et al., 2004; McLoughlin, Laramee, Peikert, Post, & Chen, 2010; Post, Vrolijk, Hauser, Laramee, & Doleisch, 2003). Below, we provide a review of these approaches. As feature-based visualization focuses more on feature extraction, subsetting, and flow tracing instead of increasing the interpretability of vector field data through visualization, therefore, this approach is excluded in the review.

2.1. Direct visualization

Direct flow visualization translates the vector field as directly as possible. It involves simple algorithms and little computation. This technique is frequently leveraged in 2D visualization; pictures are produced by placing an arrow glyph (see an example in Fig. 1) at each sample point to represent the direction and magnitude of a vector field at different locations, or, more simply, the colors of the vector field data are mapped according to the given magnitude. Because of the simplicity and computation efficiency of direct flow visualization techniques, they are frequently used in climate and meteorology data visualization. In

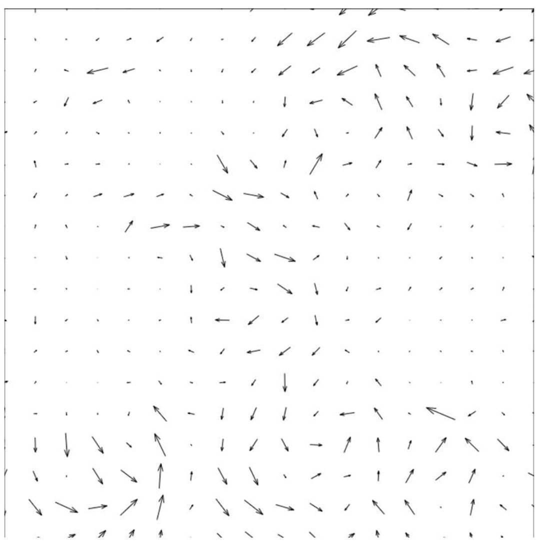


Fig. 1. An example of direct visualization using arrows glyph. The sample dataset used for the demonstration is from Polar-WRF (Weather Research Forecasting) model with time stamp 2012-01-01 00:00 GMT.

the work of Doraiswamy, Natarajan, and Nanjundiah (2013), an arrow glyph is used to visualize cloud movement and glyph colors in order to represent the time dimension and thus investigate cloud oscillation at different spatiotemporal scales. Despite its advantages, however, direct flow visualization suffers from visual complexity and a lack of visual coherency of the produced image (Edmunds et al., 2012; Edmunds et al., 2012; McLoughlin et al., 2010).

2.2. *Dense texture-based visualization*

Dense texture-based techniques represent the vector field by exploiting textures formed by a cluster of vectors. By covering the entire domain of datasets, this approach provides a dense visualization result with plenty of details, capturing many flow characteristics, including complicated patterns, such as vortices, sources, and sinks (McLoughlin et al., 2010). In general, this method is more suitable to 2D or surface visualization, and it suffers similar weaknesses in 3D representation as direct flow visualization does. The two most commonly investigated approaches in dense texture-based flow visualization are line integral convolution (LIC) and texture advection.

LIC was first introduced by Cabral and Leedom (1993). Several extensions, improvements, and variants have been proposed thereafter, such as parallel LIC (Zöckler, Stalling, & Hege, 1997), oriented LIC (Wegenkittl, Groller, & Purgathofer, 1997), volume LIC (Rezk-Salama, Hastreiter, Teitzel, & Ertl, 1999), and HyperLIC (Zheng & Pang, 2003). Given a vector field on a 2D Cartesian grid as the input, the original LIC method converts it to a white noise texture with the same input size to create a dense visualization of the flow field by integrating the path of streamlines and applying different filters. Fig. 2 illustrates examples of flow visualization using the original LIC (Figure2a) and the oriented LIC (Fig. 2b) techniques.

A representative method for texture advection flow visualization is image-based flow visualization (IBFV), which provides a faster representation for the dense, 2D, unsteady vector field (van Wijk, 2002). Applying the advection and decay operations on textures in the image space, this visualization method produces each frame by blending between the warped image to represent vector field directions, and several background images that consist of white noise textures. This method reduces computations by performing integration on advected small quadrilaterals instead of individual pixels, making the algorithm faster than many other dense texture-based flow visualization methods (Laramee et al., 2004). Other studies have attempted to identify specific hydrodynamic features and processes by implementing the IBFV algorithm to effectively visualize 2D time-dependent vector fields (Warne, Larsen, Young, & Cox, 2013). Warne argued that dense texture-based flow visualization performs better than direct and geometric-based methods in presenting the complex flow regimes of a shallow tidal barrier estuary.

While a number of dense texture-based solutions have been developed for 2D unsteady flow and surface visualization (van Wijk, 2002; Wijk 2003), their applications in 3D flow fields remain limited, especially in the case of visualizing unsteady flow, which involves high memory demands and additional memory I/O bandwidth for the efficient processing of vector data (Falk & Weiskopf, 2008). This problem becomes more significant when dealing with real-time vector field data. Moreover, perceptual issues in texture-based approaches are considered to be even more difficult than hardware limitations. Directly observing 3D data in a 3D space is a challenging task for the human eye and brain (Laramee et al., 2004). In comparison, geometric-based flow visualization based on particle animation is considered a better approach in presenting the 3D vector field.

2.3. *Geometric visualization*

In research on geometric approaches for flow visualization, the focus has mainly been on streamline visualization because of its effectiveness and the quality of its produced results, as well as the ease in its implementation compared with those of other geometric methods (McLoughlin et al., 2010). In streamline visualization, a certain number of seeds are placed on a canvas, and a curve is drawn from each seeding location; every point along the curve is tangent to the instantaneous local velocity vector. The resulting image will illustrate that the path of the velocity and the visual effect are significantly affected by the placement of streamlines. For instance, an unevenly distributed streamline that results from an arbitrary seeding strategy may fail to precisely represent some critical patterns in the vector field. Some previous studies (Bürger, Schneider, Kondratieva, Krüger, & Westermann, 2007; Edmunds, Laramee, Chen, et al., 2012; Edmunds, Laramee, Malki, et al., 2012; Spencer, Laramee, Chen, & Zhang, 2009; Ye, Kao, & Pang, 2005) have investigated an interactive or automatic seeding strategy that helps optimize streamline distribution and produce insightful visualizations.

Other studies have examined the application of streamline visualization in depicting climate phenomena. For example, a recent work (Shen, Nelson, Tao, & Lin, 2013) demonstrated how streamline visualization can help improve the understanding of tropical cyclone (TC) formation and intensification by enabling the interactive exploration of TC with environmental flows. This visualization generates streamlines at different heights to illustrate their structure and cross-scale interaction with surrounding flows. However, this visualization remains 2D. Fig. 3 shows an example of streamline visualization.

Extending the streamline visualization from 2D to 3D clearly

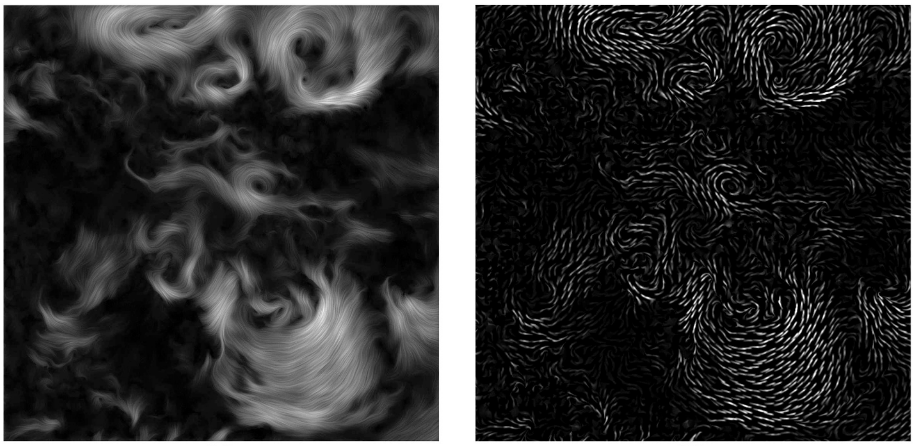


Fig. 2. Illustration of vector field visualization with LIC (a) and oriented LIC (b) techniques. The same sample dataset used in generating Fig. 1 is used here.

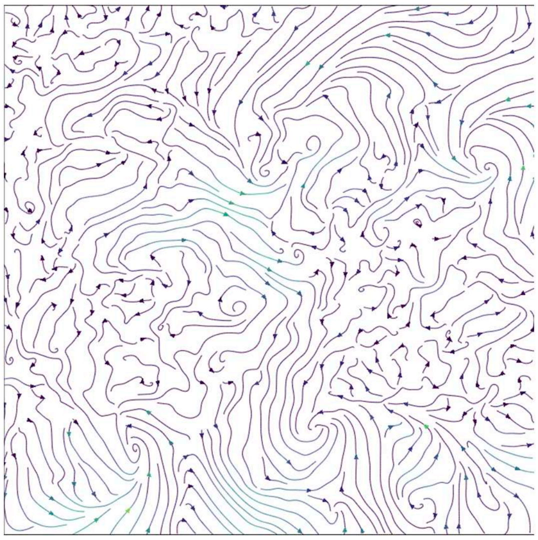


Fig. 3. An example of streamline visualization. The same sample dataset used in generating Fig. 1 is used here.

presents an advantage in providing more perceptual information and a more comprehensive view of 3D objects or phenomena. Lighting and shading provide better clues in object depth and structure in 3D, compared with 2D line primitives. The construction and rendering of unsteady, time-varying flows in 3D are clearly significant problems that involve challenges in rendering performance, visual errors and uncertainty, visual complexity, and perception issues, as well as interaction and integration issues with large data sets (Edmunds et al., 201; McLoughlin et al., 2010).

In this study, we propose a general geometric flow visualization method for vector field data visualization. This method can be applied in either a 2D or 3D vector space, and it is applicable to both steady and unsteady flow data. More design features and the rationale of this method can be found in Section 3.

3. Methodology

3.1. Vector field data

Scientific data, which can be abstracted as a vector field, are the focus of the visualization task. These data, which are time varied, multi-dimensional, and multi-variate, are the products of large-scale numerical simulation models, so they are a typical set of big data. For instance, one of the datasets we integrated into our visualization platform is GFS real-time and 10-day forecast data (Han & Pan, 2011). GFS is run by NCEP. The simulation output for the forecast of the global weather was generated every three hours and openly shared with the public. These datasets include 9 variables in 21 pressure levels, and the total data volume generated per day are about 240 GB, or a total of nearly 90 TB per year in general regularly-distributed information in binary format. With a typical vector field data, such as wind, on each pressure level covering a certain geographical extent, four variables < x,y,z,s > can be used to represent a wind vector. x, y, and z in this quadruple refer to the wind vector components in the horizontal (x,y) and vertical (z) dimensions, whereas s represents the speed component. Many other vector field data, such as ocean current, may be represented using the same 3D data model. The above data are represented in a regular data grid, namely, the same number of points is generated to represent data at different latitudes; see an example in Fig. 4a. Although our demonstration primarily uses this dataset, the proposed algorithm can also take irregular grid points (Fig. 4b) as input. An additional interpolation step will be required to convert the data into regular gridded data.

3.2. Geometric flow visualization for an unsteady data field

The time-varying and 3D characteristic of vector field climate data, such as atmospheric wind, makes it an unsteady data field. As discussed above, geometric visualization techniques are powerful in 3D unsteady data field visualization, as they provide an optimal effect for human perception and enable high visualization efficiency. We therefore use geometric visualization and line geometries to depict (Ayachit et al., 2012) the flow speed and direction at local scales, as well as (Beccario, 2019) flow features (such as cyclone for wind data), their corresponding movement, and shape transformation at the regional scales.

3.2.1. Method overview

The proposed flow/streamline visualization involves the following steps. First, a number of particles (seeds) are randomly but evenly placed within the extent of the vector field data. In Fig. 5(a), grey points show the (2D) regular grid on which the vector field data are available. Black points are randomly placed particles that serve as seeds of the streamlines. Instead of being kept static, the particles will move according to the direction and magnitude defined in the vector field. A reseeding process is also automatically applied once an existing seed moves out of the current viewing space. This process ensures that the particles can be redistributed at other positions rather than remain static. This also mitigates issues on uneven particle distribution in commonly used seeding strategies.

Second, to improve the rendering efficiency and visual effect, the streamline will not be drawn all at once or within a single iteration. Rather, we introduce a divide and conquer strategy to segment this rendering task into sequential subtasks, with each task being performed in one iteration and responsible for generating primitives that are a part of the line. As one subtask finishes drawing, a partial streamline is created, and the particle moves to a new location (Fig. 5b). The next subtask in order will continue drawing the streamline from this new particle location (Fig. 5c).

Third, a fading effect is applied to emphasize the newly drawn segments of the streamline and gradually fade out the part of the streamline that was drawn several iterations ago (Fig. 5d). To ensure a good visual effect, once the streamlines fill up a substantial portion of the canvas, they will be erased from the canvas. New seeds will then be placed, and the same rendering flow will be applied for iteratively rendering the streamlines.

The proposed divide and conquer strategy enables an animation effect even for static vector field data such that the shape and moving pattern of a phenomenon can be better captured. Furthermore, by breaking down the streamline visualization task into sequential subtasks (iterations), the amount of computation in each iteration is significantly reduced. This is a strategy designed specifically to address the challenges of big data visualization in a web-based environment. In this environment, visualization performance is always restrained by client-side resources, resulting in difficulties in achieving on-the-fly rendering.

At the level of implementation, we also combine this divide and conquer rendering strategy with hardware acceleration. The entire canvas on which the streamlines will be drawn is partitioned into several sub-regions. The streamline rendering tasks on each of these sub-regions will be handled concurrently by the parallel computing units on the client graphics processing units (GPUs). This way, real-time rendering can be better achieved.

3.2.2. Formulating the line generation process

The line generation process involves drawing the streamlines iteratively according to the vector field data. We first define an unsteady vector field as $\boldsymbol{v}(\boldsymbol{p}, \boldsymbol{t})$. The speed or magnitude of vector $\boldsymbol{v}$ changes as location $\boldsymbol{p}$ and time $\boldsymbol{t}$ change in continuous space. In this definition, $\boldsymbol{p}$ can move not only within 2D space (that is $\boldsymbol{p} \in R^2$) but also within 3D space ($\boldsymbol{p} \in R^3$).

First, k particles (seeds) will be randomly placed within the space to

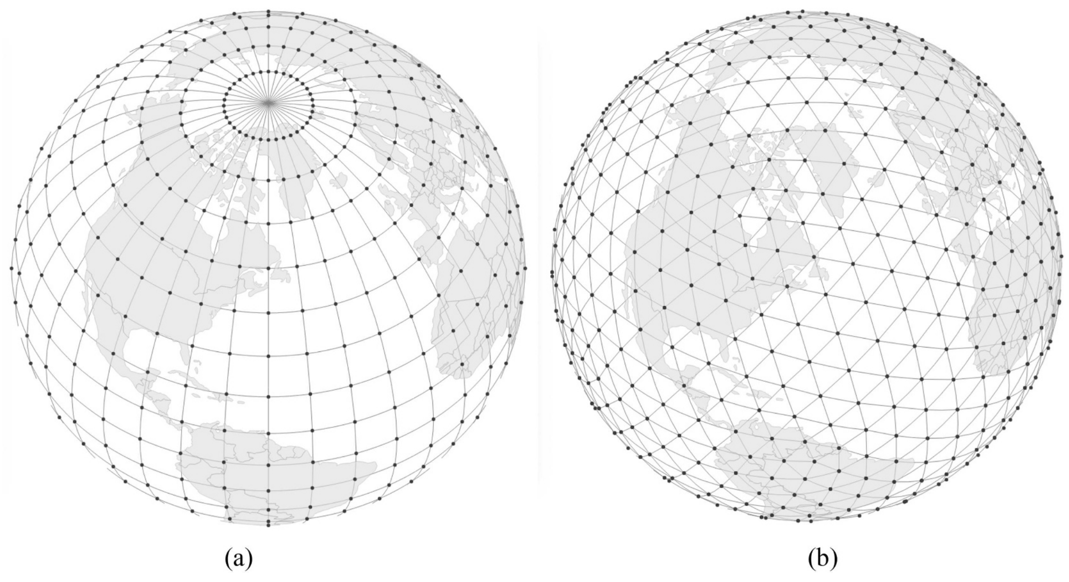


Fig. 4. Example of regular (a) and irregular gridded (b) data points from the climate simulation models.

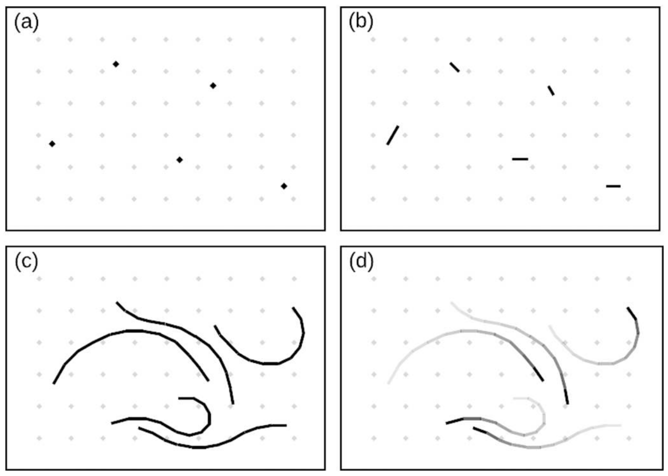


Fig. 5. – Illustration of the generation of linear geometries (i.e. wind direction) to enable streamline visualization. a) Initialize particles for streamline visualization. b) Results after the first iteration. c) After several iterations, the streamline is formed. d) Apply fading effect to the streamline.

serve as the starting location of the streamlines. $\boldsymbol{p}_i(t)$ represents the location of particle $\boldsymbol{i}$ ($i \in \{1, 2, \ldots, k\}$) at time t. Its initial position is $\boldsymbol{p}_i(0)$. A particle moves in 3D space iteration by iteration, within which a small segment of a streamline is drawn, and the segment is tangent to the local vector field. Eq. (1) defines how a particle $\boldsymbol{i}$ moves in each iteration:

$$\boldsymbol{p}_i(t + \Delta t) = \boldsymbol{p}_i(t) + \boldsymbol{v}(\boldsymbol{p}_i(t), t) \cdot \Delta t \tag{1}$$

The direction of movement is determined by vector field $\boldsymbol{v}$ at start position $\boldsymbol{p}_i(t)$ at time t. The distance of travel is jointly determined by the velocity of $\boldsymbol{v}$ and time interval Δt. Given known vector field data, Δt should be carefully selected to ensure a good visual effect. If Δt is too large, the particle travels far, and the velocity and direction at the new position may be significantly different from its previous movement. The resultant streamline may have many abrupt turns instead of running smoothly. A general guideline is that the traveled distance per unit time, $\boldsymbol{v}(\boldsymbol{p}_i(t), t) \cdot \Delta t$, should be no larger than the spatial resolution of the vector field grid. More experiments are conducted in Section 4 to illustrate how different choices of Δt make an impact on the visual effect.

The movement of the particle generates a line segment, a small portion of a streamline during the iteration at time t. The mathematical expression of the line segment $L_i(t)$ is given by Eq. (2):

$$L_i(t) = \{\boldsymbol{p}_i(t) + \boldsymbol{v}(\boldsymbol{p}_i(t), t) \cdot \tau \mid \tau \in [0, \Delta t]\} \tag{2}$$

According to Eq. (1), this line segment spans from $\boldsymbol{p}_i(t)$ to $\boldsymbol{p}_i(t + \Delta t)$.

The movement of all k particles will follow the above defined pattern. Correspondingly, at each iteration, k line segments will be generated simultaneously. Besides illustrating the movement, each line segment can also be displayed in different colors to indicate the magnitude of the flow, such as velocity. We then further define a value (σ) assignment to a line segment at any arbitrary position $\boldsymbol{p}$ and time t with Eq. (3):

$$\sigma = \begin{cases} \|\boldsymbol{v}(\boldsymbol{p}, t)\|, & if\ p \in L_i(t),\ \forall\ i \in \{1, 2, ..k\} \\ 0, & otherwise \end{cases} \tag{3}$$

During the iteration at time t, the value at position $\boldsymbol{p}$ is assigned as the magnitude of the vector field at the same position, if $\boldsymbol{p}$ is on any line segments $L_i(t)$ drawn at this iteration. Otherwise, the value is set to 0. Thereafter, a color scheme $c(x)$ is applied to the drawing space, converting numerical values to red green blue alpha (RGBA) colors. Alpha is a parameter that shows the transparency at a certain location. It ranges from 0 (fully transparent) to 1 (fully opaque). Thus, at each position $\boldsymbol{p}$, we have a color $c(\sigma(\boldsymbol{p}, t))$, an RGBA quadruplet, assigned to it.

When the process is performed iteratively, two streamlines may inevitably pass the same position at the same or different time. In order to better determine the color pattern at this position, a blending function $b()$ is introduced to combine the incoming color C at iteration t and the accumulated color C_{acc} from $t = 0$ to t- Δt. Eq. (4) defines how the color pattern is determined at location $\boldsymbol{p}$ and time :

$$C_{acc}(\boldsymbol{p}, t) = \begin{cases} c(\sigma(\boldsymbol{p}, 0)), & t = 0 \\ b(c(\sigma(\boldsymbol{p}, t)), \gamma \cdot C_{acc}(\boldsymbol{p}, t - \Delta t)), & t > 0 \end{cases}. \tag{4}$$

As seen in Eq. (4), a new parameter γ is introduced in this blending function. γ is a fading factor that determines how much the color accumulates before the current time point $C_{acc}(\boldsymbol{p}, t - \Delta t)$ contributes to the blended result. It has a value range of [0,1]. When $\gamma = 1$, there is no fading effect applied. When $\gamma = 0$, the previous color will be completely erased. With this fading factor applied, the newly drawn colors will be brighter, and the older ones will be dimmer. The blended color will support a better interpretation of the results.

Given two colors C_1 and C_2, the blend function emulates the overlay of a translucent color C_1 onto C_2. Eq. (5) provides its mathematical expression. This blending function is also known as *alpha blending*, which emulates the overlay of a translucent color on another one. This blending function is shown in Eq. (5) as follows:

$$b(C_1, C_2) = C'_{RGBA} = \begin{cases} C'_{RGB}\,(C_1, C_2) = C_{1A}C_{1RGB} + (1 - C_{1A})C_{2A}C_{2RGB} \\ C'_{A}\,(C_1, C_2) = C_{1A} + (1 - C_{1A})C_{2A} \end{cases} \tag{5}$$

The subscript *RGBA* represents a different color component.

3.3. Visualization on a virtual globe

Realizing real-time data rendering, especially in a cyber-environment, is challenging. Highly efficient data transfer, memory management, and vivid visualization are required. To address these big data challenges, we leverage the data compression method introduced by Li and Wang (2017) to mitigate the impact on performance while visualizing voluminous data. Compared with the application of applied streaming compression to scalar data in Li and Wang (2017), compressing vector field data follows a similar workflow: (Ayachit et al., 2012) transforming multi-dimensional grids into a large 2D image through vertical layer-based reorganization and colorization encoding of each data point, (Beccario, 2019) applying step 1) to the data at each timeframe and generating a series of 2D images, each of which presents data at a single timeframe, and (Bürger et al., 2007) leveraging video codecs to compress the image series into a video. Note that vector field data are normally described by multiple variables. Each variable is a component of the vector. Therefore, in step (Ayachit et al., 2012), each of these data components will be encoded into a 2D image according to some predefined rule and will be recovered according to the rule once the data arrive at the browser side. The advantage of using video-based encoding instead of image- or other text-based encoding is that the data stream can be parsed and rendered at the client side as it is being transmitted. There is no need to wait for all data to arrive before beginning the decompression and the rendering process. Therefore, high rendering performance and an interactive framerate can be achieved.

An additional challenge is in the rendering of 3D spatially enabled vector field data into a 3D geographical space, which is the virtual globe in our case. A key step is to ensure the accurate placement of corresponding data values from the original data space to their actual spatial location above or below the spherical earth surface. Here, we proposed a spatial data (re)projection pipeline (Fig. 6) to achieve the 3D visualization of multi-dimensional spatial data. First, to best represent the geo-location on a spherical earth surface, the gridded data could either be represented by latitude and longitude in a world geographical system (WGS), for instance, WGS 84, or a projected coordinate system, such as Web Mercator, which uses meters to represent the offset of a point from the origin in its x, y, and z coordinates. Once gridded data are parsed, a value in a data point can be referred by its row and column indices. Hence, the first step involves the conversion of the grid cell index into its actual x, y coordinates according to the corresponding data projection. Second, the x and y coordinates, as well as the height information (z dimension) in the data, need to be further projected into the 3D Cartesian coordinate system (CCS) because this system is what is commonly used by virtual globes as their default coordinate system. In a CCS, the origin is located in the center of the earth, with the z axis often directed toward the North Pole. Finally, the 3D coordinates in the 3D CCS will be converted to 2D screen coordinates to display the final visual effect.

4. Performance experiments and results

This section examines the performance of the proposed 3D flow visualization method by conducting several quantitative analyses and comparisons of the visual effects. Considering that we developed this method to realize a web-based vector field visualization, the client machine we used in this experiment only has an average hardware configuration to simulate a client with average performance. It has a 16 GB memory, a quad-core processor at 3.4 GHz, and an AMD Radeon HD 6970 M video card with 2 GB graphics memory. The runtime environment uses the latest Google Chrome (v65.0.3325.181) with MacOS 10.11.6. All the images included in this section are rendered with a perspective projection. The angle of the field of view of the viewing frustum is 60 degrees. The viewing direction is set to be perpendicular to the virtual globe surface. The data in use are GFS real-time forecast, as described in Section 3.1.

4.1. Impact of randomly generated particles on the rendering performance

As discussed in Section 3.2.1, the flowlines are drawn from a number of seed particles evenly placed on the virtual globe. For each seed, the following attributes need to be derived through interactive

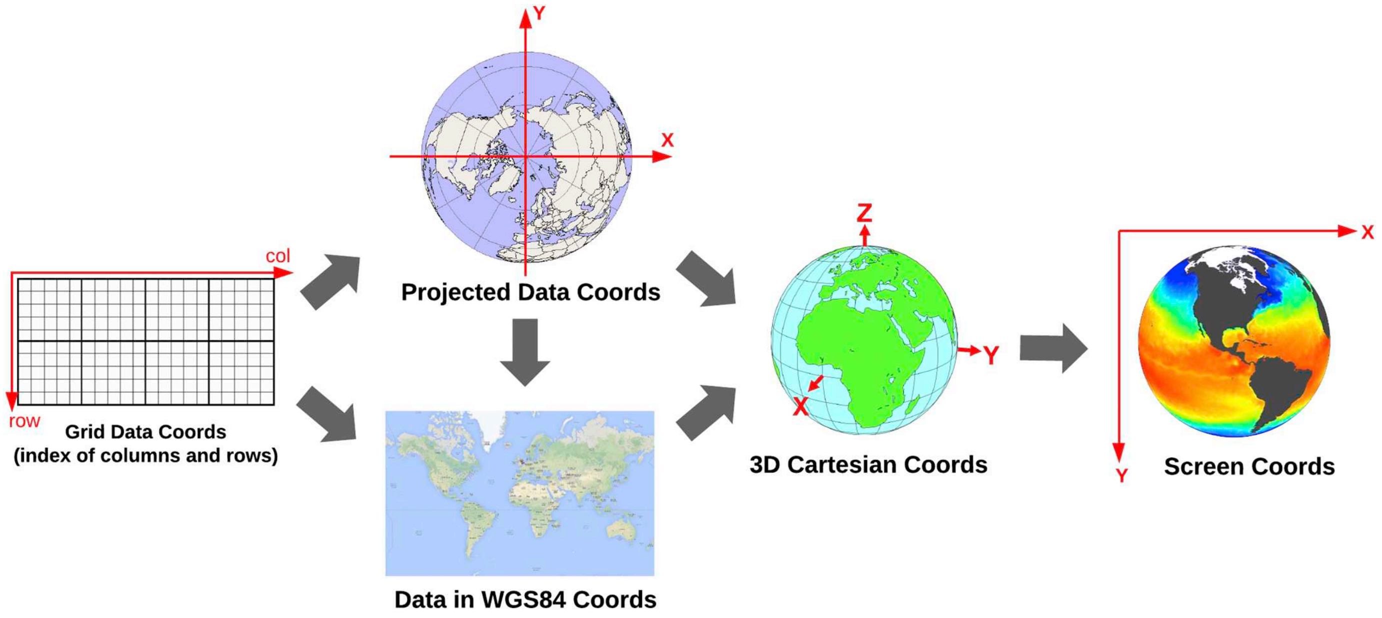


Fig. 6. A data reprojection pipeline to enable the virtual globe visualization of multi-dimensional spatially enabled scientific data.

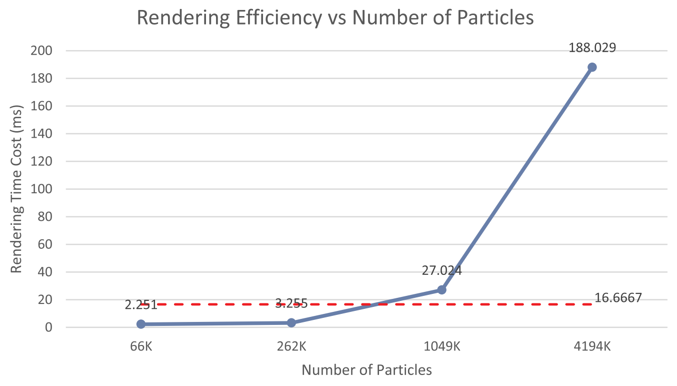


Fig. 7. Rendering efficiency with different numbers of particles.

calculation in the visualization process: (Ayachit et al., 2012) the right drawing location on the virtual globe and (Beccario, 2019) the color and angle of the flowline to reflect the magnitude and directions of the vector field. Therefore, the more particles to generate, the more computation will be involved. To identify the proper number of initially generated particles given the constraints of computing power at the client GPU, we conducted the following experiment to compare the time cost per frame for flowline rendering with the changing particle number. The results are shown in Fig. 7.

In Fig. 7, the x axis represents the number of generated particles, and the y axis indicates the average time cost for rendering each frame. Ideally, this time should be < 1/60 s (approximately 16.67 ms) to match the refresh rate (60 Hz) that most commercial monitors support. The results show a dramatic increase in rendering time as the number of particles increases. Limiting the number of particles below 0.7 million is better to provide most users a smooth animation. In PolarGlobe, we use this as the particle setting, and it is also used in the subsequent experiments.

4.2. Impact of the fading factor on the visual effect of the vector field

In this section, we further investigate the impact of the fading factor on the visual effect. As discussed in Section 3, a fading effect is introduced when blending the two scenes of vector field data, which represent a part of a vector or its change over time (Eq. (4)). The fading factor (γ) controls the effect of fading, and it also aims to reduce visual complexity and ease the perception of the direction of the vectors. A fading factor with an excessively large value may result in dense and chaotic flow visualization. By contrast, when the value of the fading factor is small, the length of flow lines will be short, making it difficult to observe traces of particles and the direction of vectors. We evaluated the impact on such a visual effect by applying different fading factors. The comparison investigated a cyclone visualization in Northern Europe. The viewing distance (altitude of the view point) is 4×10^6 meters. The comparison results from left to right are shown in Fig. 8. The fading factors from left to right are set to 0.52, 0.846, and 0.952. It is clear that when γ is small (0.52), one can hardly observe the shape of the cyclone, whereas when γ is high (0.952), the cyclone trace becomes very dense, and traces at different pressure levels intersect. When γ is set to a proper value (0.846) in this case, the shape and directions are clear and easy to inspect.

However, a static fading factor cannot always guarantee the best visual effect. The density of the drawn flow lines varies when the zoom level changes. In a virtual globe environment, zooming is achieved by modifying the viewing distance. In order to optimize the visual effect of flowlines and the animation, deriving a mapping between the viewing distance and the fading factor is important to obtain a dynamic balance between flowline density and the viewing distance. We formulate such a mapping with the following equation:

$$\gamma = max(0, 1 - k \cdot d) \quad (5)$$

where d is the viewing distance and k is a constant which adjusts the value of fading factor as the viewing distance changes. An empirical value $k = 1.6 \times 10^{-8}$ is used in our work based on the rational that the fading factor γ should stay at around 0.8 to avoid the streamlines to be too dense when the full globe view is presented (at $d = 12{,}000km$). The max function ensures that the produced fading factor will be in the desired value range. A comparative experiment is performed to demonstrate the visual effect at different viewing distances and examine the feasibility of the above equation. The results are shown in Fig. 9.

In Fig. 9, the left column presents the rendered images with a static fading factor. With this configuration, the visual effect is acceptable when the viewing distance is at around 5000 km. However, the sparse geometries drawn at a smaller viewing distance (d = 1000 km) made it more challenging to interpret the direction of the vector fields. With a far viewing distance (d = 10,000 km), geometries become too dense and disturbing to observe the patterns and changes. The direction of geometries can sometimes be hardly interpreted, especially when some nearby geometries flow into different directions. Additionally, the base map becomes difficult to recognize when the geometries are too dense. The visual effect during an animation will be even worse when the data are visualized on a large scale and on full screens. The right column presents the results that apply the dynamic fading factor derived from Equation (Doraiswamy et al., 2013). At all zooming scales/viewing distances, the density of the rendered geometries is well balanced, resulting in a cleanly rendered and easily interpreted visualization.

4.3. Impact of scene switching frequency on the visual effect of the vector field

Besides the fading factor, which decides how information in the old scene and that in the new scene are blended together, another factor that influences the visual effect of the flowline is the frequency in loading a new scene. This is the same as Δt in Eq. (1), which indicates

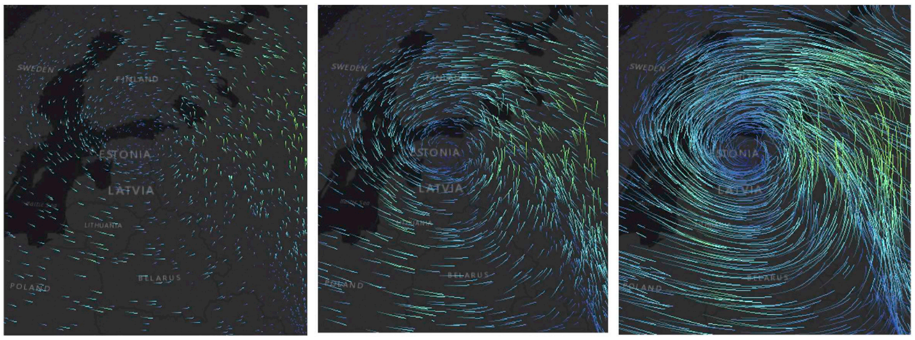


Fig. 8. Visual effect with different fading factors.

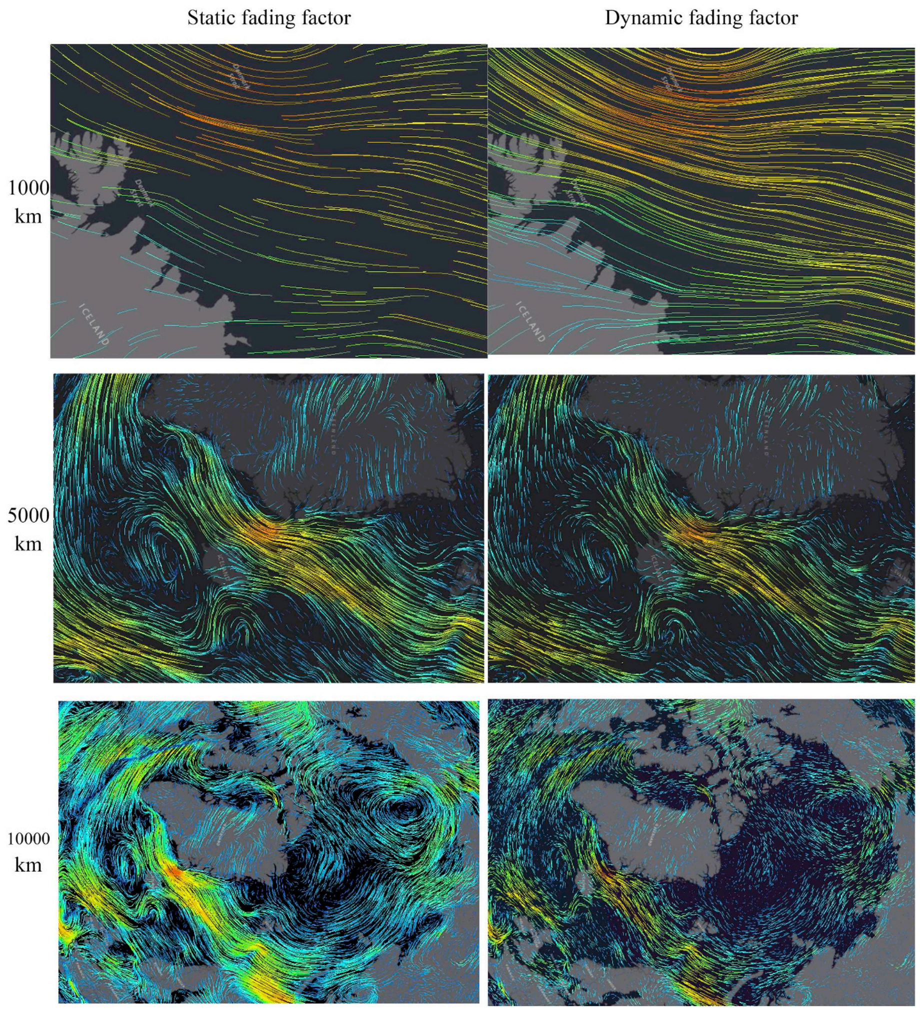


Fig. 9. Visual effect at different viewing distances.

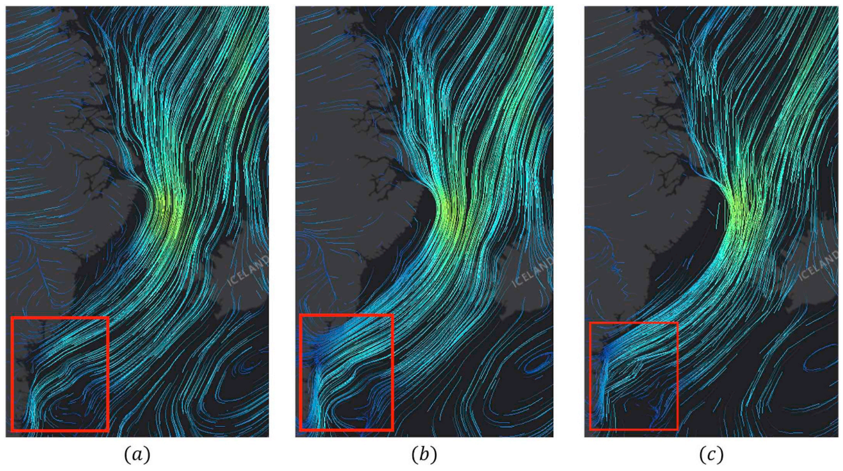


Fig. 10. Flowline deformation with an increased Δt. In case (b), the interval is set to be four times of that in case (a), and Δt in case (c) is sixteen times as that in case (a). The viewing distance is 3000 km in the scenes.

the time span that a flowline is going to move according to the speed and direction at the original location. A scenario with too frequent scene switching, or a small internal Δt, will introduce extra computation, that is, interpolation from the gridded data to estimate the data value at any arbitrary spatial location that a flowline moves to. By contrast, a low frequency (large Δt) may possibly generate a flowline with abrupt changes because if a particle moves too far, the data values at the new location may be more significantly different from those at the particle's original position. Fig. 10 illustrates this effect. When a proper Δt is set, the flowline goes rather smoothly both globally and locally (red box in Fig. 10a). However, as Δt increases, the generated flowlines show more deformation and become less realistic (Fig. 10b

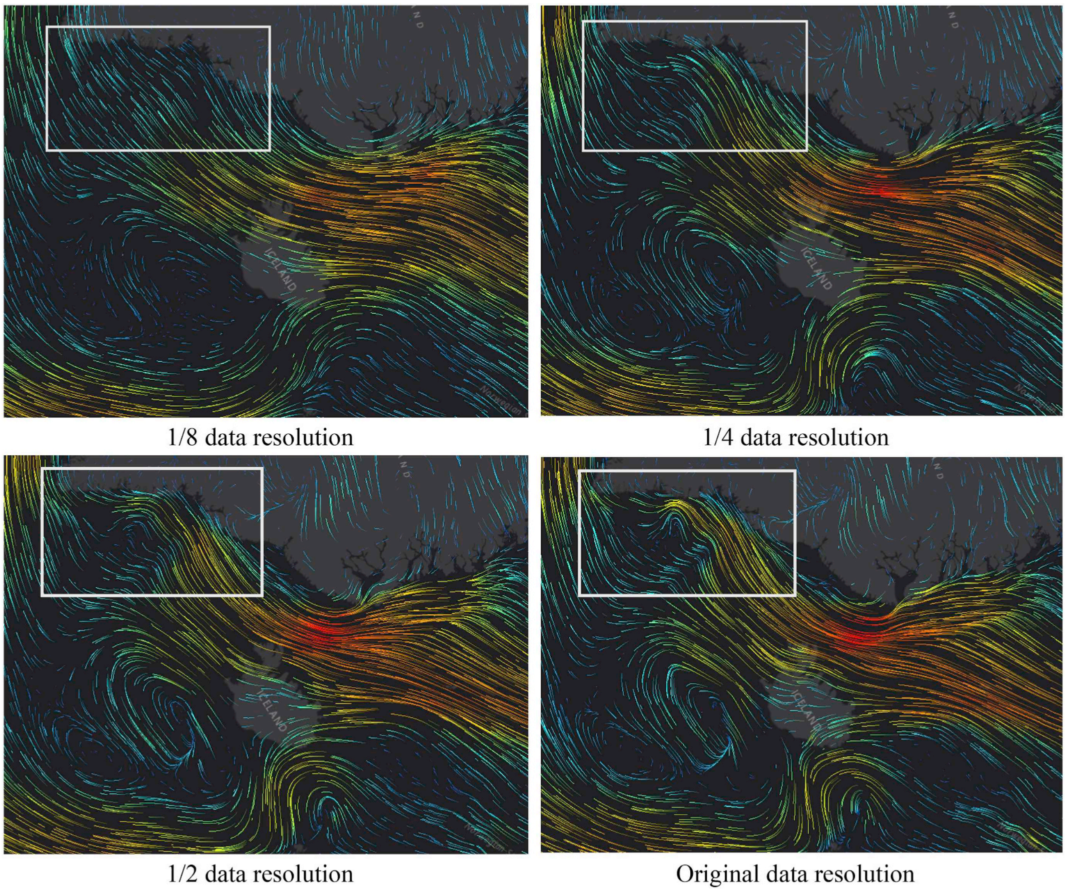


Fig. 11. Impact of data resolution on the visual effect.

and c). In our experiment, the gold standard is to set the Δt such that the moving distance of a particle within one scene is less than the spatial resolution of the data itself.

4.4. *Impact of data resolution on the visual effect and efficiency*

A notable advantage of this proposed method for vector field visualization is the lack of an impact of data volume or data resolution on the rendering efficiency, as the generation of the flowline is controlled by the number of particles placed on the virtual globe instead of the number of points in the original data. Therefore, even when the data resolution is low, our method remains capable of generating a satisfactory visual effect. When the data resolution becomes high, more precise visualization occurs without sacrificing the rendering performance. We conducted an experiment to examine this from the perspectives of both visual effect and rendering efficiency, as shown in Figs. 11 and 12, respectively.

We down-sampled the original data from the GFS model to create the half-resolution, quarter-resolution, and one-eighth-resolution vector fields. The data size decreases accordingly. Visualization with the same parameters but different data resolutions is enabled by setting the optimal number of total particles derived from experiment IV.A. The visual effects are presented in Fig. 11. The visual effects at four data resolutions are all vivid. But as the data resolution increases, the rendered image captures and conveys more details, which present local changes in the flowlines (see the areas in white boxes in Fig. 11). The effect of the coastline on the data flow is also better captured in the visualization using higher-resolution data. The rendering efficiency with changing data resolutions is shown as Fig. 12. Neither an obvious positive nor negative correlation between these two factors can be observed. The rendering efficiency remains quite consistent across data at different resolutions. Therefore, we have confirmed that data size and data resolution can hardly affect the rendering efficiency of the proposed method.

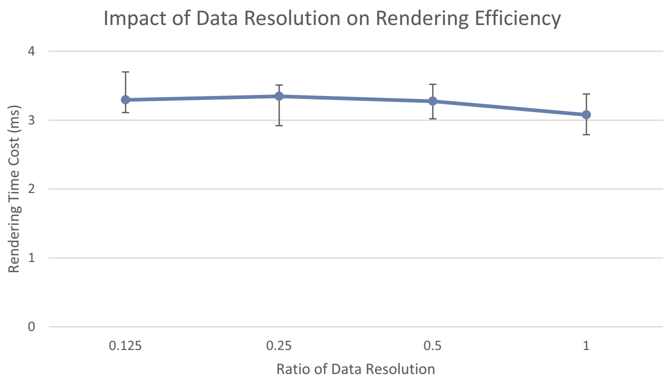
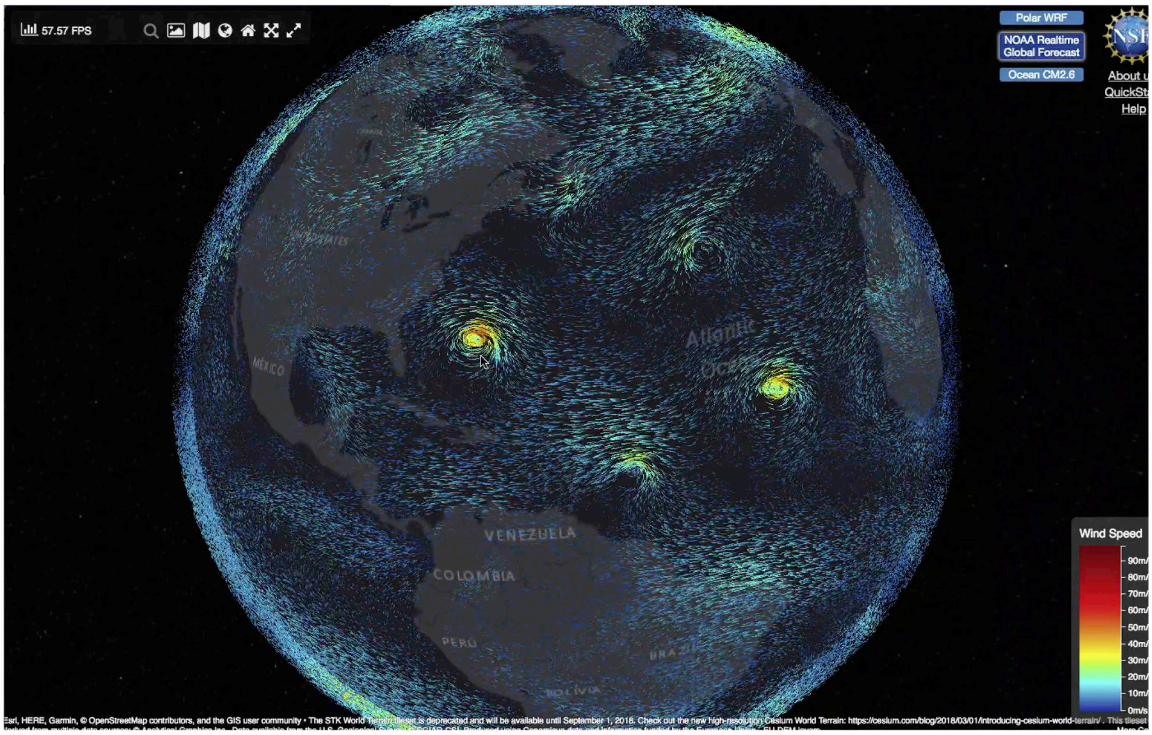


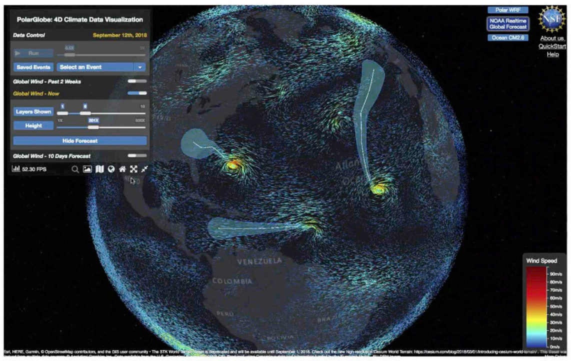
Fig. 12. Impact of data resolution on rendering efficiency. The x axis lists four different data resolutions and y axis indicates the average time cost for rendering each data frame.

4.5. *Leveraging PolarGlobe to support climate studies*

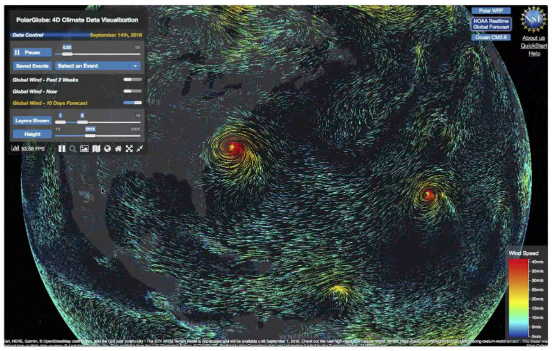
Utilizing the proposed streamline visualization technique and leveraging the real-time climate forecast data from GFS, PolarGlobe is capable of vividly demonstrating changes in the atmosphere and in real time. Fig. 13(a) illustrates the real-time visualization (on September 12th, 2018) of global wind in the 2018 North Atlantic hurricane season. Three active tropical storms that were formed can be clearly observed: category 4 Hurricane Florence, Hurricane Isaac, and Hurricane Helene.



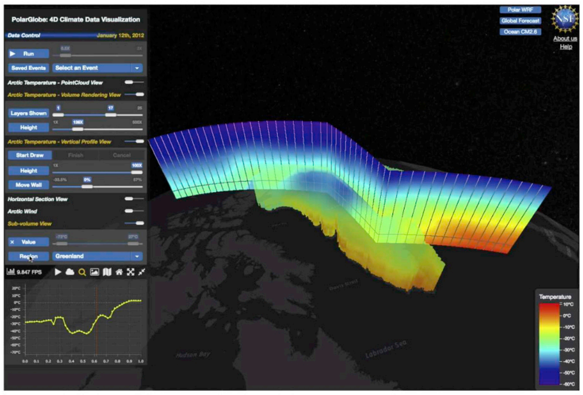
(a) Real-time visualization (on September 12th, 2018) of global wind in the 2018 North Atlantic hurricane season.


PolarGlobe: 4D Climate Data Visualization
Data Control
September 12th, 2018
Global Wind - Past 2 Weeks
Global Wind - Now
Hide Forecast
Global Wind - 10 Days Forecast
Wind Speed

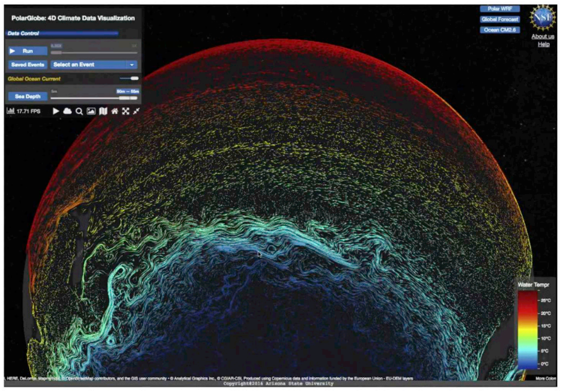
(b) Display of projected trajectories of three observed hurricanes: Hurricane Florence, Hurricane Isaac, and Hurricane Helene.


PolarGlobe: 4D Climate Data Visualization
Data Control
September 14th, 2018
Pause
Global Wind - Past 2 Weeks
Global Wind - Now
Global Wind - 10 Days Forecast
Wind Speed

(c) Real-time visualization on September 14, 2018 when Hurricane Florence made landfall.


PolarGlobe: 4D Climate Data Visualization
Data Control
January 12th, 2012
Arctic Temperature - PointCloud View
Arctic Temperature - Volume Rendering View
Arctic Temperature - Vertical Profile View
Horizontal Section View
Arctic Wind
Sub-volume View
Greenland
Temperature

(d) Spatiotemporal changes of atmospheric temperature in the Greenland region.



(e) A multi-variate view of undersea water temperature (color) and the movement of ocean current (direction) near the Antarctic ocean.

Fig. 13. Leveraging PolarGlobe to observe changes in climate and on earth and in real-time.

In this same week, storm activity reached a historic high in the Atlantic hurricane season. Projected trajectories can also be viewed and turned on and off to aid comprehension (Fig. 13b). On September 14th, as the hurricanes progressed and Florence finally made landfall, one can observe a very clear and distinct eye wall with a much stronger accumulated wind speed (Fig. 13c). All these screenshots are captured in real time. PolarGlobe also has the ability to cache real-time data in the past two weeks and 10-day forecast data from the GFS model.

Besides integrating the GFS model, PolarGlobe has also incorporated data from other numerical simulation models, such as the Polar version of the Weather Research Forecast Model—PolarWRF—and the ocean-atmosphere circulation model CM 2.5. These models include a variety of variables covering both land and ocean on earth. Fig. 13(d) shows the monitoring of spatiotemporal changes in temperature in the Greenland region leveraged by high-performance volume rendering, spatial filtering, and value picking (Li & Wang, 2017). Fig. 13(e) shows a multi-variate view of undersea water temperature (color) and the movement of ocean current (direction) near the Antarctic ocean leveraging the proposed streamline visualization techniques over big data. These enabling techniques can significantly help answer questions on global climate change and solve pressing issues in the Polar regions.

5. Conclusions

This study introduces PolarGlobe, a large-scale, multi-dimensional, virtual globe-based scientific visualization tool that allows online access to big numerical simulation data on the climate and ocean in real time. Specifically, we introduced a streamline visualization technique that incorporates the following methodological advances: (Ayachit et al., 2012) the algorithm is naturally parallelizable by partitioning unsteady vector field rendering tasks into finer granularity in the algorithm design; (Beccario, 2019) the seed placement strategy ensures high-performance rendering and an interactive framerate regardless of the original data resolution; (Bürger et al., 2007) a data (re)projection pipeline allows the automatic transformation of raw scientific data in any data projection to a 3D virtual globe projection and, eventually, to the screen coordinate system for multi-dimensional visualization; (Cabral & Leedom, 1993) a careful calibration and dynamic adjustment of rendering parameters ensure an optimal visual effect across different spatial and temporal scales; and (Cesium, 2019) the integration of the proposed technique into the PolarGlobe visualization platform enabled by a highly efficient big data transmission technique over the Internet allows anyone from the globe to access the portal and perform interested analysis.

This work has produced a novel visualization solution that enable interactive, dynamic, real-time visualization of multi-dimensional, time-series climate data on an online virtual globe. Different from commonly used climate visualization, which is always single-layer based and/or uses 2D presentation, the proposed tool allows situating climate phenomena in a near-real geographical context. With PolarGlobe, the unknown spatiotemporal dynamics in the climate can be uncovered, and the regional to global climate complexity, as well as the interactions between the local terrain and changes in the atmosphere, can be better understood. With the addition of the time dimension, the temporal change of climate variables is captured in real-time. To sum up, PolarGlobe responds directly to the former President Barack Obama's Climate Action plan by providing a powerful tool that help uncover the driving factors of the extreme weather and climate disasters. Besides supporting climate science (Seckel, 2018), the techniques reported in this paper can be easily adapted to visualize data in other Earth science domains, such as oceanography and polar sciences. PolarGlobe also provides an easy-to-access and user-friendly graphic interface by which not only scientists but also general public and younger generations would be capable of interactively viewing and understanding our living planet.

In the future, we will further advance the PolarGlobe platform by developing an adaptive and general interface to allow the fusion and visualization of climate data from diverse simulation models and sources. We will also pack the PolarGlobe system to make it open source, thus benefiting the broader climate science and scientific visualization communities.

Acknowledgements

This project is supported by the National Science Foundation (BCS-1455349 and PLR-1504432).

References

Ayachit, U., Geveci, B., Moreland, K., Patchett, J., & Ahrens, J. (2012). *The ParaView visualization application. High perform. vis. extreme-scale sci.* (Insight 383–400).

Beccario, C. (2019). Earth wind map. http://earth.nullschool.net.

Bürger, K., Schneider, J., Kondratieva, P., Krüger, J. H., & Westermann, R. (2007). Interactive visual exploration of unsteady 3D flows. *EuroVis,* 251–258.

Cabral, B., & Leedom, L. C. (1993). Imaging vector fields using line integral convolution. *Proceedings of the 20th annual conference on computer graphics and interactive techniques* (pp. 263–270). ACM.

Cesium (2019). An open-source JavaScript library for world-class 3D globes and maps. https://cesiumjs.org/.

Doraiswamy, H., Natarajan, V., & Nanjundiah, R. S. (2013). An exploration framework to identify and track movement of cloud systems. *IEEE Transactions on Visualization and Computer Graphics, 19*, 2896–2905. https://doi.org/10.1109/TVCG.2013.131.

Dyer, J., & Amburn, P. (2010). Desktop visualization of meteorological data using paraview. *Kitware Source, 14*, 7–10.

Edmunds, M., Laramee, R. S., Chen, G., Max, N., Zhang, E., & Ware, C. (2012). Surface-based flow visualization. *Computers and Graphics, 36*, 974–990.

Edmunds, M., Laramee, R. S., Malki, R., Masters, I., Croft, T. N., Chen, G., & Zhang, E. (2012). Automatic stream surface seeding: A feature centered approach. *Computer Graphics Forum, 31*, 1095–1104. https://doi.org/10.1111/j.1467-8659.2012.03102.x.

Falk, M., & Weiskopf, D. (2008). Output-sensitive 3D line integral convolution. *IEEE Transactions on Visualization and Computer Graphics, 14*, 820–834. https://doi.org/10.1109/TVCG.2008.25.

GEOS-R. GEOS-17 post-launch testing and transition to operations [WWW document]. (2018). https://www.goes-r.gov/users/transitionToOperations17.html (accessed 2.25.19).

Han, J., & Pan, H.-L. (2011). Revision of convection and vertical diffusion schemes in the NCEP global forecast system. *Weather and Forecasting, 26*, 520–533. https://doi.org/10.1175/WAF-D-10-05038.1.

Hansen, C. D., & Johnson, C. R. (2011). *Visualization handbook.* Elsevier.

Helbig, C., Bauer, H.-S., Rink, K., Wulfmeyer, V., Frank, M., & Kolditz, O. (2014). Concept and workflow for 3D visualization of atmospheric data in a virtual reality environment for analytical approaches. *Environment and Earth Science, 72*, 3767–3780. https://doi.org/10.1007/s12665-014-3136-6.

Ingram, R. G., & Chu, V. H. (1987). Flow around islands in Rupert Bay: An investigation of the bottom friction effect. *Journal of Geophysical Research, Oceans, 92*, 14521–14533. https://doi.org/10.1029/JC092iC13p14521.

Johansson, J., Opach, T., Glaas, E., Neset, T., Navarra, C., Linnér, B., & Rød, J. K. (2017). VisAdapt: A visualization tool to support climate change adaptation. *IEEE Computer Graphics and Applications, 37*, 54–65. https://doi.org/10.1109/MCG.2016.49.

Johnson, C., Moorhead, R., Munzner, T., Pfister, H., Rheingans, P., & Yoo, T. S. (2005). *NIH-NSF visualization research challenges report.* Institute of Electrical and Electronics Engineers.

Keena, N., Etman, M. A., Draper, J., Pinheiro, P., & Dyson, A.. Interactive visualization for interdisciplinary research [WWW document]. (2016). https://www.ingentaconnect.com/content/ist/ei/2016/00002016/00000001/art00016 (accessed 2.25.19).

Laramee, R. S., Hauser, H., Doleisch, H., Vrolijk, B., Post, F. H., & Weiskopf, D. (2004). The state of the art in flow visualization: Dense and texture-based techniques. *Computer graphics forum* (pp. 203–221). Wiley Online Library.

Li, W., Shao, H., Wang, S., Zhou, X., & Wu, S. (2016). Chapter 9 - A2CI: A cloud-based, service-oriented geospatial cyberinfrastructure to support atmospheric research. In T. C. Vance, N. Merati, C. Yang, & M. Yuan (Eds.). *Cloud computing in ocean and atmospheric sciences* (pp. 137–161). Academic Press. https://doi.org/10.1016/B978-0-12-803192-6.00009-8.

Li, W., & Wang, S. (2017). PolarGlobe: A web-wide virtual globe system for visualizing multidimensional, time-varying, big climate data. *International Journal of Geographical Information Science, 31*, 1562–1582. https://doi.org/10.1080/13658816.2017.1306863.

Lynch, C. (2008). Big data: How do your data grow? *Nature, 455*, 28–29. https://doi.org/10.1038/455028a.

McLoughlin, T., Laramee, R. S., Peikert, R., Post, F. H., & Chen, M. (2010). Over two decades of integration-based, geometric flow visualization. *Computer graphics forum* (pp. 1807–1829). Wiley Online Library.

Openlayers (2019). A high-performance, feature-packed library for all your mapping needs. https://openlayers.org/.

Post, F. H., Vrolijk, B., Hauser, H., Laramee, R. S., & Doleisch, H. (2003). The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum,* *22*, 775–792. https://doi.org/10.1111/j.1467-8659.2003.00723.x.

Rautenhaus, M., Böttinger, M., Siemen, S., Hoffman, R., Kirby, R. M., Mirzargar, M., ... Westermann, R. (2018). Visualization in meteorology—A survey of techniques and tools for data analysis tasks. *IEEE Transactions on Visualization and Computer Graphics, 24*, 3268–3296. https://doi.org/10.1109/TVCG.2017.2779501.

Rezk-Salama, C., Hastreiter, P., Teitzel, C., & Ertl, T. (1999). Interactive exploration of volume line integral convolution based on 3D-texture mapping. *Proceedings of the conference on Visualization'99: Celebrating ten years* (pp. 233–240). IEEE Computer Society Press.

Santos, E., Poco, J., Wei, Y., Liu, S., Cook, B., Williams, D. N., & Silva, C. T. (2013). UV-CDAT: Analyzing climate datasets from a User's perspective. *Computing in Science & Engineering, 15*, 94–103. https://doi.org/10.1109/MCSE.2013.15.

Schnase, J. L., Duffy, D. Q., Tamkin, G. S., Nadeau, D., Thompson, J. H., Grieg, C. M., ... Webster, W. P. (2017). MERRA analytic services: Meeting the big data challenges of climate science through cloud-enabled climate analytics-as-a-service. *Comput. environ. urban syst., geospatial cloud computing and big data. Vol. 61. Comput. environ. urban syst., geospatial cloud computing and big data* (pp. 198–211). . https://doi.org/10.1016/j.compenvurbsys.2013.12.003.

Seckel, S. (2018). *ASU professor creates climate data visualization tool that can reveal changes in the atmosphere in real time [WWW document]*. ASU Access Excell. Impact. URL https://asunow.asu.edu/20181001-creativity-illustrating-dance-earth (accessed 2.25.19) .

Shen, B. W., Nelson, B., Tao, W. K., & Lin, Y. L. (2013). Advanced visualizations of scale interactions of tropical cyclone formation and tropical waves. *Computing in Science & Engineering, 15*, 47–59. https://doi.org/10.1109/MCSE.2012.64.

Skytland, N. (2012). *What is NASA doing with big data today. NASA 31 May 2018*.

Spencer, B., Laramee, R. S., Chen, G., & Zhang, E. (2009). Evenly spaced streamlines for surfaces: An image-based approach. *Computer graphics forum* (pp. 1618–1631). Wiley Online Library.

Wang, S., Li, W., & Wang, F. (2017). Web-scale multidimensional visualization of big spatial data to support earth sciences—A case study with visualizing climate simulation data. *Informatics, 4*, 17. https://doi.org/10.3390/informatics4030017.

Wang, F., Li, W., Wang, S., & Johnson, C. R. (2018). Association rules-based multivariate analysis and visualization of spatiotemporal climate data. *ISPRS Int. J. Geo-Inf. 7*, 266. https://doi.org/10.3390/ijgi7070266.

Warne, D. J., Larsen, G., Young, J., & Cox, M. E. (2013). Image-based flow visualisation (IBFV) to enhance interpretation of complex flow patterns within a shallow tidal barrier estuary. *Environ. Model. Softw. 47*, 64–73. https://doi.org/10.1016/j.envsoft.2013.04.007.

Wegenkittl, R., Groller, E., & Purgathofer, W. (1997). Animating flow fields: Rendering of oriented line integral convolution. *Computer Animation'97* (pp. 15–21). IEEE.

Wijk, J. J. V. (2003, October). Image based flow visualization for curved surfaces. *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (pp. 17). IEEE Computer Society.

van Wijk, J. J. (2002). Image based flow visualization. *Proceedings of the 29th annual conference on computer graphics and interactive techniques, SIGGRAPH '02* (pp. 745–754). New York, NY, USA: ACM. https://doi.org/10.1145/566570.566646.

Ye, X., Kao, D., & Pang, A. (2005). Strategy for seeding 3D streamlines. *Visualization, 2005* (pp. 471–478). IEEE.

Zheng, X., & Pang, A. (2003). HyperLIC. *Visualization, 2003* (pp. 249–256). IEEE.

Zöckler, M., Stalling, D., & Hege, H.-C. (1997). Parallel line integral convolution. *Parallel Computing, 23*, 975–989.