Geometric Quantification of Features in Large Flow Fields

Wesley Kendall and Jian Huang*
The University of Tennessee, Knoxville

Tom Peterka†
Argonne National Laboratory

ABSTRACT

Interactive exploration of flow features in large-scale 3D unsteady flow data is one of the most challenging problems in visualization today. In an effort to comprehensively explore the complex feature spaces in these datasets, we have designed a scalable framework for investigating a multitude of characteristics from traced fieldlines. This new capability has allowed us to examine various neighborhood-based geometric attributes in concert with other scalar quantities, a type of analysis that was not previously possible due to the large computational overhead and I/O requirements. We have integrated visual analytics methods into our approach by allowing users to procedurally and interactively describe and extract high-level flow features. In this work, we show the generality and expressiveness that is offered by this approach, and we show its efficacy by exploring various phenomena in a large global ocean modeling simulation.

Keywords: Geometric Flow Analysis, Feature Extraction, Particle Tracing, Parallel I/O, Large-Scale Data Analysis

1 INTRODUCTION

Flow fields, especially 3D unsteady flow fields, contain a vast space of information that can be expressed as geometric, temporal, spatial, and derived scalar characteristics. Quantifying global-scale phenomena using these characteristics is essential for understanding qualitative events in a broad set of application domains such as oceanography, climate modeling, and geophysics. In many cases, the phenomena are qualitatively well understood but quantitatively hard to specify in temporal and 3D spaces.

One example is a cold-core oceanic eddy a phenomenon defined by a swirling region of ocean with the center of the vortex being cooler than the outside. One could define this feature by tracing the flow and extracting the traces that follow a rotational pattern. This then begs the question of how to define the rotation, and more importantly how to describe the cooling effect near the center of the vortex. The search space to discover cold-core eddies is vast, because many parameters affect the outcome of the feature characterization. For instance, large- and small-scale eddies will have to be defined with different rotational parameters. Temperature measurements are also dependent on the geographical area of the ocean. A second example is global circulation across the entire ocean. These examples together are important to analytical questions. For example, what locations are the most likely to collect contaminated debris from the Fukushima nuclear power plant accident of 2011?

For those analytical needs that involve large flow fields, domain experts need to effectively query data at a higher level of abstraction, and they need the ability to quickly iterate and refine their hypotheses. Aiming to meet this overall goal, we developed a complete analytical system using ocean modeling as a target application. We notice that a system of such kind does not exist in this application domain.

*e-mail: {kendall, huangj}@eecs.utk.edu
†e-mail: tpeterka@mcs.anl.gov

Interactively pruning a problem space on extreme-scale flow datasets brings significant challenges. While interactivity may not be a problem for small-scale data, the focus on parallelism and its relationship to the feature extraction process becomes the most influential factor for maintaining interactivity on large data. Here, we have specifically focused on this issue. We have directly coupled graphical and procedural characterization of geometric flow features (i.e. attributes computed directly from traced fieldline geometry) with two highly scalable data processing and feature extraction components.

The scalability of our system paired with expressive feature characterization is powerful. It has allowed us to examine major ocean currents as well as long-term ocean eddy activities. Key to these is the scalable analysis power that allows a user to interactively assess geometric flow features defined on a variable-length window. To our knowledge, this capability is novel.

We have chosen to focus on geometric flow field features for two reasons. First, traced flow geometry is often orders-of-magnitude smaller than the original scalar representation of the flow, making it a viable reduced representation of the original data. Second, there are a multitude of derived geometric characteristics that may be computed from traced geometry. The expressiveness offered by geometric features has already been shown [12], and, as we will show, geometric features with a variable-length neighborhood size are powerful for feature characterization.

2 A PARALLEL PIPELINE FOR GEOMETRIC FLOW ANALYTICS

Users interact with our data analytics system through a remote interface. Our back-end system has a pipeline of components, where every component is inherently parallel. The pipeline is illustrated in Figure 1. In the following, we describe the overall data flow of our pipeline before detailing each major component in later sections.

Our system relies on an external simulation to produce time-varying flow fields. We access the flow data via a parallel file system. In our usage examples (Section 6) and driving application (Section 7), the simulations produced velocity components and other scalar values as tuples on a rectilinear grid stored across multiple files for each timestep.

We use OSUflow, a leading flow tracing package [8], to generate fieldlines. The default OSUFlow configuration densely seeds the flow field with uniform distribution in the spatial and temporal domains. We allow users to control the density of seeding. OSUflow then computes flow tracing in parallel. Other than seeding, our system is oblivious of whether the flow is steady or unsteady and how flow tracing is computed. Due to this reason, our system uses fieldline as a term to generally represent pathlines for unsteady flows and streamlines for steady flow. We note, however, all test datasets used in this paper are unsteady 3D flow fields, and all geometric features are computed from pathlines.

With fieldlines as the geometry, our system implements a scalar merging step to interpolate scalar values so that scalar characteristics (such as salinity and temperature values) are associated with each fieldline point.

After the above steps, the analytics problem is now transformed into a completely data-parallel problem where querying data can be easily parallelized. We use the Scalable Querying Interface (SQI) [3] to support parallel querying. Users construct queries in a

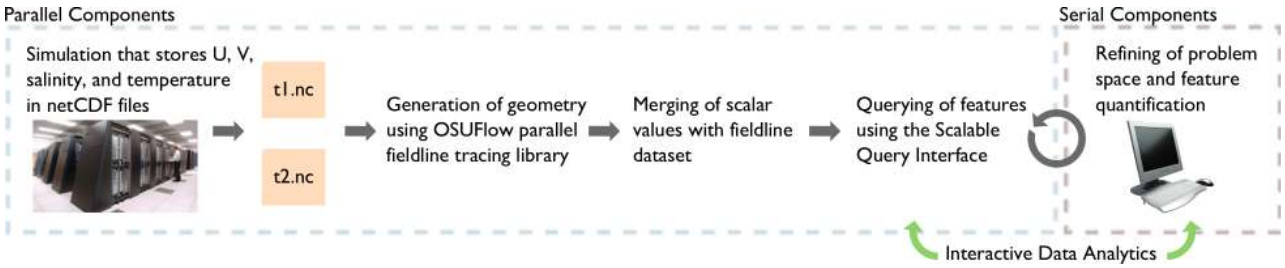
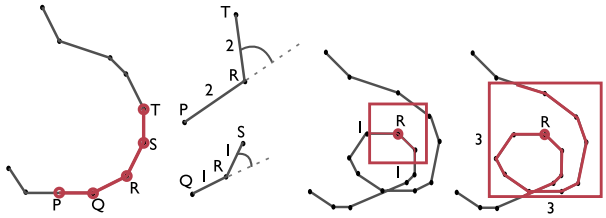


Figure 1: Example of our pipeline for large-scale analytics of flow data. We provide the example of a simulation dumping off many files in the netCDF format with multiple variables. We outline parallel and serial components separately. The interactive analysis process occurs between a remote client and a querying backend. Section 2 provides an overview of the pipeline.

procedural manner with a remote interface, send them to the backend querying system, and the interface renders queried data interactively. The user iteratively refines the feature description and extracts the data during exploration. Resulting feature specifications can be stored in a concise format and used for either sharing or extracting the feature again for other parallel analysis.

In the following sections, we describe the details of our work in parallel geometry processing, feature characterization, feature extraction, and visualization.



(a) Angle of Turn (b) Residence

Figure 2: 2D examples of our neighborhood-based geometric features. Angle of turn is calculated by the angle of vectors formed from the end points at varying arc lengths from a vertex. Residence length is computed by the arc length of a fieldline that resides in a box around a vertex. The residence time can also be computed by dividing this length by the velocity magnitude along the line.

3 Scalable Geometry Processing

As noted in [7], the sheer volume of flow datasets presents a challenge for visualization. In the context of our application, the processing of the geometry (i.e. the fieldline tracing and scalar merging) deals with the entire simulated flow dataset. We use methods to scale the geometry processing to today's largest architectures in order to alleviate the data-intensive nature of this step.

For parallel fieldline tracing, we use recent research implemented in the OSUFlow particle tracing library [8]. We treat OSUFlow as a black box and interact with it primarily by seeding the flow traces. In the following, we describe how we use OSUFlow within our system. Due to space limit, we refer readers to [8] for details such as particle-termination criteria and integration techniques.

OSUFlow operates in a distributed-memory environment, where nodes must explicitly pass data to one another using the message-passing model. OSUFlow also traces time-varying fieldlines out of core, only loading the time span necessary during particle tracing.

It is important to normalize the dataset so that the unit of the physical space velocity matches with scales of the temporal and spatial grids. OSUFlow assumes this has been done. On startup, OSUFlow partitions the dataset into 4D blocks and assigns them round robin to processes. For better load balancing, OSUFlow partitions the dataset at a fine granularity and assigns eight blocks per process. OSUFlow then sends the partition to the Block I/O Layer (BIL) [4]. BIL is a high-performance parallel I/O library that operates on multivariable datasets stored across many files. Results from [4] show its ability to utilize up to 90% of the available bandwidth on modern parallel file systems. In our application examples, BIL reads 4D blocks spread across multiple files in the netCDF format.

After BIL reads the data, OSUFlow randomly places particles in a uniform distribution by a density set by the user. It then integrates the flow field using a 4^{th} order adaptive step-size Runge-Kutta integration kernel. OSUFlow transfers particles that exit a block's subdomain to a communication queue. Processes synchronize during tracing and exchange outlying particles to the owner of the proper subdomain. Particles progress at the same speed through time. This behavior allows OSUFlow to fetch timesteps during tracing and stream datasets that cannot entirely fit in memory. After particle tracing, the partial fieldline traces reside across the processes. We stitch the partial fieldlines together by using parallel sample sorting algorithm to sort the geometry while keeping it distributed across processes.

The system then proceeds to merge other useful scalar attributes in with the resulting geometry (i.e. scalar merging). To perform this step, we first assign each process a block of data and read all of the associated scalar values in parallel using BIL. After this, we read the computed geometry in parallel and bin the fieldline vertices to the processes that contain their bounding block of scalar data. Depending on the data, the scalar-merging step either computes attributes (such as divergence and vorticity of the flow) or interpolates attributes (such as temperature and salinity variables). We then take the resulting points (which are now $u, v, w, scalar1, scalar2, \ldots$ tuples) and store them back in their original geometric ordering. Our implementation of scalar merging assumes that the data can be held entirely in memory. This restriction can be removed by an implementation using out-of-core data streaming.

4 Feature Specification and Procedural Characterization

We utilize a rich feature set and characterization method for extracting features. Our feature set consists of the following quantities.

- Dimensional Values - Each point of the fieldline contains x, y, z, t tuples that allow the user to relate dimensional information of the fieldline to the feature of interest. For example, one would need this information to extract turbulent flow regions that progress downwards in the z direction through time.

- Scalar Quantities - Each point of the fieldline also contains scalar properties that are useful for analyses. For the ocean and atmospheric examples in this work, temperature, pressure, salinity, CO_2, and other properties are often of inter-

est when relating streamline geometry to a feature of interest. Furthermore, other derived quantities can be computed from the scalar fields such as flow divergence and vorticity. Scalar quantities such as these allow for more expressiveness in feature characterization, for example, allowing the user to extract fieldlines that travel through areas of high pressure and high temperature gradients.

- Neighborhood-Based Geometric Quantities - The heart of our feature specification comes from computing various geometric properties for each point of the fieldlines. These properties are defined based on a variable-sized neighborhood, which is a meaningful parameter in regards to specifying transient and long-term flow features. Furthermore, these variables can be computed on the fly since they are only dependent on the fieldline data. The neighborhood-based geometric attributes we explore are angle of turn and residence. The properties and computation of these features are discussed in the following.

Angle of Turn Our use of angle of turn is patterned after the use of winding angle, which can be computed along the fieldline and used to find swirling patterns such as vortex cores [10, 11]. Another similar example is computing the curvature along the fieldline [11, 12]. For finer granularity, we used a variable-length window along the fieldline and computed the angle of turn at each vertex. Figure 2a illustrates this. Given a point R and a neighborhood radius of one, we move an arc length of one to the right to get point S and an arc length of one to the left to get point Q. The angle of turn is computed by subtracting the angle formed by $\overrightarrow{RS}$ and $\overrightarrow{RQ}$ from 180°. Similarly, we can use a neighborhood radius of two to compute angle of turn over a longer length. Using combinations of various neighborhood sizes offers intuitive methods when describing fieldlines with swirling characteristics or fieldlines that stay straight or make abrupt transitions.

Residence Residence describes the residence length and residence time of the fieldline at each of its vertices. A 2D illustration is provided in Figure 2b. Given a vertex R, we create a box around it and measure the total arc length of the fieldline from the right and left of R until it exits the box. Residence time is computed by taking the summation of the arc length of every segment inside the box and dividing it by the average velocity magnitude of the two points on each segment.

Residence length and time provide an interesting span space. High length and low time indicate fast swirling areas, while high length and high time indicate slower swirling regions. Low length and high time are useful for viewing fieldlines moving towards a critical point while low length and low time indicate fast and relatively straight fieldlines. Similar to angle of turn, these attributes may be computed at varying neighborhood sizes to assess small- and large-scale phenomena.

Feature Characterization Using Query Trees Procedural feature characterization has been used in a variety of research for scalar datasets. The most related example is SimVis [1], which uses a feature definition language to apply logical operations to ranges of scalars. Another is the approach of [14], which uses set operations to procedurally combine and compare queried volumes. We use a similar concept known as "query trees." Query trees are the formulation of a tree that contains queries as leaves and various logical operations ($\neg$, $\wedge$, $\vee$, $\oplus$) at the nodes of the tree. The queries themselves are Boolean range queries and may be used to restrict ranges of any attribute of the dataset, whether it be spatial, temporal, scalar, or geometric. In contrast to scalar-based methods such as those in SimVis, our queries operate on individual vertices and return the entire fieldline geometry when a user-specified threshold of vertices of the fieldline match the given query.

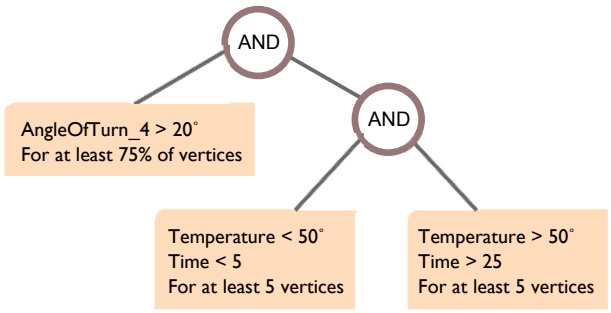


Figure 3: An example of a query tree. The left-most query extracts fieldlines that primarily swirl. The other two queries ensure the fieldlines start with a temperature below 50 degrees and end with a temperature above 50 degrees.

Figure 3 shows an example of a query tree. In this example, the left-most node issues a query for fieldlines that have an angle of turn (within a neighborhood size of four) that is greater than 20 degrees for at least 75% of the vertices on the fieldline. The other two queries specify fieldlines that have a temperature lower than 50 degrees for at least five vertices in the first five timesteps, and a temperate higher than 50 degrees for at least five vertices in the last five timesteps. The logical *and* of all of the queried data returns fieldlines that "mostly rotate while starting cool and then finishing warm."

The degree of freedom presented by query trees allows specification of concepts like "fieldlines that stay straight for half of their existence and also exhibit swirling patterns for at least ten percent of their continuation." Similarly, it also allows flexibility in relating fieldline geometry to scalar, spatial, and temporal quantities. It is easy to formulate "fieldlines that cross through the upper right quadrant of the dataset, avoid the center region, and then go through the bottom left quadrant" with query trees. We provide more examples of these types of concepts in Section 6. We discuss how the queries are remotely processed and visualized in the next section.

5 Parallel Feature Extraction and Visualization

Query trees integrate smoothly into the parallel pipeline since we utilize a large-scale querying system built with the Scalable Query Interface (SQI) [3]. The querying system starts by assigning and reading fieldline data in parallel using $MPI\_File\_read\_all$. Once data is read, fieldlines are shuffled among processes for better load balancing during querying. SQI then builds a distributed search structure on top of the vertices to extract relevant data when querying.

After the search structure is built, we have to maintain additional data structures to aid in processing queries. During querying, a count array maintains the amount of vertices that matched the query for each fieldline. A bit in a bitfield is set for each fieldline that has the user-specified threshold of vertices matching the query. This bitfield is maintained during recursive processing of the query tree and combined at the nodes during logical operations with other bitfields. The final bitfield at the root of the tree describes all fieldlines that matched the query.

The user has the ability to limit the amount of fieldlines that are returned. This is useful for creating clutter-free visualizations and for simply overviewing the entire flow field. Since we originally distribute the fieldlines randomly, each process returns up to F/P randomly chosen fieldlines from their local results, where F is the limit set by the user and P is the number of processes. Although this method does not always exactly adhere to the set limit, it does provide a fast approximation that avoids unnecessary communication. Once this step is over, the resulting fieldlines are gathered to the

(a) Low Residence Time (b) High Angle of Turn

Figure 4: Examples from the tornado dataset using (a) low residence time to show the fast and relatively straight flow, and (b) high angle of turn to highlight a vortical region. In (b), we have also included a snapshot of the extracted vortex core using a region-growth algorithm.

root process and then streamed to the interface in packed messages.

Visualization is performed on the client side using traditional flow visualization techniques. We render cylinders or lines for the fieldlines and use up to three variables of interest for controling the color, width, and opacity of the fieldlines. For providing context information in visualizations, the user has the option to use as many query trees as they want, with each tree representing a different feature in the same viewport. This allows them to highlight different features of interest or show them in the context of the entire dataset.

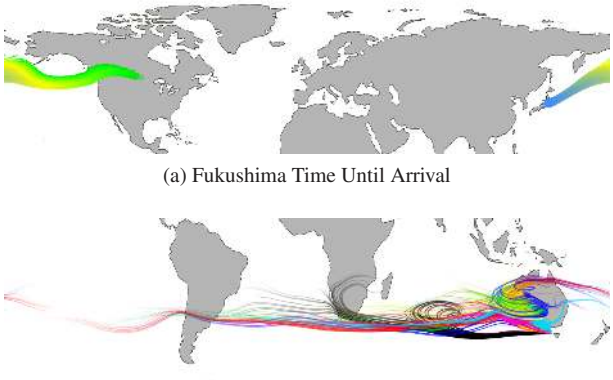
6 Representative Usage Examples

We provide usage examples from two small-scale datasets to show the flexibility of our approach. The first dataset is a tornado simulation dataset. It is stored as floating-point u,v,w tuples on a $128 \times 128 \times 128$ grid across 50 timesteps. Roger Crawfis at the Ohio State University provided the tornado dataset. The second dataset is a subset of the GEOS-5 atmospheric modeling dataset that was provided by Leslie Ott at the NASA Goddard Space Flight Center. Our subset of GEOS-5 dataset simulates atmospheric windfields (floating-point u,v,w tuples) in daily intervals across two years (starting at year 2000) on a $288 \times 181 \times 72$ curvilinear grid. Eight different parameter runs are present, which start the meteorology with different initial parameters. The different parameter runs are useful for studying the sensitivity and convergence of the conditions over time.

We note that the parameters used in queries are specific to the dataset, and the units of measurement for the geometric queries are based on the grid spacing and velocity magnitudes. What this means, for example, is that a residence time of 120 could be considered a "low" time for the tornado dataset but have a different meaning in other datasets. Also, when using the term "neighborhood size" for geometric attributes, we are specifically talking about either the radius of the arc length (for angle of turn) or the radius of the box (for residence) in grid units.

Neighborhood-Based Geometric Feature Examples Here, we use the tornado dataset to show examples of using neighborhood-based geometric characteristics to extract features.

To view the areas of fast and relatively straight flow, we issued a query for fieldlines that contain low residence times (< 120) in a neighborhood size of 4 for at least 5% of their vertices. The result is shown in Figure 4a. The visualization shows the fieldlines around the core of the tornado that have the high velocities. The width of the lines is modulated by time, showing the clockwise rotation of



(a) Fukushima Time Until Arrival

(b) Cape Grim Incoming Flow Variability

Figure 5: Dimesion-based flow features. The first figure shows a query for Fukushima starting in March 2000 and colors the lines based on time. The second figure shows the incoming flow into Cape Grim in May 2000 for three months in advanced. The flow is colored by different model runs to examine internal model variability.

the tornado. The color is modulated by velocity magnitude, showing the inner parts of the funnel (in purple) that go faster than the outer parts (in blue) that have lost speed.

To view the core and vortical regions of the tornado, we issued a query for fieldlines that have high angle of turn ($> 25°$) in a neighborhood size of 2 for at least 25% of their vertices. We also queried the entire dataset to provide context information and restricted the result to 200 random fieldlines to avoid visual clutter. The visualization is shown in Figure 4b. The core of the tornado is shown with a summary of the surrounding fieldlines. The color of the core structure is modulated by time; one can observe that the funnel of the tunnel moves upward from lower timesteps (in blue) to later timesteps (in purple). The width of the core structure is modulated by velocity magnitude, showing decreased velocity when the flow reaches the top of the tornado. To the right of the rendering, we have also included a snapshot of the vortex core that was extracted from one timestep of the dataset using a region-growth algorithm from [2] that segments vortex core areas. As expected, the queried vortical area closely matches the extracted core line. This suggests that angle of turn could be a viable feature in the detection of vortices, however, future study is needed to confirm this.

Dimension-Based Feature Examples As described earlier, we can relate dimensional and scalar values to flow features of interest. We use the GEOS-5 atmospheric modeling dataset to illustrate this.

In the first example, we examined the patterns of the flow field directly after the Fukushima nuclear disaster that happened in March of 2011. Although our dataset is simulated across year 2000 - 2001, the yearly patterns of the flow field stay relatively similar from year to year. We issued a query for the fieldlines that started in the lower atmospheric layers of March 2000. We show a rendering from one parameter run of the dataset in Figure 5a. The fieldlines are colored by time, with blue representing 1 – 3 days, yellow 3 – 5 days, and green 5 – 7 days. The visualization verifies real-world events since low-level radioactive particles reached the United States within seven days of the Fukushima disaster.

In the second example, we examine the variability of the different parameter runs in the GEOS-5 dataset. According to our atmospheric scientists, Cape Grim, Tasmania is an area of high variability among parameter runs of the GEOS-5 model. We examined the variability of the flow coming into Cape Grim by issuing a query

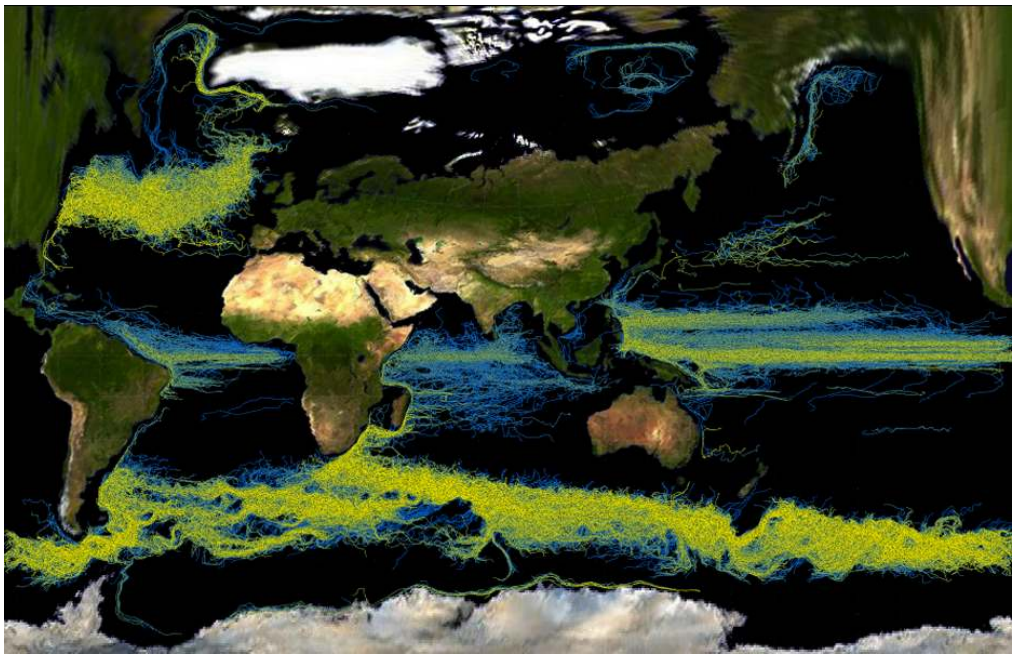


Figure 6: A global overview of the major ocean currents in the POP dataset. The currents were extracted by querying for fieldlines that exhibited very low residence time for most of their existence. Many of the major currents, including the Equatorial, Antarctic Circumpolar, and Gulf are easily observed. The color is modulated by yellow for deep to blue for shallow ocean currents.

for fieldlines across all eight parameter runs that flow directly into Cape Grim. A visualization showing fieldlines flowing into Cape Grim in May 2000 with a three-month lead time is shown in Figure 5b. The eight colors represent the eight parameter runs. Big differences are seen in the black, magenta, and light-blue fieldlines, which arrive from the strong westerly winds. The other parameter runs appear to be driven downward by a large vortex located near Cape Grim, which could be one of the main factors driving the large variability among the parameter runs.

7 Driving Application – Large Scale Oceanic Feature Quantification

Our driving application is to visualize flow features from the Parallel Ocean Program (POP), a high-resolution eddy-resolving ocean circulation model [5]. POP was started using observational analysis, and the general circulation is well represented. To allow inclusion of the Arctic Ocean, it employs a displaced tripole grid. The grid is 2.5D and has 40 layers of u and v velocity components at a 3,600 by 2,400 resolution that spans monthly from February 2001 to September 2003. Along with the u and v components, POP generates salinity and temperature variables. The dataset is 165 GB.

Our analyses were conducted on *Intrepid*, a Blue Gene/P supercomputer at the Argonne National Laboratory. Intrepid contains 40,960 nodes consisting of quad-core 850 MHz IBM PowerPC processors. Results were streamed over the Internet and the resulting fieldlines were rendered interactively using an NVidia Quadro FX 3800 graphics card.

We demonstrate ocean flow features characterized using windows of variable lengths in Section 7.1. The primary results – to demonstrate scalability of our data analytics system – are shown in Section 7.2. All of the coauthors are computer scientists and due to that reason, results in Section 7.1 are meant solely for evaluating an analytical technique and should not be considered with significance in the field of oceanography.

Post et al. [9] surveyed flow feature extraction methods in 2003. In the framework set up by that survey, we did not find previous work that, like our work, characterized qualitative events such as "major ocean currents," although many researchers had studied vortex extraction. The most relevant and recent is a point based method in 2011 [13], where the authors extracted 2D eddies using each spatial location's vorticity and strain. Our method is a curve-based method. According to Sadarjoen et al. [10], curve-based methods are invariant to the rotational speed of the vortex and hence often more useful in the general detection of vortices.

7.1 Exploration of Eddies and Major Currents

Before exploration, we traced roughly two million fieldlines to capture the entirety of the flow field. Although not used in our primary examples, we computed vorticity, divergence, velocity magnitude, and gradient magnitude for our tests. We also stored the salinity and temperature variables from the dataset with the fieldline geometry. The stored data was roughly 3 GB, which is a significant reduction from the original 165 GB. Once the data was loaded into memory, we computed angle of turn, residence time, and residence length at neighborhood sizes with radii from 1 to 128 in powers of 2. The in-memory footprint of the data was roughly 10 GB.

The first features we aimed to quanity were the major ocean currents. To do this, we examined the areas of the ocean that exhibited swift and relatively straight movement. Specifically, we queried for fieldlines that had a low residence time (< 10) with a neighborhood size of 8 for all of their vertices. A summary visualization of 10,000 randomly sampled fieldlines from this query is shown in Figure 6, and it is colored by shallow areas of the ocean (in blue) to deeper areas (in yellow). Some of the major currents that can be observed in this figure are the Equatorial Currents, which travel almost perfectly horizontal, and the Antarctic Circumpolar Current, the most dominant current in the Southern Hemisphere. The Gulf Stream is also highly recognizable, bordering the continental shelf of the United States and flowing towards Europe. Some other smaller currents include the Alaskan Current and the Labrador Current, which flows between Greenland and Canada. Another small current is the Beaufort Gyre, a shallow, wind-driven current in the Arctic Ocean.

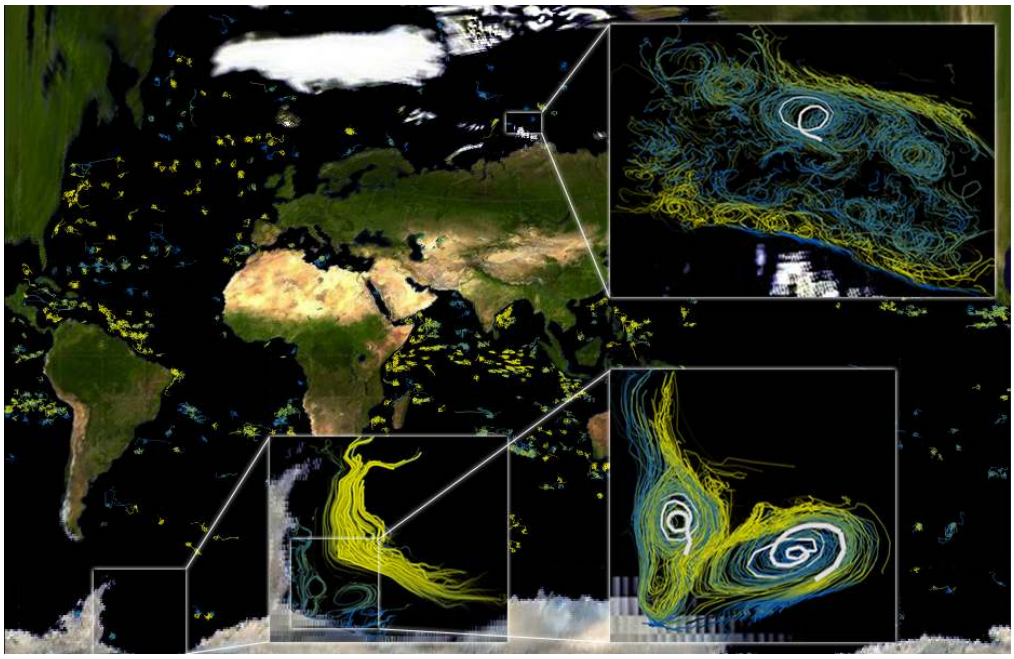


Figure 7: A global overview of some of the major ocean eddies in the POP dataset. These were found by querying for fieldlines that exhibited high residence length for the majority of their existence. Some of the areas are magnified and colored to show cold-core eddy activity. The color mappings are explained in Section 7.

Processes	Fieldline Tracing	Scalar Merging	Startup	Geometric Attributes	Query	Network Latency	Overall Latency
64	N/A	N/A	31.20	14.38	0.43	0.039	0.47
128	N/A	N/A	15.18	7.31	0.22	0.037	0.26
256	67.68	134.27	7.82	3.67	0.11	0.037	0.15
512	47.89	91.45	4.32	1.84	0.061	0.042	0.10
1,024	36.24	70.56	2.71	0.93	0.034	0.058	0.092

Table 1: Average application timing results (in seconds) for the global ocean explorations depicted in Figures 6 and 7.

The major currents have considerable effects on various small-scale phenomena in the ocean. One of the primary benefits of simulating ocean currents at such high resolution is the ability to resolve smaller high-turbulence areas such as eddies. Eddies can range from centimeters to hundreds of kilometers in diameter and can persist for periods of days to many months. We examined various long-term eddies by querying for fieldlines that have high residence length (> 125) in a neighborhood size of 4 for at least 50% of their vertices. 10,000 randomly sampled fieldlines are shown in the global visualization in Figure 7, and they are colored by shallow eddies (in blue) to deeper eddies (in yellow). The main areas that exhibit these characteristics are close to the shorelines, where major currents usually flow past and create turbulent activity.

Eddies have interesting properties that we observed in more detail. At the bottom of Figure 7, we zoomed into a portion of the Weddell Sea, an area that has attracted attention to eddy activity. The center image at the bottom of Figure 7 shows the relatively straight areas of the current, obtained by querying for fieldlines that had a low angle of turn ($< 25°$) in a neighborhood size of 4 for at least 50% of their vertices. The color is modulated by temperature from cold (in blue) to hot (in yellow), and the width of the lines is modulated by salinity. A higher-salinity and warmer current is observed that appears to be driving turbulent activity close to the Antarctic coast. We zoomed in on this area of turbulent activity and again queried for fieldlines that exhibit a low angle of turn, but this time only for 10% of their vertices. The eddies appear to be cold-core eddies, which are classified by having centers that are cooler than the surrounding flow. Cold-core eddies also have the property of being cyclonic, meaning they rotate clockwise in the Southern Hemisphere and counterclockwise in the Northern Hemisphere. This was verified by examining the centers of the eddies and querying for fieldlines that have a very high residence length (> 200) with a neighborhood size of 8 for at least 20% of their vertices. The returned fieldlines are colored in white and their widths are modulated by time to show the clockwise rotation of the eddies.

A similar experiment was carried out for the Arctic Ocean, another widely studied area for eddy activity [6]. The result, shown at the top of Figure 7, shows cold-core eddy activity that results from a higher temperature flow from above. Analogous to the eddies in the Weddell Sea, we extracted a core region of the major eddy to show its cyclonic nature. Since these cold-core eddies are in the Northern Hemisphere, they rotate counterclockwise.

7.2 Timing Results

We provide timing results from exploration of the POP dataset at scales from 64 to 1,024 processes. Since our scalar merging preprocessing step currently only works on datasets that can be loaded in memory, we only provide preprocessing times from 256 to 1,024 processes. The timing results (in seconds) are shown in Table 1.

The "fieldline tracing" and "scalar merging" columns represent the one-time step that occurs to generate the fieldline geometry and merge scalar quantities with it. At 1,024 processes, we were able to trace roughly two million fieldlines through the entire ocean dataset in 36 seconds. This number includes the time spent reading

the dataset and writing the fieldline geometry. The scalar merging takes about twice as long, primarily because it is reading in additional salinity and temperature quantities from the dataset. Overall, the efficient I/O methods allowed us to obtain a very reasonable preprocessing overhead for even a dataset in the hundreds of gigabytes.

The "startup" and "geometric attributes" timings convey the one-time overhead associated with starting our application. The startup times include I/O overhead and the time it takes to load balance the data and build the necessary querying data structures. The geometric attributes time represents the time it takes to precompute all of the geometric attributes of the fieldlines. Both of these steps showed high scalability. At 1,024 cores, users would be able to recompute an entirely new and extensive feature space in under a second.

The "query," "network latency," and "overall latency" times describe the average time it took to query and the network latency associated with sending the results from the global ocean queries in Figures 6 and 7. The querying times were highly scalable, obtaining an average time of 0.034 seconds at 1,024 processes for the largest global ocean queries that we performed. The network latency actually became the bottleneck at larger scales, which resulted in less scalability for the overall latency of feature extraction. The overall latency times between submitting a query and obtaining the first parts of the result, however, were very interactive.

8 Conclusion and Future Work

In this work, we have shown how data analytics can be used to facilitate the quantification of flow features using geometric and other derived attributes. More importantly, we have shown how two high-performance infrastructures can be integrated into the visual analysis pipeline for interactively exploring very large flow datasets. Using the power of parallel processing is a necessity not just for the processing of large data, but also in allowing exploration of very large feature spaces.

To the best of our knowledge, we are the first to assess the usage of geometric flow features defined on a variable-length window. We believe these types of features can complement other scalar-based approaches that detect vortices and other flow phenomena. As we showed in the paper, these features helped us quantify global eddy and circulation activity in a large-scale ocean simulation. They were also key to defining other events in an atmospheric simulation.

In the future, we would like to experiment with using our system for analysis of even more complex oceanic features. One such example is quantitatively addressing attributes of the Rossby Radius of Deformation, a long standing topic in oceanography. We would also like to examine the usefulness of neighborhood-based geometric features in domains other than flow, primarily those that make use of temporal trend analysis. One other possible avenue of study includes using clustering of fieldline results for visualization purposes.

Acknowledgements

We thank Dr. Robert Jacobs of Argonne National Laboratory for providing us with the initial motivation to work on this research direction. We thank Dr. David Erickson III of Oak Ridge National Laboratory for guidance on studying atmospheric models. We owe Melissa Allen of University of Tennessee for her close collaboration on all technical aspects to properly implement flow advection in atmospheric models. Han-Wei Shen's input was also pivotal to the formation of this work. Our work was funded by DOE SciDAC Ultrascale Visualization Institute (DOE DE-FC02-06ER25778).

References

[1] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proc. of VisSym*, pages 239–248, 2003.

[2] M. Jiang, R. Machiraju, and D. Thompson. A novel approach to vortex core region detection. In *Proc. of VisSym*, pages 217–225.

[3] W. Kendall, M. Glatter, J. Huang, T. Peterka, R. Latham, and R. Ross. Terascale data organization for discovering multivariate climatic trends. In *SC '09: Proceedings of ACM/IEEE Supercomputing 2009*, pages 1–12, Nov. 2009.

[4] W. Kendall, J. Huang, T. Peterka, R. Latham, and R. Ross. Visualization viewpoint: Towards a general I/O layer for parallel visualization applications. *IEEE Computer Graphics and Applications*, 31(6):6–10, Nov./Dec. 2011.

[5] M. E. Maltrud and J. L. McClean. An eddy resolving global 1/10 ocean simulation. *Ocean Modelling*, 8(1-2):31–54, 2005.

[6] T. O. Manley and K. Hunkins. Mesoscale eddies of the arctic ocean. *Journal of Geophysical Research*, 90:4911–4930, 1985.

[7] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. In *Computer Graphics Forum*, volume 29, pages 1807–1829, 2010.

[8] T. Peterka, R. Ross, B. Nouanesengsey, T.-Y. Lee, H.-W. Shen, W. Kendall, and J. Huang. A study of parallel particle tracing for steady-state and time-varying flow fields. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 577–588, May 2011.

[9] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[10] I. Sadarjoen, F.H.Post., B. Ma, D. Banks, and H. Pagendarm. Selective visualization of vortices in hydrodynamic flows. In *Proc. of IEEE Visualization*, pages 419–422, 1998.

[11] I. A. Sadarjoen and F. H. Post. Detection, quantification, and tracking of vortices using streamline geometry. *Computers & Graphics*, 24(3):333–341, 2000.

[12] K. Shi, H. Theisel, H. christian Hege, and H. peter Seidel. Path line attributes - an information visualization approach to analyzing the dynamic behavior of 3d timedependent flow fields. In *In Proc. of TopoInVis*, pages 75–88, 2007.

[13] S. Williams, M. Hecht, M. Petersen, R. Strelitz, M. Maltrud, J. Ahrens, M. Hlawitschka, and B. Hamann. Visualization and analysis of eddies in a global ocean simulation. *Computer Graphics Forum*, 30(3):991–1000, 2011.

[14] J. Woodring and H.-W. Shen. Multi-variate, time varying, and comparative visualization with contextual cues. *IEEE Trans. on Visualization and Computer Graphics*, 12(5):909–916, 2006.