Visualizing Oceanic and Atmospheric Flows with Streamline Splatting

Yinlong Sun[*], Erich Ess[*], David Sapirstein[*], Matthew Huber[§]
[*]*Department of Computer Sciences, Purdue University, W. Lafayette, IN, 47907-1398*
[§]*Department of Earth & Atmospheric Sciences, Purdue University, W. Lafayette, IN, 47907*

ABSTRACT

The investigation of the climate system is one of the most exciting areas of scientific research today. In the climate system, oceanic and atmospheric flows play a critical role. Because these flows are very complex in the span of temporal and spatial scales, effective computer visualization techniques are crucial to the analysis and understanding of the flows. However, the existing techniques and software are not sufficient to the demand of visualizing oceanic and atmospheric flows. In this paper, we use a new technique called streamline splatting to visualize 3D flows. This technique integrates streamline generation with the splatting method of volume rendering. It first generates segments of streamlines and then projects and splats the streamline segments onto the image plane. The projected streamline segments can be represented using a Hermite parametric model. Splatted curves are achieved by applying a Gaussian footprint function to the projected streamline segments and the results are blended together. Thus the user can see through a volumetric flow field and obtain a 3D representation view in one image. The proposed technique has been applied to visualizing oceanic and storm flows. This work has potential to be further developed into visualization software for regular PC workstations to help researchers explore and analyze climate flows.

Keywords: Flow visualization, oceanic and atmospheric flows, streamlines, splatting, volume rendering

1 INTRODUCTION

The relationship between climate and extreme weather events is one of the most exciting areas of scientific research today. The climate system is composed of a host of components—the atmosphere, ocean and lakes, ice, biosphere, and anthroposphere, which interact each other in a complex way. Among these components, oceanic and atmospheric flows are particularly challenging to investigate. This is because real oceanic and atmospheric flows are three-dimensional, turbulent, time-dependent, anisotropic, compressible, non-stationary, non-conservative, forced, and dissipative. Due to the inherent tremendous complexities involved in oceanic and atmospheric flows in the span of temporal and spatial scales, effective computer visualization techniques play an important role in the analysis and understanding of the climate system.

In oceanic and atmospheric flows, vortical structures, such as eddies, waves, and associated swirling patterns, are fundamental to the physical behavior of the fluids. These structures are crucial to better understanding of the climatic and biospheric interactions that they participate in. Midlatitude storms, mesovortices, thunderstorms, and tornadoes are examples of atmospheric vortical phenomena, important for atmospheric heat transport, in the damage that they cause, and in the precipitation associated with them. In the ocean, examples of vertical flows include Gulf Stream meanders, mesoscale eddies, and cold-core rings, which are important for issues such as poleward ocean heat transport and primary productivity.

In the past decade, computer-based techniques for visualizing flows have advanced greatly. However, the existing techniques are still not sufficient to the demand in visualizing oceanic and atmospheric flows. For example, the existing techniques tend to emphasize the most vigorous, persistent, and large-scale currents, and ignore the time-varying, evanescent, and fine-scale aspects of the flows that possess significant values of research. An example of a typical, but obviously unsatisfying visualization of NCOM fields using flow lines is shown in Figure 1. Significant and eventually redundant effort is needed to pick out plotting parameters that emphasize the curliness of the flow at small scales. Another key question is how to display the global view and local features of 3D flows sufficiently and to achieve real-time rendering.

In this paper, we apply a novel technique called streamline splatting to 3D flow visualization. This technique integrates streamline generation with the splatting method of volume rendering. In the first stage we generate segments of streamlines and in the second stage we project and splat the streamline segments on the image plane. The projected streamline segments can be represented using a Hermite parametric model, which simplifies the computation. Curve splatting is achieved by applying a Gaussian footprint function of each projected curve segment on the image plane, and the results are blended together. Thus the user can see through a volume of the flow field and obtain a 3D representation view in one image. The proposed technique has been applied to visualizing ocean and storm flows.

This paper is organized as follows. Section 2 reviews the existing flow visualization techniques and discuss their advantages and disadvantages in the application for oceanic and atmospheric flows. Section 3 describes the computation models and datasets used in this paper. Section 4 presents the visualization technique of streamline splattting. Section 5 shows the results of applying streamline splattting to oceanic and atmospheric datasets. Section 6 gives conclusions and future research directions.

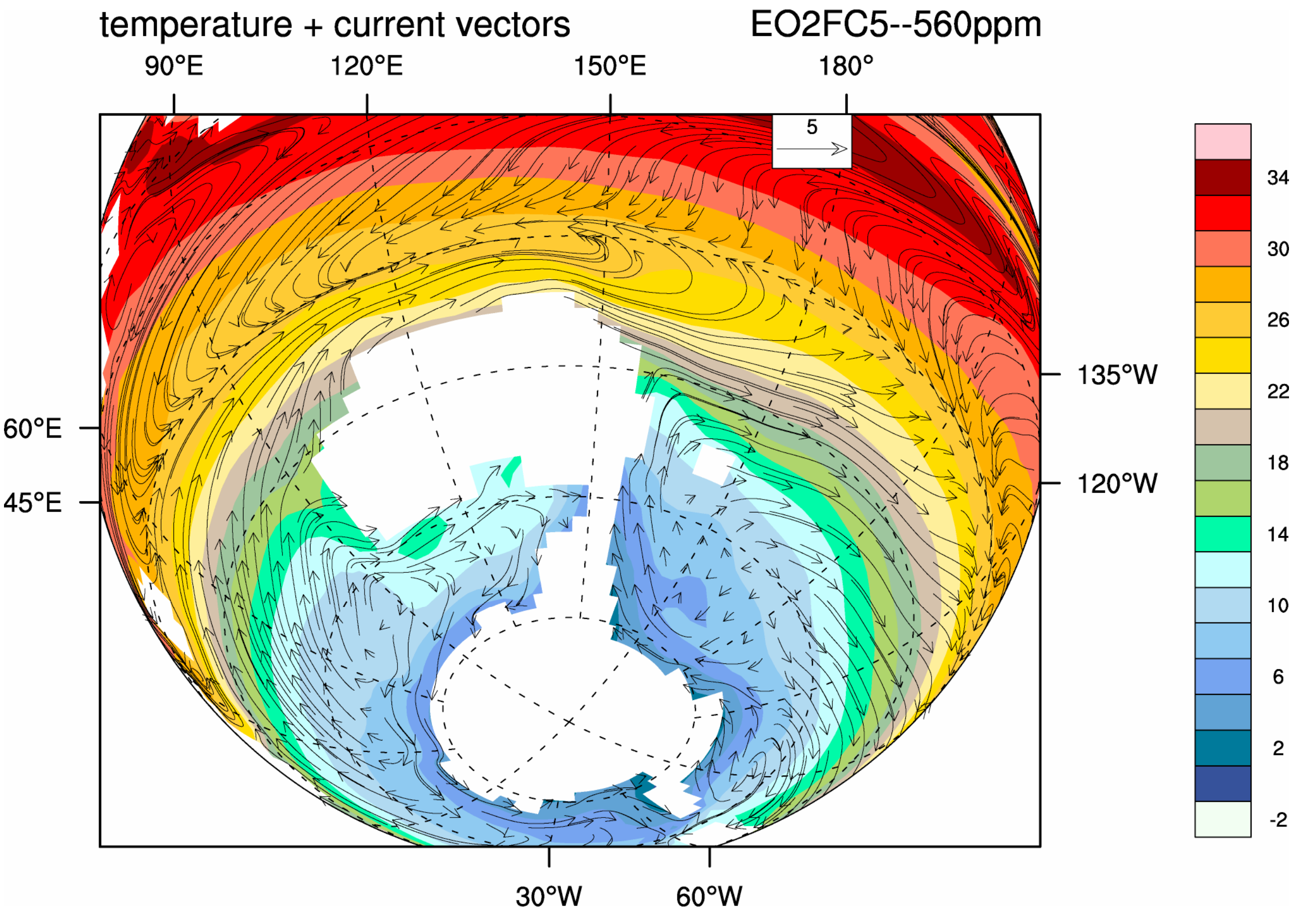

Figure 1: An example of a typical, but obviously unsatisfying visualization of NCOM fields.

2 RELATED WORK

Current computer techniques for flow visualization can be classified into geometric and texture-based approaches. The geometric approach creates flow lines such as streamlines, streaklines, pathlines, and timelines, and displays them with graphical primitives. These methods are straightforward, easy to implement, fast in rendering, and widely used to visualize flows. In particular, these methods have the advantage in linking to lab flow experiments, for example, by comparing computer-generated images with the experimental counterparts such as photos of real flows [1, 2].

A major drawback of the flow line methods is that the effectiveness of visualization heavily depends on the strategy of seeding flow lines. Often there is no obvious way to determine the initiation points of flow lines. To generate effective flow lines, significant user interaction and prior knowledge of the flow field are needed. Besides, there is an issue on the number of flow lines to be generated. If too many are generated, the visualization view is cluttered and interesting patterns are difficult to be recognized. On the other hand, if too few flow lines are generated, local details and delicate structures might be missed in the display. These problems limit the user's ability to fully explore flow structures, from the broad view to the fine scale of the flow. An example of a typical, but obviously unsatisfying visualization of NCOM fields is shown in Figure 1. Significant and eventually redundant effort is needed to pick out plotting parameters that emphasize the curliness of the flow at small scales.

To avoid the drawbacks of flow lines, the texture-based approach has been developed where image mapping is used to generate a representation of a flow with both the global view and local details [3]. This approach has been applied to visualize ocean flows [4]. Van Wijk [5] proposed a spot noise method using stretched ellipses to create 2D textures that can be mapped onto parametric surfaces. Max et al. [6] further utilized the spot noise method to visualize 3D velocity fields near contour surfaces. Cabral and Leedom [7] presented Line Integral Convolution (LIC) using 1D low-pass filter to convolve a white noise texture along the principal curve of a vector field. Various optimizations and extensions have been proposed to improve the computational efficiency and quality of LIC, including resolution-independent LIC, parallel LIC, and LIC with flow feature detection [8-14]. These LIC techniques provide rapid, informative representations of complex flows while requiring little user intervention.

Recently, Jobard et al. proposed Lagrangian-Eulerian Advection (LEA) to reduce the computational time in order to generate 2D flow visualization at interactive frame rates [15]. This algorithm produces animations with high spatio-temporal correlation. Based on the Lagrangian and Eulerian formalisms, a dense set of particles is stored as coordinates in a texture and an iteration is defined by a Lagrangian step (backward time integration of the particles) and an Eulerian step (update of the image pixel colors). Each frame depicts the instantaneous structure of the flow, whereas an animated sequence of frames reveals the motion of a dense collection of particles that are released into the flow. The LEA algorithm has also been extended to visualize unsteady flows [16]. As a case study, Grant et al. [17] applied LEA to visualize ocean flows with consideration of the vertical motion.

Image-Based Flow Visualization (IBFV) is another remarkable flow visualization technique developed recently by van Wijk [18]. The key idea of this technique is that each rendered image is the result of warping the previous image with blending of a background image. This method provides a framework to generate a wide variety of flows of objects, from moving particles, streamlines, moving textures, to topological images. Furthermore, in a combination of LEA and IBFV, Laramee et al. [19] introduced a visualization technique to directly display unsteady flows on surfaces.

To achieve fast rendering speed, hardware enhancement has been exploited such as in image blending operations and accelerations of streamline integration and texture convolution. Heidrich et al. [20] presented the first hardware acceleration of LIC. Recently, Weiskopf et al. [21] developed a hardware-based technique of LEA for 2D flow visualization. Telea and van Wijk [22] implemented hardware acceleration for IBFV by composing a volume into a series of 2D slices.

While texture-based techniques have been extensively used for 2D visualization, they are still not sufficient to directly visualize 3D vector fields. To visualize 3D vector fields, these techniques often need to slice a 3D volume in order to expose the elements of the structure in terms of the slices. However, direct 3D visualization is necessary in visualization of complex flows. For example, in atmospheric flows, the vigorous up and down drafts within a horizontally swirling column of air often cause the most damage or produce the most rain. While ocean flows tend to be more two-dimensional in parallel with the sea level, the dynamics of eddies, frontal systems, and deep convection in the ocean are inherently three-dimensional; for example, the upwelling of nutrients from below the upper 100 meters of the ocean and the resulting oceanic 'plankton bloom' are associated with primary productivity. One approach to overcome the limitations of LIC is to interactively modify the rendered representation in order to highlight the interesting flow structures [23]. A different approach is to inspect the flow field to detect and analyze the critical points [24]. A level-set method [25] has been proposed to overcome the limitations of the texture-based approach. However, this approach requires converting a 3D vector field into a scalar field, which demands large memory space and significant pre-processing time. Some LIC-based technique such as LETS [17] has been extended to handle 3D flows, but the technique only works for the case with weak vertical velocities and is not suited for general 3D cases, because velocity fields with vertical velocities comparable to the horizontal velocities will be distorted surfaces very quickly.

Related to the work reported here, techniques for direct volume rendering of 3D vector fields were investigated by Crawfis and Max [26, 27] using texture splats. Splatting [28, 29] is an efficient method and could work with hardware acceleration to determine the projected location of the voxel. Its problem is aliasing and footprint decision. EWA volume splatting [30] was developed to compensate the aliasing artifacts, but it involves intensive computation. Adaptive EWA splatting has been proposed with hardware acceleration [31].

In current visualization tools of real atmospheric and oceanic data, Vis5D and its extension D3D [32] are commonly used in practice, such as by the National Weather Service. These tools however are still subject to numerous limitations, for example, in presenting details and rendering speed. Related to this work, ParVox developed at JPL [33] is a general purpose volume rendering system that runs on massively parallel high performance platforms, and utilizes the splatting algorithm. POPTEX, a tool developed for analyzing output of ocean models, implements texture mapping utilizing hardware specific features of the SGI system. These tools are designed for large system memories and computational speeds and are not suitable to work on common PCs.

3 COMPUTATIONAL FLOW MODEL AND DATA

Various computational models have been proposed to study oceanic and atmospheric flows. One type of models works with relatively low resolutions but runs over large domains, i.e. the whole planet. These are called general circulation models (GCMs). GCMs have proven to be crucial to the understanding of past and future climate changes. Another category of models works at high resolutions and runs over limited domains with their boundary conditions specified by the output from GCMs or from measured data. These models can capture the smaller and more energetic scales of oceanic and atmospheric structures, and therefore they are more vital to weather prediction and to understanding the flow dynamics at delicate levels.

In this paper, we will consider two datasets that are generated from computational models for oceanic and atmospheric flows. The oceanic data was computed in order to better understand the dynamics of global warming, as described by Huber and Sloan [34]. The model simulated modern climate and that of a past 'globally warm' climate 50 million years ago using the fully coupled general circulation model, the Community Climate System Model (CCSM) developed at the National Center for Atmospheric Research (NCAR). CCSM version 1 (v.1) is described and validated for modern conditions in Boville and Gent [35] and references therein. The results presented here are produced with an updated version 1.4, described for modern studies in Boville et al. [36] and Blackmon et al. [37]. The atmospheric and land models are the Community Climate Model 3.6 and the Land Surface Model 1.2 at T31 (~3.75° by 3.75°) resolution. These are coupled to NCAR's Ocean Model (NCOM) and the Community Sea Ice Model. The latter models are on a stretched grid with .9° equatorial and 1.8° high latitude meridional grid spacing; zonal resolution is 3.6°. Components are connected through the Flux Coupler v. 4.0.5. The inclusion of state of the art treatments of boundary layer (KPP) and isopycnal eddy mixing (GM) in the ocean are notable features of this model [35]. CCSM also produces a response to future global warming well in the range produced by other models, with equilibrium sensitivity of ~2.1 °C for a doubling of $p\mathrm{CO}_2$. Here we present ocean circulation results from NCOM. NCOM periodically writes output for every month of simulated time containing velocity (U,V,W), tracer (temperature, salinity, ideal age), and other fields, in netcdf format.

The atmospheric data was generated in order to better understand so-called 'supercell thunderstorms' which tend to be the parent storm for tornados. We applied the Weather Research in Forecasting (WRF) model (http://www.wrf-model.org/) which is jointly supported by a number of federal agencies (NOAA, NCAR, NSF, AWRA). This model has prognostic equations for wind, convection and precipitation, cloud water variables, and many other quantities, and these are output at regular intervals in netcdf format. These fields include cloud water, rain water, ice, snow, and graupel. WRF is a fully three-dimensional, non-hydrostatic, weather forecasting model run at ~1 km resolution.

Figure 2 displays the visualization results of the oceanic dataset using flow line and LIC methods. In Figure 2(a) streamlines of the ocean flow and the land boundary surfaces are shown. Red stands for high temperatures and blue for low. In Figure 2(b), the top images are LIC representations of the oceanic flows at different depths, and the bottom images incorporate color mapping (red for high and blue for low temperatures). The areas with uniform brown colors are solid lands. The images and their combination reveal useful information of the ocean, but there is still a lack of a complete view of 3D visualization with both global and local features within one image.

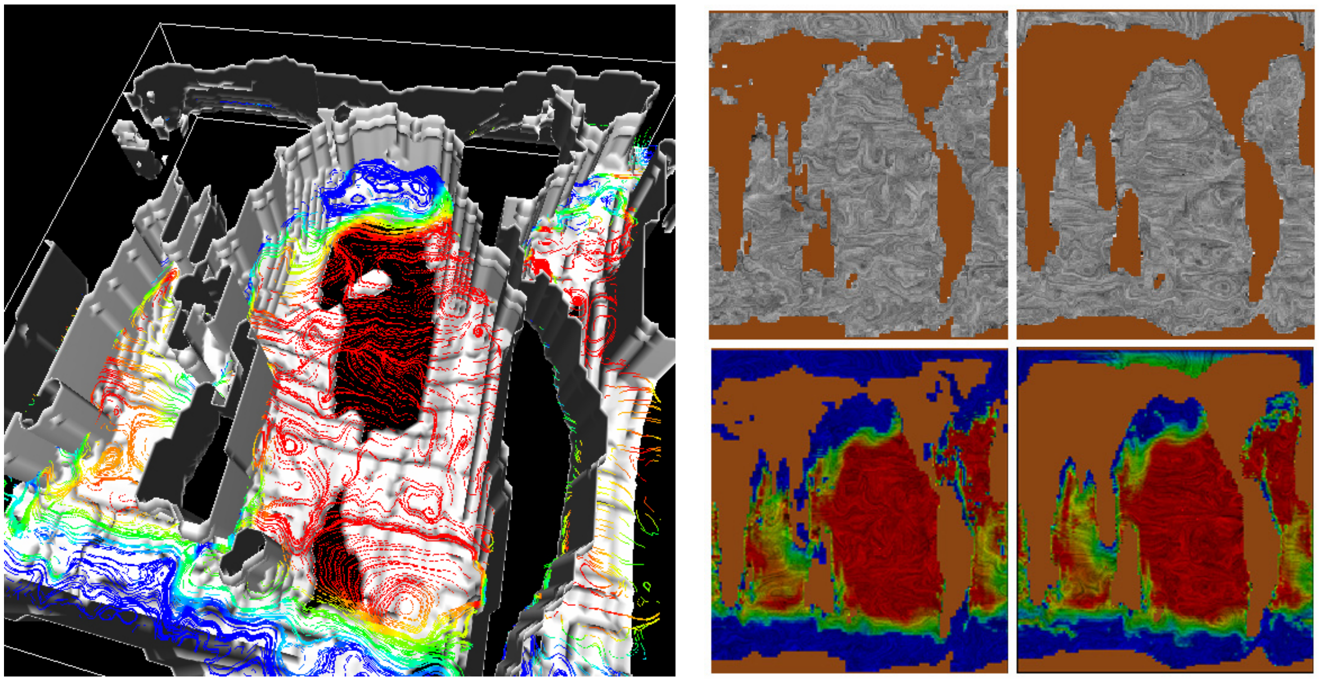

Figure 2: (a) Visualization of the oceanic flow with streamlines. Red and blue colors are for high and low temperatures. The surfaces in white color are the boundaries of lands. (b) The top two images are LIC representations of the oceanic flows at different depths. The bottom two images are LIC representations for the same depths but with color mapping (red for high and blue for low temperatures). The areas in uniform brown are solid lands.

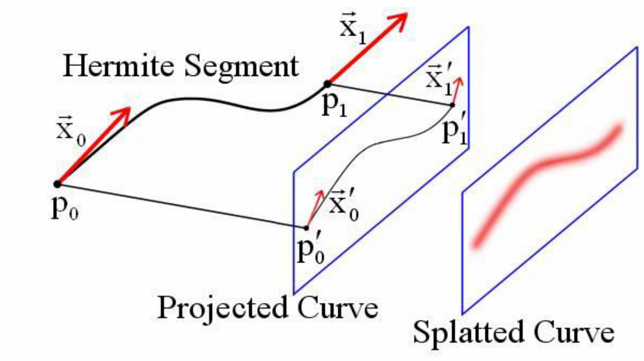

Figure 3: Projection and splatting on the image plane of a streamline segment represented using the Hermite model.

4 STREAMLINE SPLATTING

The method of streamline splatting we propose here integrates the techniques of streamline generation and volume rendering. In the first stage, this method generates segments of streamlines. In the second stage the streamline segments are projected and splatted on the image plane. The splatting is created by applying a Gaussian footprint function of each projected curve segment (Figure 3). Thus all projected streamlines have gradually varying intensities centered at the geometric projected curves, and the colors are blended together. As a result, in the rendered image, the user can see through a volume of the flow field and obtain a 3D representation view in one image.

4.1 Generation of Streamline Segments

In the first stage, we use a fourth-order Runge-Kutta ODE solver [38] to generate segments of streamlines. To ensure that the vector field is well represented by the generated streamline segments, we employ the idea proposed by Stalling and Hege [10]. Specifically, we divide the vector field into equally sized regions and assign each region a bucket. We count the times whenever a streamline intersects a region and store the total count in the corresponding bucket. After an initial set of streamlines are generated, we test each bucket. If the ratio is below the criterion, a new streamline is created from a cell in that region.

4.2 Curve Projection and Splatting

In the second stage, we projects a curve segment onto the image plane and then apply the projected 2D curve with splatting to create soft shade (Figure 3). Consider a generated streamline segment that starts at point $\mathbf{P}_0$ and terminates at $\mathbf{P}_1$, and the velocities at the two end points are $\vec{x}_0$ and $\vec{x}_1$. Let $\mathbf{P}_0'$ and $\mathbf{P}_1'$ be the projected points of $\mathbf{P}_0$ and $\mathbf{P}_1$, and let $\vec{x}_0'$ and $\vec{x}_1'$ be the projected velocities of $\vec{x}_0$ and $\vec{x}_1$. We can describe the 2D projected curve using a Hermite parametric model, which specifies a curve segment given the two end points $\mathbf{P}_0'$ and $\mathbf{P}_1'$, and the vectors $\vec{x}_0'$ and $\vec{x}_1'$ at these end points. The Hermite curve segment is then stepped along with the parameter $t \in [0,1]$ at discrete intervals of size Δt, where a Gaussian footprint [29] is applied to each interval to create the splatting. This 2D approach greatly simplifies the computation comparing the approach that traces a 3D curve segment and produces the splatting. Figure 4 shows images rendered using streamline splatting of a 3D vector field viewed from different viewing directions.

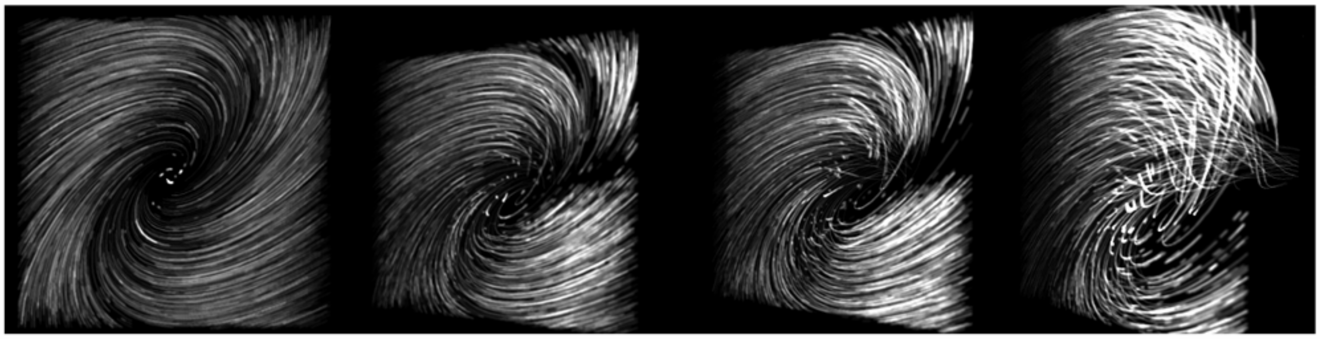

Figure 4: (a) Images rendered using streamline splatting for vector field $f(\mathrm{x,y,z}) = (y-x, z-x, z-y)$ under different viewing directions.

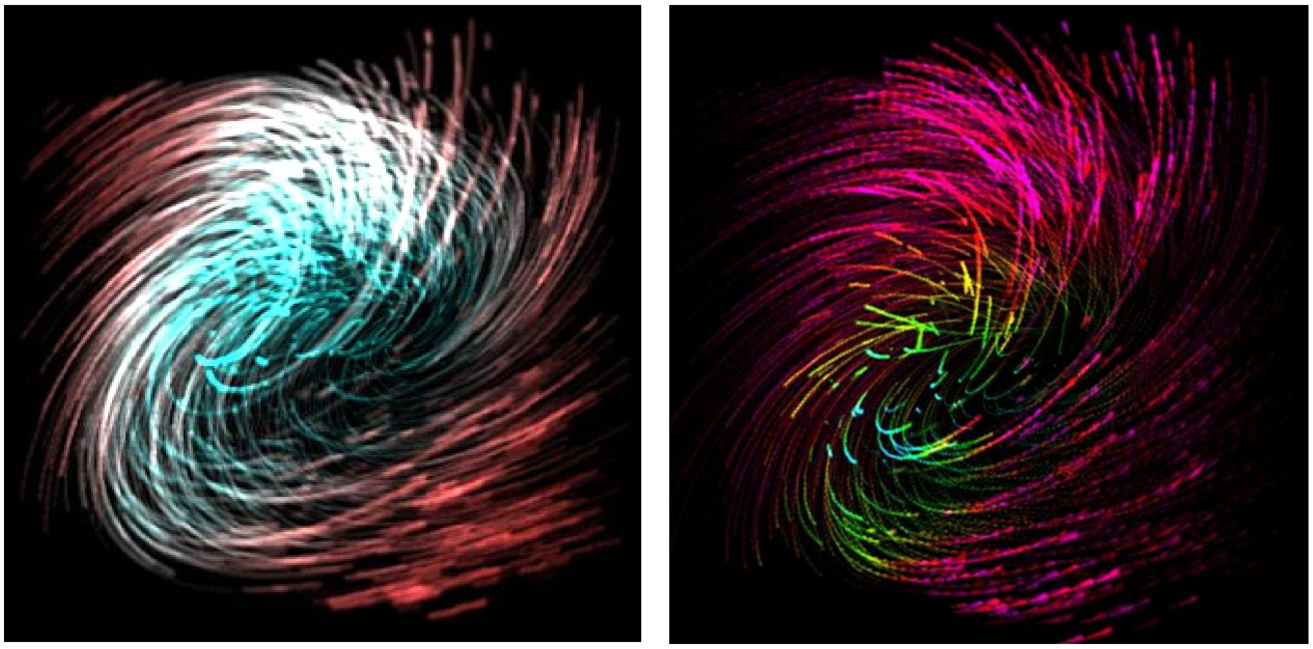
(a) (b)

Figure 5: Images rendered using streamline splatting for vector field $f(\mathrm{x},\mathrm{y},\mathrm{z}) = (y-x, z-x, z-y)$ using color mapping without (a) and with (b) the white-out control .

4.3 White-out Control

One problem of our method is the white-out effect. This effect occurs when splatted curves blend with different colors, resulting unsaturated or even white colors if the original curve colors are complementary. If complementary colors are used to convey scalar information (such as temperature) in rendered images, the color shifts and white-outs may lead to misinformation. The white-out effect can be clearly seen in Figure 4(a).

We use a simple method to prevent the white-out effect. Our method employs the spatial divisions and buckets for streamline distribution. First, we determine the bucket that a point lies in, then trace through the bucket array following the direction of the camera, and sum the total number of streamline intersections. This method gives the number of streamlines passing through the region, making it possible to limit the streamlines to be drawn through a specific part of the final image. Figure 4(b) shows the result with applying method to control the white-out effect.

4.4 Rendering Speed

One goal of this work is to achieve image creation for 3D flow visualization with fast rendering speed on common PCs. Using streamline splatting, to generate dense visualization, the rendering times are typically on the scale of seconds on regular computers. Some test results are given in Table 1. These tests use vector field $f(\mathrm{x},\mathrm{y},\mathrm{z}) = (y-x, z-x, z-y)$, with all parameters being constant except the number of streamline segments to be generated. One set uses the spatial white-out control described in Section 4.3, and the other disables it. These tests were run on an AMD Athlon XP 1800+ 1.53 GHz, 512 MB system memory, with a 128 MB NVIDIA GeForce FX 5700 Ultra graphics adapter. The average values of five trials are displayed in Table 1.

From Table 1, we can see that the rendering times with white-out control off or on are comparable. When the number of streamline segments is relatively small, the rendering with the white-out control off takes less time than with the white-out control on. But when the number of streamline segments is relatively large, the rendering with the white-out control off takes longer time. This can be explained as follows. When a small number of streamline segments are generated, with the white-out control, relatively more computational time is used on streamline projection and splatting. However, when a large number of streamline segments are generated, the computation for the white-out control procedure dominates the entire performance. Overall, however, the rendering speeds are not sufficient for interactive visualization. But with hardware support, we expect that interactive visualization using streamline splatting is possible on common PCs.

Number of Streamline	Time without White-out Control (seconds)	Time with White-out Control (seconds)
1000	0.3	0.4
2000	0.7	0.8
4000	1.3	1.4
8000	2.7	2.4
16000	5.1	3.9

Table 1: The rendering time of with and without the white-out control for different numbers of streamlines.

5 APPLICATION TO CLIMATE FLOWS

Figure 6 shows the rendered results of visualization using streamline splatting for the oceanic flow dataset described in Section 3. For both images the white-out control is enabled. Warm colors (magenta) stand for high velocities and cold colors (blue) for low velocities. The black regions are continents. The image (a) has relatively less streamline segments than the image (b).

In Figure 7, the images are rendered using streamline splatting showing atmospheric flows in the top view (top row) and in the side view (bottom row). Different columns correspond to the atmospheric flows at different time steps during a storm development. Warm colors stand for high velocities and cold colors for low velocities. A rendered image with high resolution is shown in Figure 8.

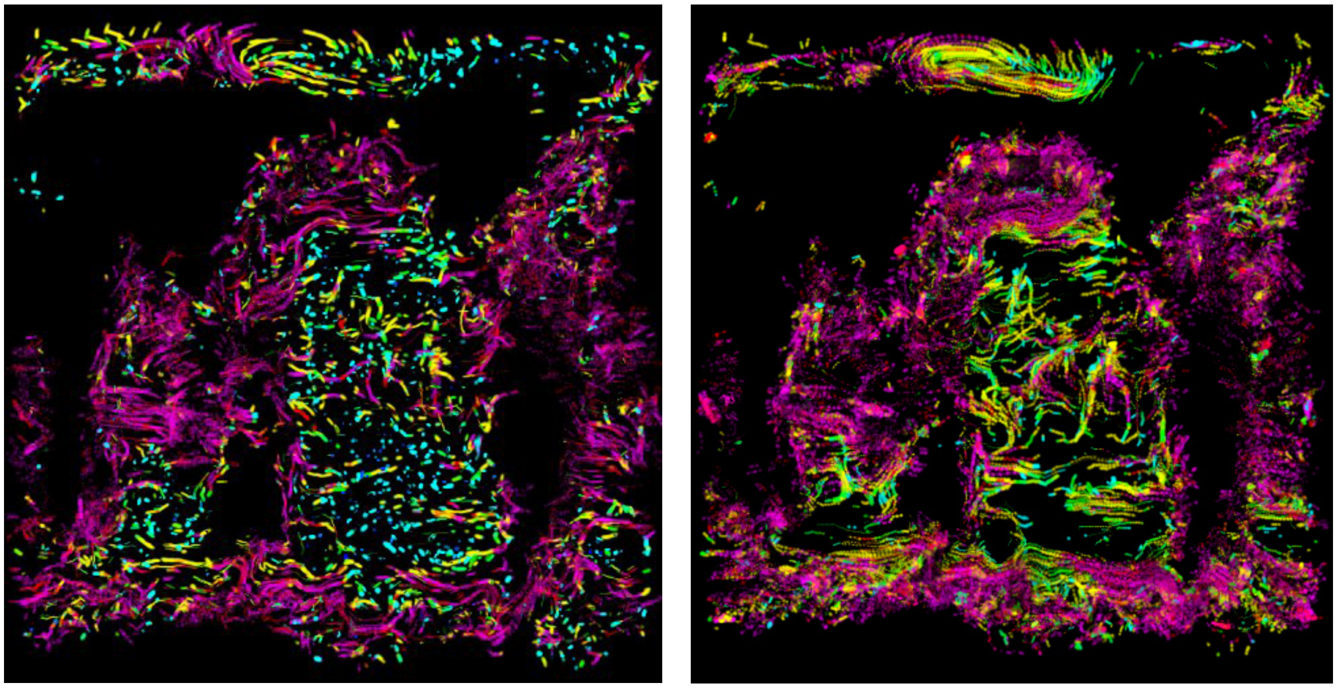
(a) (b)

Figure 6: Rendered images of top view for the oceanic flow using streamline splatting with white-out control. Warm colors stand for high velocities and cold colors for low velocities. The black regions are continents. Image (a) has relatively less streamline segments than image (b).

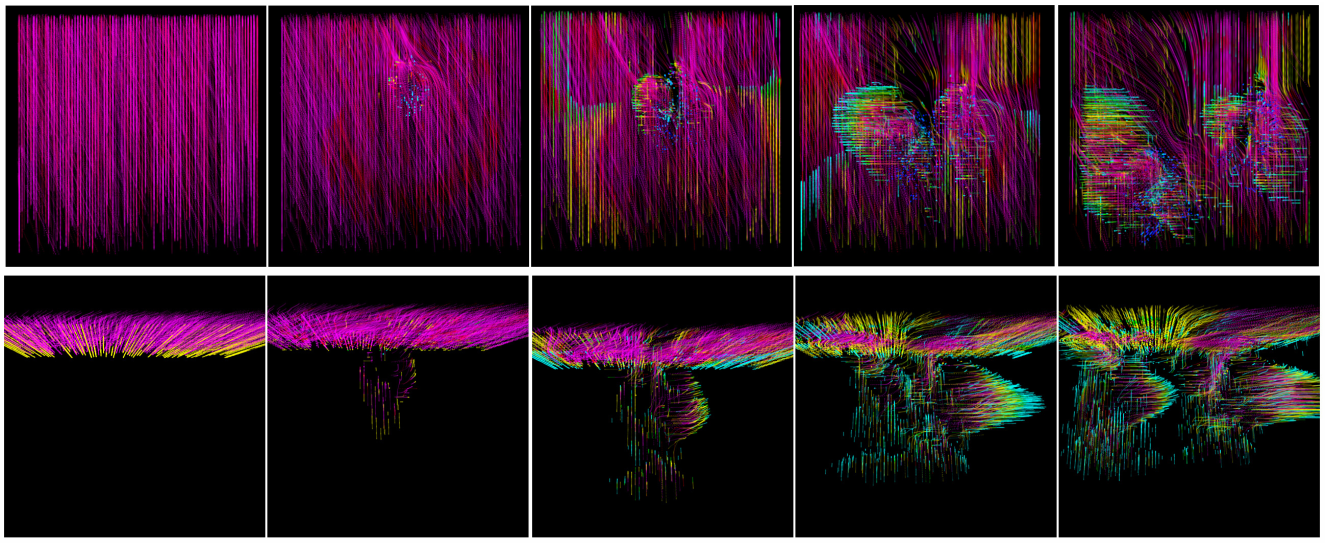

Figure 7: Images are rendered using streamline splatting for atmospheric flows for the top view (top row) and side view (bottom row). Columns correspond to the atmospheric flows at different time steps during the storm development. Warm colors stand for high velocities and cold colors for low velocities.

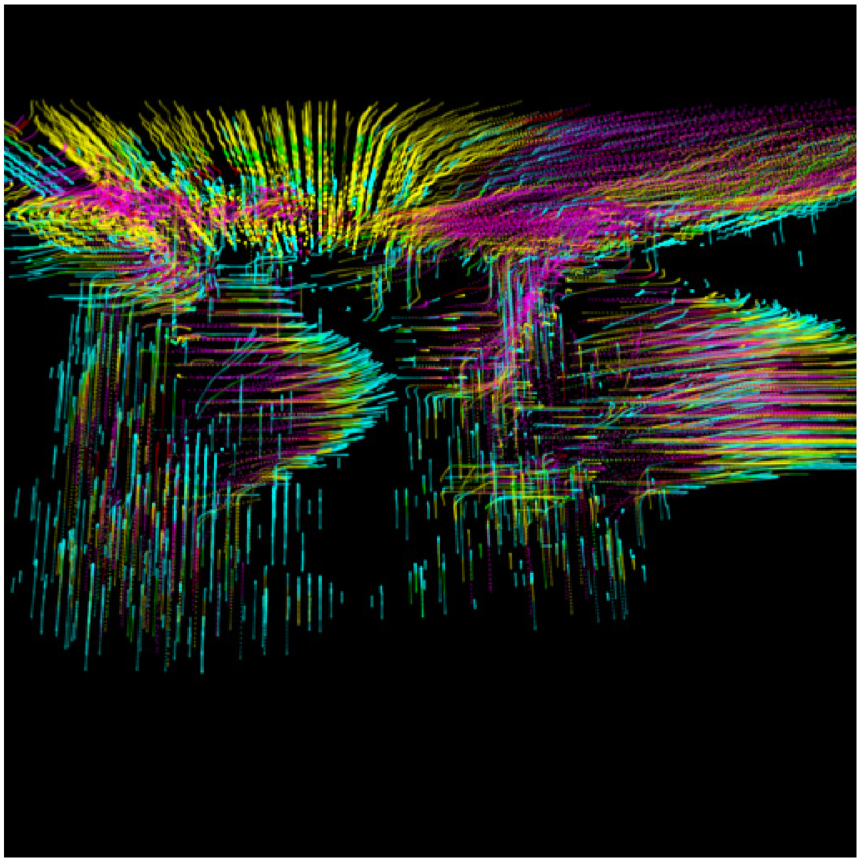

Figure 8: A high resolution image for the last image in Figure 7.

6 CONCLUSIONS

We have applied a new technique called streamline splatting to visualize 3D vector fields. This technique integrates streamline generation and splatting of volume rendering. The first stage generates segments of streamlines and the second step projects and splats the streamline segments on the image plane. The projected streamline segments can be represented using a Hermite parametric model, which simplifies the computation. Curve splatting is achieved by applying a Gaussian footprint function to the projected curve segments, and the results are blended together. Thus the user can see through a volume of the flow field and obtain a 3D representation view in one image. This work has potential to be further developed into visualization software for regular PC workstations to help researchers explore and analyze climate flows.

In streamline splatting, colors can be used to code the streamlines with some scalar information such as temperature. One problem encountered in our method is the white-out effect where the blending of splatted curves of different colors generates unsaturated or even white colors. Ee used a simple method to prevent the white-out effect. The solution employs the spatial divisions and buckets to be used for streamline distribution. First, we determine the bucket that a point lies in, then trace through the bucket array following the direction of the camera, and sum the total number of streamline intersections. This method gives the number of streamlines passing through the region, making it possible to limit the streamlines to be drawn through a specific part of the final image.

As future work, the proposed streamline splatting needs to be tested with other flow datasets. Several aspects of the method can be further improved. On white-out control, for example, one can choose the color map such that color blending avoids generating the white color (this is possible by not using complementary colors in the color map). Besides, the current streamline splatting technique has not considered the opacity and occlusion of the generated streamline segments with soft shade extensions. In the future, an opacity control along with a classification transfer function such as those used in medical volume rendering can be implemented. With this opacity control, only those streamline segments that are most close to the viewer will be visible. This will effectively remove the problem of the white-out effect.

An important goal of this research is to achieve interactive rendering of 3D flow visualization on common PCs. The current streamline splatting method is capable of providing a competitive rendering performance. On regular PCs, to generate dense visualization, the rendering times of streamline splatting are typically on the scale of seconds. Although this is still not sufficient for interactive rendering, but with further implementation optimization and hardware support, it is promising that streamline splatting is able to achieve interactive visualization on regular PCs.

REFERENCES

[1] W.-J. Yang, *Computer-Assisted Flow Visualization: Second Generation Technology*. Boca Raton, FL: CRC Press, 1994.

[2] H. Aref, R. D. Charles, and T. T. Elvins, "Scientific Visualization of Fluid Flow," in *Frontiers of Scientific Visualization*, C. A. Pickover and S. K. Tewksbury, Eds. New York, NY: John Wiley & Sons, 1994.

[3] N. Max and B. Becker, "Flow visualization using moving textures," presented at Proceedings of the ICASW/LaRC Symposium on Visualizing Time-Varying Data, 1995.

[4] A. Johannsen and R. J. Moorhead, "AGP: Ocean Model Flow Visualization," *IEEE Computer Graphics & Applications*, vol. 15, pp. 28-33, 1995.

[5] J. J. van Wijk, "Spot Noise: Texture Synthesis for Data Visualization," presented at Computer Graphics, Proc. of ACM SIGGRAPH, 1991.

[6] N. Max, R. Crawfix, and C. Grant, "Visualizing 3D Velocity Fields Near Contour Surfaces," presented at Proceedings of IEEE Visualization 1994, 1994.

[7] B. Cabral and L. Leedom, "Imaging Vector Fields Using Line Integral Convolution," presented at Computer Graphics, Proc. of ACM SIGGRAPH, 1993.

[8] H. Battke, D. Stalling, and H.-C. Hege, "Fast Line Integral Convolution for Arbitrary Surfaces in 3D," in *Visualization and Mathmatics*, 1995, pp. 181-195.

[9] L. K. Forssell and S. D. Cohen, "Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation, and Unsteady Flows," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, pp. 133-141, 1995.

[10] D. Stalling and H.-C. Hege, "Fast and Resolution Independent Line Integral Convolution," presented at Siggraph 1995 Proceedings, 1995.

[11] H.-W. Shen, Christopher R. Johnson, Kwan-Liu Ma, "Visualizing Vector Fields Using Line Integral Convolution and Dye Advection," presented at Proceedings of the 1996 Symposium on Volume Visualization, 1996.

[12] V. Interrante and C. Grosch, "Strategies for Effectively Visualizing 3D Flow with Volume LIC," presented at Proceedings of IEEE Visualization 1997, 1997.

[13] H.-W. Shen, Kwan-Liu Ma, Christopher R. Johnson, "Global and Local Vector Field Visualization Using Enhanced Line Integral Convolution," presented at Proceedings of the 1996 Symposium on Volume Visualization, 1996.

[14] H.-W. Shen and D. Kao, "UFLIC: A Line Integral Convolution Algorithm for Visualizing Unsteady Flows," presented at Proceedings of IEEE Visualization 1997, 1997.

[15] B. Jobard, G. Erlebacher, and M. Y. Hussaini, "Lagrangian Eulerian Advection for Unsteady Flow Visualization," presented at Proceedings of IEEE Visualization 2001, 2001.

[16] B. Jobard, G. Erlebancher, and M. Y. Hussaini, "Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, 2002.

[17] J. Grant, G. Erlebacher, and J. O'Brien, "Case study: visualizing ocean flow vertical motions using Lagrangian-Eulerian time surfaces," presented at IEEE VIS 2002, 2002.

[18] J. J. van Wijk, "Image Based Flow Visualization," *ACM Transactions on Graphics, Proc. of ACM SIGGRAPH 2002*, vol. 21, pp. 745-754, 2002.

[19] R. S. Laramee, R. Jobard, and H. Hauser, "Image Space Based Visualization of Unsteady Flow on Surface," presented at IEEE Visualization 2003, 2003.

[20] W. Heidrich, R. Westermann, H.-P. Seidel, and T. Ertl, "Application of pixel textures in visualization and realistic image synthesis," presented at ACM Symposium on Interactive 3D Graphics, 1999.

[21] D. Weiskopf, G. Erlebacher, M. Hopf, and T. Ertl, "Hardware-Accelerated Lagrangian-Eulerian Texture Advection for 2D Flow Visualization," presented at Proceedings of the 7th international fall workshop on Vision, Modeling, and Visualization (VMV2002), Erlangen, Germany, 2002.

[22] A. Telea and J. J. van Wijk, "3D IBFV: hardware-accelerated 3D flow visualization," presented at Proc. of IEEE Visualization 2003, 2003.

[23] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl, "Interactive Exploration of Volume Line Integral Convolution Based on 3D-Texture Mapping," presented at Proceedings of the Conference on Visualization '99, 1999.

[24] L. Hesselink and T. Delmarcelle, "Visualization of Vector and Tensor Data Sets," in *Scientific Visualization: Advanced and Challenges*, 1994, pp. 367-390.

[25] R. Westermann, C. Johnson, and T. Ertl, "A Level-Set Method for Flow Visualization," presented at Proceedings IEEE Visualization '00, 2000.

[26] R. Crawfis and N. Max, "Direct Volume Visualization of Three-Dimensional Vector Fields," presented at Proceedings of the 1992 Workshop on Volume Visualization, 1992.

[27] R. Crawfis and N. Max, "Texture Splats for 3D Scalar and Vector Field Visualization," presented at Proceedings of the 4th Conference on Visualization '93, 1993.

[28] L. Westover, "Interactive Volume Rendering," presented at Proceedings of the Chapel Hill Workshop on Volume Visualization, 1989.

[29] L. Westover, "Footprint Evaluation for Volume Rendering," presented at Proc. of ACM SIGGRAPH 1990, 1990.

[30] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "EWA Volume Splatting," presented at IEEE Proceedings of Visualization 2001, 2001.

[31] W. Chen, L. Ren, M. Zwicker, and H. Pfister, "Hardware-Accelerated Adaptive EWA Volume Splatting," presented at Proceedings of IEEE Visualization 2004, 2004.

[32] E. Szoke, U. H. Grote, P. T. McCaslin, and P. A. McDonald, "D3D: overview, update, and future plans," presented at Interactive Symposium on AWIPS, Orlando, FL, 2002.

[33] P. P. Li, S. Whitman, R. Mendoza, and J. Tsiao, "ParVox-A parallel splatting volume rendering system for distributed visualization," presented at IEEE Parallel Rendering Symposium, 1997.

[34] M. Huber and L. C. Sloan, "Heat transport, deep waters, and thermal gradients: Coupled simulation of an Eocene 'Greenhouse' climate," *Geophys. Res. Lett.*, vol. 28, pp. 3481-3484, 2001.

[35] B. A. Boville and P. R. Gent, "The NCAR Climate System Model Version One," *Journal of Climate*, vol. 11, pp. 1115-1130, 1998.

[36] B. A. Boville, J. T. Kiehl, P. J. Rasch, and F. O. Bryan, "Improvements to the NCAR CSM-1 for transient climate simulations," *Journal of Climate*, vol. 14, pp. 164-179, 2001.

[37] M. Blackmon and etal., "The Community Climate System Model," *Bulletin of the American Meteorological Society*, vol. 82, pp. 2357-2376, 2001.

[38] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, UK: Cambridge University Press, 1992.