Explorative, Immersive, and Interactive Visualization of Volume Flow Data

Zhanping Liu
Old Dominion University
Norfolk, VA
z1Liu@odu.edu

Sean Leonard
Old Dominion University
Norfolk, VA
sleon010@odu.edu

Simon Su
U. S. Army Research Laboratory
Aberdeen Proving Ground, MD
simon.m.su.civ@mail.mil

David Kao
NASA Ames Research Center
Moffett Field, CA
david.l.kao@nasa.gov

Yuzhong Shen
Old Dominion University
Norfolk, VA
yshen@odu.edu

Luis Bravo
U. S. Army Research Laboratory
Aberdeen Proving Ground, MD
luis.g.bravorobles.civ@mail.mil

ABSTRACT

While scientific visualization is an increasingly important paradigm for big data analysis, flow visualization is primarily dedicated to the depiction of tangential directions and topological features of velocity vector data that may result from modeling and simulation of oceanographic–atmospheric phenomena as well as computational fluid dynamics. This paper presents a suite of novel techniques for explorative, immersive, and interactive visualization of volume flows by exploiting the high-performance algorithms of our ActiveFLOVE (FLOw Visualization Environment) package, i.e., a collection of C++ classes developed from scratch without dependence on any third-party library/tool. In the form of seeds uniformly placed on a 3D lattice, a dynamically reconfigurable seeder is attached to and centered at the user. With one streamline spawn from each seed, a set of illuminated curves is "dragged" by the user for examining regions of interest or investigating the context within reach. Regardless of the data size, this seeder-based user-driven mechanism works at real-time frame rates on an ordinary laptop without the need for either special graphics hardware or any parallel implementation. This capability comes true not only because the exploration-oriented characteristic confines computing to only a subset of data for the local region and some context but also because an ultra-fast, highly accurate, and very robust streamline integrator serves as the horsepower. Meanwhile, flow structures can be probed by toggling on, translating (left / right / upward / downward / forward / backward), and rotating a rectangular view-perpendicular cutting plane, like a magnifying glass, on which a 2D projection of the flow is delineated by a multi-resolution representation selected on the fly among a clean layout of arrows, an aesthetic placement of evenly-spaced streamlines, and a realistic Line Integral Convolution texture. In addition, we design a pixel-based focus-guided content culler that can be translated and rotated, optionally hooked on a view-orthogonal cutting plane, to address visual clutter and view occlusion by pruning part of the scene to highlight the rest. Furthermore, we extend the proposed techniques to VR, via Unreal and HTC Vive, to enhance the look and feel of volume flows for more immersive exploration to facilitate scientific discovery.

ABOUT THE AUTHORS

Dr. Zhanping Liu is an assistant professor of the department of Computational Modeling and Simulation Engineering at Old Dominion University. He was a 2014-summer USRA-sponsored faculty visitor of the division of NASA Advanced Supercomputing (NAS) at NASA Ames Research Center and a 2013-summer FRPP-sponsored faculty visitor of the division of Mathematics and Computer Science (MCS) at Argonne National Lab during his employment as an assistant professor of the department of Computer Science at Kentucky State University (2011 ~ 2016). Much earlier, Dr. Liu was a research staff member of the Biomedical Image Analysis group of the School of Medicine at the University of Pennsylvania (2010 ~ 2011), the Scientific Computing and Visualization group of Kitware, Inc. (2008 ~ 2010), and Visualization Analysis and Imaging Lab of the High-Performance Computing Collaboratory (HPC^2, formerly the NSF-founded Engineering Research Center for Computational Field Simulation) at Mississippi State University (2001 ~ 2008) after a post-doctoral position with the Lab for Micto-CT Image Reconstruction and Visualization of the College of Medicine at the University of Iowa (2000 ~ 2001). He received the PhD degree in Computer Science from Peking University (2000) and the BS degree in applied Mathematics (specifically, Information Science) from Nankai University (1992), P. R. China. Dr. Liu's research interests remain in scientific data visualization, particularly vector (flow) visualization and parallel visualization (via multi-threading, GPU, and MPI). He has *independently* developed eight (8) data visualization packages / systems (including ActiveFLOVE: FLOw Visualization Environment) using C/C++ from scratch, without dependence on any third-party library/tool. More information about Dr. Liu's research and development is available at www.zhanpingliu.org.

Mr. Sean Leonard is an undergraduate student (2017 ~ present) of the department of Computer Science at Old Dominion University. He conducted research and development to enhance ExActiveFLOVE, i.e., an extension of Dr. Zhanping Liu's software package called ActiveFLOVE (FLOw Visualization Environment), through a one-year-long project funded by ODU-PURS (Program for Undergraduate Research and Scholarship). He was inducted into ODU-URHS (Undergraduate Research Honor Scholars) in 2018 and a 2019-summer intern at Newport News Shipbuilding. Mr. Leonard's research interests consist in scientific visualization, particularly vector (flow) data visualization and the extension to virtual reality platforms.

Dr. Simon Su is a Computer Scientist of the Computational and Information Sciences Directorate at the U.S. Army Research Laboratory as well as a member of DOD Supercomputing Resource Center (DOD-DSRC). His research efforts have focused on Dynamic Visual Computing Framework for Data-Intensive Analytics to enable both immersive and non-immersive visual analysis of heterogeneous data. Dr. Su is responsible for research and development of data visualization and 3D interaction using advanced immersive and interactive technologies. Much of his work supports the analysis and assessment of data generated by DOD-DSRC users, for which he collaborates closely with academic and government research partners. More recently, his work has evolved to focus on Big Data Immersive Analytics to support visualization and collaborative interaction of large scientific simulation. In addition, he explores Immersive Analytics to address the needs of data visualization in other areas of the Army. Dr. Su earned the PhD degree in Computer Science from University of Houston (2001).

Dr. David Kao is a researcher in the Advanced Computing Branch of the NASA Advanced Supercomputing (NAS) Division at NASA Ames Research Center. He has developed numerous collaborations (both internal and external to NASA) and created applications for scientific visualization in several disciplines. He has led several innovative software projects at NASA. Dr. Kao's research focuses on scientific visualization, numerical flow visualization, and computer graphics and is a subtopic manager for NASA's Small Business Innovation Research. He has won several NASA Group Achievement Awards, NASA Ames Honor Awards, and NASA Space Act Award for the software code Unsteady Flow Analysis Toolkit. He has served as an associate editor for the IEEE Transactions on Visualization and Computer Graphics and is a co-chair of the IS&T Visualization and Data Analysis Conference. Dr. Kao received the PhD degree in Computer Science from the Arizona State University.

Dr. Yuzhong Shen received B.S.E.E. degree from Fudan University (Shanghai, China) in 1990, the M.S.C.E. degree from Mississippi State University (Starkville, MS) in 2000, and the Ph.D. degree from the University of Delaware (Newark, DE) in 2004. He is currently a full professor of the Department of Computational Modeling and Simulation Engineering at Old Dominion University. His research interests include visualization and computer graphics, virtual reality, augmented reality, modeling and simulation, and signal and image processing.

Dr. Luis Bravo is a senior staff researcher at the Vehicle Technology Directorate of the U S Army Research Laboratory. His work involves the development of high fidelity physics based models enabling detailed investigations of complex engine processes including turbulent spray breakup, chemically reacting flows, and particle-laden turbulence. He leads several research efforts in partnership with academia, industry, and cross-service agencies that span basic and applied research in combustion, propulsion, and computational sciences. He is a Senior Member of the American Institute of Aeronautics and Astronautics (AIAA), member of American Physical Society, Combustion Institute and the American Society of Mechanical Engineers. He holds an appointment as an adjunct Professor of Aerospace Engineering at the University of Cincinnati as well as Chair of Modeling & Simulation at the Propulsion and Power System Alliance fostering collaborations within government agencies. He is the recipient of multiple prestigious awards including the HPCMP Frontier award employing exceptional supercomputing resources towards the development of next generation predictive models for turbulent liquid sprays and dispersed multi-phase flows. Dr. Bravo received his PhD degree in Mechanical Engineering from the University of Maryland, College Park in 2013. Subsequently, he worked at ARL first as a post-doctoral fellow and later as a permanent research staff member. He has authored over 75+ research articles, delivered 20+ talks, including 2 plenary talks, and 1 patent.

Explorative, Immersive, and Interactive Visualization of Volume Flow Data

Zhanping Liu
Old Dominion University
Norfolk, VA
z1Liu@odu.edu

Sean Leonard
Old Dominion University
Norfolk, VA
sleon010@odu.edu

Simon Su
U. S. Army Research Laboratory
Aberdeen Proving Ground, MD
simon.m.su.civ@mail.mil

David Kao
NASA Ames Research Center
Moffett Field, CA
david.l.kao@nasa.gov

Yuzhong Shen
Old Dominion University
Norfolk, VA
yshen@odu.edu

Luis Bravo
U. S. Army Research Laboratory
Aberdeen Proving Ground, MD
luis.g.bravorobles.civ@mail.mil

INTRODUCTION

As advanced physics-based modeling and high-fidelity continuous simulation give rise to massive amounts of data, complex patterns and intricate features may not *all the time* be properly extracted or effectively presented through non-visual computing paradigms like data mining and machine learning. *Scientific visualization* (SciVis) is an intuitive methodology for improved analysis of data “mounted” on a 2D/3D grid of discrete sample points at which one or multiple physical variables are defined. Stemming from computer graphics, drawing on image processing, and empowered by high-performance computing, SciVis is an integral stage of the modeling-simulation-visualization-analysis pipeline, facilitating our comprehension and interpretation of the spatiotemporal dynamics embedded in a data field. While scalar data visualization shows the geometric structure or spatial distribution of a non-directional property (e.g., pressure, temperature, humidity, density, salinity, and precipitation), *flow visualization* (Laramee et al., 2004; McLoughlin et al., 2009) is mainly focused on the delineation of directional information out of velocity vector data (e.g., ocean currents, wind flows, electromagnetic fields, and incompressible fluids). Flow visualization has seen widespread applications and demonstrated tremendous capabilities in analyzing vector data arising from computational fluid dynamics (CFD) and oceanographic-atmospheric simulations.

Directional information is encoded by two or three intrinsically coupled component values at *each* sample point of a data field. A valid representation requires that an anisotropic entity of enough length, constituted by either a geometric primitive (e.g., an arrow or a curve made up of line segments) or an array of consecutive texture elements (i.e., *texels*: fragments equivalent to pixels in 2D; volumetric elements or *voxels* in 3D) with similar intensity values, be employed to convey the local tangential direction at an individual point, or with more continuity, an integral path of movement (Liu et al., 2012). Exhibiting such spatial coherence along the flow direction (or *tangential coherence*) imposes an extra demand on the limited 2D screen real-estate that already struggles with content complexity. By itself, the representation of tangential coherence adds to visual clutter, view occlusion, and depth ambiguity in 3D settings. In fact, this situation is further compounded by placing many anisotropic entities in the domain to provide sufficient coverage so as to reveal salient patterns and capture elusive features.

There has been significant research in visualizing flows ranging from steady to unsteady and from 2D to surface and further to volume, though steady volume flow visualization is still an open problem due to the three aforementioned issues, i.e., visual clutter, view occlusion, and poor depth cueing, among others. The latter two indicate that 3D texture-based techniques (Laramee et al., 2004) such as the volume extension of *Line Integral Convolution* (LIC, Figure 1.*a*, Cabral & Leedom, 1993) or VLIC, are not good at dealing with volume flows because the dense representation, despite an advantage for 2D flows, turns into a drawback for the 3D case. To exacerbate these two issues, transfer functions designed via painstaking efforts for Direct Volume Rendering (DVR, Levoy, 1988), are awkward in “combing” flow streaks out of a 3D texture, yielding either a vague cloud (Shen et al., 1996) or short thick stream “tubes” (Interrante & Grosch, 1997). In addition, the voxelization of integral curves, performed for the convolution process, introduces errors to flow streaks and hence incurs jagginess to the resulting stream “tubes”. Furthermore, both 3D LIC and DVR are extremely compute-intensive, preventing VLIC from interactive visualization of even a small volume flow unless graphics hardware acceleration or parallel computing is utilized.

As a geometry-based technique (McLoughlin et al., 2009), *streamline*s (Figure 1.*b*, Liu et al., 2006; Han et al., 2019) are straightforward, effective, and efficient for visualizing large steady volume flows. They enable fast generation of a 1.5-dimensional representation with intuitive accurate depiction of the flow direction. Originating from a *seed* position (where a particle is released), a streamline refers to a field line (i.e., the trajectory of the particle) that is point-wise tangent to velocity vectors. Irrespective of any particle (analogous to a car), a streamline denotes a permanent trace (analogous to a road), among numerous others, in a time-independent vector field. This continuous curve is constructed by using small to tiny line segments to connect a sequence of sample points which are some consecutive "snapshots" of the particle in motion driven by the flow. These sample points are obtained via numerical integration from the seed step by step. In essence, numerical integration (i.e., a simulation procedure) solves an ordinary differential equation (i.e., a form of physics-based modeling) that governs the *step-by-step movement* (i.e., *advection*) of the particle. Also known as an integral curve, a streamline itself does not provide information in the direction perpendicular to the flow (i.e., *the orthogonal direction*). This "sparse" coverage, with 1.5-dimensional continuity (but a 1D manifold), offers a "see-through" effect in a 3D scene with many streamlines and hence helps alleviate view occlusion issues that tend to be caused by other geometry-based techniques, e.g., stream surfaces, stream ribbons, and stream tubes.

We believe that an interactive, immersive, and explorative framework is well suited for visualization of large steady volume flows. **(1) Interactive**: High computational performance needs to be prioritized to attain interactive to real-time frame rates even on an ordinary desktop PC with a single CPU but without the need for parallel implementation (by either GPU or MPI, Liu, 2019), regardless of the data size. This philosophy is aligned with a famous saying, "interactivity is the key to visualization". **(2) Immersive (engaging)**: High-quality delineation of tangential flow directions is instrumental to visual perception and mental reconstruction of the overall pattern, salient structures, and intricate features (Liu et al., 2012). An ideal case is that the user is deeply engaged in computer-generated virtual yet insightful visually appealing display of the flow (Liu et al., 2006) during a sense-making process even on regular PC platforms. Certainly, realistic effects and immersive experiences may be further augmented in VR environments. **(3) Explorative**: By adjusting visualization parameters, performing operations on the data, and manipulating the view, user interaction is crucial for fruitful flow analysis. Given a large unknown dataset and the limited 2D screen real-estate, user exploration in 3D space is indispensable for identifying and understanding complex structures and elusive features by minimizing view occlusion, visual clutter, and depth ambiguity. Explorative visualization also enables feature-oriented retrieval and ROI (Region of Interest) -based examination, confining the computational cost and memory footprint that both would otherwise be extremely aggravated by the entirety of big flow data.

Motivated by these observations, this paper presents our recent work on explorative, immersive, and interactive visualization of steady volume flow data. These three characteristics are made possible by a high-performance engine called *ActiveFLOVE* (FLOw Visualization Environment, Liu & Moorhead, 2016) that we have developed and maintained in the past. A set of "building blocks" of ActiveFLOVE, i.e., fundamental algorithms of flow visualization in the form of C++ classes, allows us to make a variety of combinations to bring forth new methods and add cutting-edge functionalities such that a significant extension of ActiveFLOVE, or *ExActiveFLOVE* for short to denote a suite of contributions of this paper, can be achieved. Specifically, a 3D rectangular uniform grid is adopted to mimic a seeder, i.e., a "machine" for "planting" many seeds simultaneously, that can be dynamically re-configured in the position, capacity, and resolution. With the seeder centered at the user all the time and with one 3D streamline produced rapidly and accurately from each seed point, a bunch of integral curves is then "dragged" by the user and rendered with a special illumination model for explorative visualization of the flow at real-time frame rates. In addition, we devise a rectangular cutting plane that is toggled on/off, translated (left / right / upward / downward / forward / backward), and rotated (around the vertical axis) within the volume on the fly but always perpendicular to the view direction for deriving a 2D vector field, to which arrows, automated placement of evenly-spaced streamlines, and LIC can alternatively be applied to show various profiles of the flow, emulating a magic lens. Furthermore, we design an interactive content-culling widget, which, possibly fastened on a view-perpendicular cutting plane, removes part of the graphical elements to avoid visual clutter and view occlusion but to highlight an ROI. With the novel capabilities mentioned above for delivering interactive immersive explorative visualization of volume flows on traditional PC platforms, ExActiveFLOVE also supports an extension to VR devices such as HTC Vive by means of Unreal, heightening the illusion as if the user were positioned and navigating in the 3D data.

The remainder of this paper is organized as follows. Section 2 begins with a concise survey of the literature closely related to our work, followed by a treatise of such important components of our ActiveFLOVE package that are exploited in ExActiveFLOVE. In section 3, we present ExActiveFLOVE, with details involving seeder-based user-driven placement of streamlines, extraction and visualization of the 2D projection of the volume flow on a view-orthogonal cutting plane, a content culler for tackling clutter, occlusion, and depth issues, and VR extension. Results and discussions are provided in section 4 to demonstrate the effectiveness and strengths of ExActiveFLOVE for visualizing steady volume flows. Section 5 concludes this paper with a brief summary and outlook on future work.

PREVIOUS WORK

Two well-known literature reviews (McLoughlin et al., 2009; Laramee et al., 2004) on geometry-based methods (e.g., arrows, streamlines, pathlines, streak lines, stream ribbons, stream tubes, and stream surfaces) and texture-based approaches (e.g., LIC and UFLIC), respectively, are very comprehensive. Thus, this section is focused on flow visualization algorithms that are related the most to and directly employed in our work (ExActiveFLOVE), i.e., streamlines and LIC. Also introduced is ActiveFLOVE on which ExActiveFLOVE is built.

Related Algorithms

As an integral method for the steady-state case, streamlines (Liu et al., 2006; Han et al., 2019) are probably the most important in flow visualization. Streamlines pave the way for many other algorithms, including stream ribbons, stream tubes, stream polygons, stream surfaces, and stream volumes. If the advection step size, a measure of pseudo time for calculating streamlines, is defined as a real time interval, numerical integration applies to unsteady flows to yield pathlines (i.e., particle traces) (Nguyen et al., 2019) from which to implement streak lines (Liu et al., 2003) and streak surfaces. It is worth emphasizing that streamlines, generated in continuous space (Stalling & Hege, 1995) or discretized / approximated by pixels / voxels (Cabral & Leedom, 1993; Shen et al., 1996), underlie many texture-based methods. Although numerical accuracy with respect to streamline integration was analyzed in depth three decades ago, it has not gained sufficient attention. In practice, fixed step sizes are utilized by a plethora of streamline-based algorithms nowadays. In this way, a relatively large step size tends to overshoot a flow trace and fails to capture high-curvature structures. On the other hand, a small and even tiny step size may still miss features in heavily turbulent areas, whereas a huge yet unnecessary amount of computational time is consumed in laminar parts. As opposed to this rigid strategy, an adaptive scheme dynamically adjusts the size based on an estimation of the error that the current step of integration introduces. It is intended to achieve both fast and accurate streamline generation. A simple measure takes the angle between the previous line segment and the new line segment being created by the current step, which is then compared against a designated range of angles (e.g., [5°, 10°]). The step size is enlarged and shrunk if the angle underflows and overflows, respectively (Liu et al., 2003). With an adaptive step size determined for the current advection, there are a couple of options for integrating the streamline one hop ahead to obtain the next sample point along the curve, e.g., Euler integrator, mid-point integrator, 4th-order Runge-Kutta Integrator (or RK4, Liu et al., 2006), and even RK5 in order of increasing accuracy. Except for the first method, the velocity vector at each of several sample points needs to be evaluated to constitute a stage. A single step of streamline integration comprises two, four, and five stages in the latter three methods, respectively, which are hence called *multi-stage integrators*. The choice of a step size and that of an integrator in combination have an overwhelming influence on the numerical accuracy and computational speed of streamline generation.

Governed by physical dynamics, a flow usually converges in some areas and diverges in others, exhibiting topological patterns revolving around critical points. A simplistic selection of seeds, either randomly or at a set of grid points, often results in an incomplete representation, a cluttered delineation, or both. Thus, appropriate seed selection and even further control over the growth of each streamline are crucial for informative elegant distribution of streamlines. In fact, automated layout of streamlines is a hot research topic. Some efforts are oriented towards highlighting topological features, with primary seeds placed around critical points by means of templates before secondary seeds are planted elsewhere for streamlines to fill remaining voids. Most algorithms are instead dedicated to Evenly-Spaced Streamline (ESS) placement, of which one solution resorts to image processing to fulfill implicit computationally expensive density control (Turk & Banks, 1996). The other, as used by the majority, turns to geometry-based inter-point distance check to approximate inter-streamline distance evaluation (Jobard & Lefer, 1997) in a way to conduct explicit relatively fast density control. Our ADVanced ESS algorithm (ADVESS, Figure 1.*c*, Liu et al., 2006) adopts an RK4 integrator with Adaptive Step Size and Error Control (RK4-ASSEC) to attain highly accurate and ultra-fast generation of streamlines and makes use of cubic Hermite polynomial interpolation to produce uniform samples along each streamline in support of inter-point distance control. ADVESS remains the fastest algorithm for high-quality ESS layout, up to 10 times faster than others. In addition, it is the only ESS algorithm that offers rapid robust loop detection to avoid any clutter from tightly spiraling streamlines.

Although 3D texture-based methods (Shen et al., 1996) have obvious and severe drawbacks in *direct* visualization of volumetric flows (Section 1), their 2D counterparts, particularly 2D LIC (or LIC hereinafter unless otherwise noted, Cabral & Leedom, 1993), possess advantages in visualizing the 2D profile of a volume flow projected on a cutting plane. Inspired by and adapted from image processing techniques, LIC applies a low-pass filter (e.g., one based on a box kernel) to a white noise texture (or an especially designed noise pattern) to convolve an array of pixels along a 1.5-dimensional curve instead of those within a rectangular neighborhood. The basic idea is to expose the close correlation between the samples of each streamline, in the form of the associated consecutive pixels carrying similar intensity values, through a smearing operation on a dense aggregation of massless particles. This image synthesis procedure emulates what happens when a rectangular area of fine sands is blown by a gust of strong wind. In more detail, from the center of each pixel (the *target*) of the output LIC image, a seed is released to integrate a streamline in both negative and positive directions, but with the same length wherever possible. Then, the correlated pixels (as the contributors) are located for the target pixel (as the receiver). Next, white noise is indexed to get the texture values for these contributing pixels. Finally, convolution is performed on these values to calculate a normalized weighted sum as the value of the target pixel. This pipeline is repeated for each and every pixel to produce the entire LIC image. The strengths of LIC consist in the high-resolution, dense, realistic representation of a planar flow.

ActiveFLOVE

There have been some open-source systems or packages with more or less support for flow visualization. Analogous to the Complex Instruction Set Computer (CISC) architecture, the cross-platform general-purpose design leads to an excessive scope (e.g., with hundreds of data file readers), a fat software framework, heavy dependencies (on many third-party libraries/packages), cumbersome user interface, considerably outdated algorithms, and low computational performance (even in parallel settings). These widely known caveats hinder such colossal packages from delivering novel robust fast accurate flow visualization to meet emerging and oftentimes even routine needs. Thus as part of our previous work, we developed a high-performance toolkit called FLOw Visualization Environment (ActiveFLOVE, Liu & Moorhead, 2016), using C/C++ from scratch, that encompasses a collection of effective, efficient, high-fidelity, robust, and versatile yet compact visualization components as well as basic utility modules. Inspired by the Reduced Instruction Computer Set (RISC) architecture and the smart OpenGL graphics engine, ActiveFLOVE is devoted to such fundamental elements of flow visualization that are indeed necessary for constructing geometry-based methods (e.g., streamlines, pathlines, streak lines, and stream surfaces) and texture-based approaches (e.g., LIC, UFLIC, and AUFLIC, Liu & Moorhead, 2005), leaving data format, image format, realistic rendering, user interface, and other peripheral items (e.g., MPI/GPU/multithreading parallelization, wrapper access, plugin deliverables, standalone systems, and *VR extension*) for higher levels of work in the stack.

ActiveFLOVE is characterized by a self-contained philosophy (without dependence on any third-party library/package), a thin framework, a shallow hierarchy, highly accurate flow representation (range of errors: $10^{-6} \sim 10^{-5}$), ultra-fast flow advection (via the RK4-ASSEC streamline integrator, Liu et al., 2006, capable of generating 7,800 streamlines with 6.1 million points per second for a wind volume flow on a Cartesian grid and 1,600 streamlines with 1.58 million points per second for the blunt-fin volume flow on a curvilinear grid, both using a single ordinary CPU), very robust implementation, and great ease of use. In fact, RK4-ASSEC is algorithmically the same as the integrator of FastLIC (Stalling & Hege, 1995), and to the best of our knowledge, these two are arguably the most accurate and the fastest in the literature. The outstanding computational performance, with interactive to real-time frame rates (tens of frames per second) on ordinary laptops without the requirement of parallel support (MPI and GPU), allows ActiveFLOVE to serve as a solid engine for explorative visualization of large flow data defined on Cartesian, curvilinear, and unstructured grids. With a pool of primitive "building blocks" that can be configured and assembled in many ways, ActiveFLOVE facilitates the design of new algorithms and addition of composite functionalities as our recent work, i.e., *Ex*ActiveFLOVE, demonstrates in the next section.

EXACTIVEFLOVE (EXTENSION OF ACTIVEFLOVE)

In this section, we present our work on an ***ex***tension of ActiveFLOVE, namely ExActiveFLOVE, for explorative immersive interactive visualization of volume flow data. Featured in ExActiveFLOVE are real-time seeder-based user-driven placement of 3D streamlines, a dynamically reconfigurable and maneuverable view-perpendicular cutting plane on which arrows, ESS, and LIC are employed to visualize a 2D profile of the volume flow, an interactive culler for filtering contents of the scene, and a combination of Unreal with HTC Vive for VR support.

User-Driven Placement of 3D Streamlines

Most work in volume flow visualization, including the use of streamlines, has been limited to outside-in views of the overall pattern and / or manual tedious rigid seeding schemes. These strategies usually cause either a coarse depiction or a cluttered image, particularly for complex structures. The high computational speed of the RK4-ASSEC integrator (Section 2.1) makes it possible for us to implement our advocacy of explorative visualization of volume flows (Section 1) by devising a seeding template which is controlled by the user at great ease for immersive navigation in the 3D domain *even on traditional PC platforms*. Bringing the user inside the data alleviates view occlusion, enables in-depth examination, and helps identify local features. Essentially, this solution amounts to user-guided flow visualization, which in turn implies that a user-centric seeding device is a straightforward choice. While the user intends to investigate an ROI around a group of seeds, the associated streamlines usually extend beyond the ROI and run into the contextual area. In this way, the entire view provides focus+context visualization of the flow. Meanwhile, streamline integration may be terminated if the curve length exceeds a threshold since further advection might just add cluttering to the view. This option makes user-guided explorative visualization well suited for real-time analysis of big flow data in that at any time or in each frame, only a (small) subset is actually accessed to show an ROI and the context within reach, with the rest ignored. As the user navigates in the volume, the ROI and the context both change accordingly. With the computational cost and memory footprint under necessary and feasible control, this approach unleashes the strengths of streamlines in visualizing volume flows.

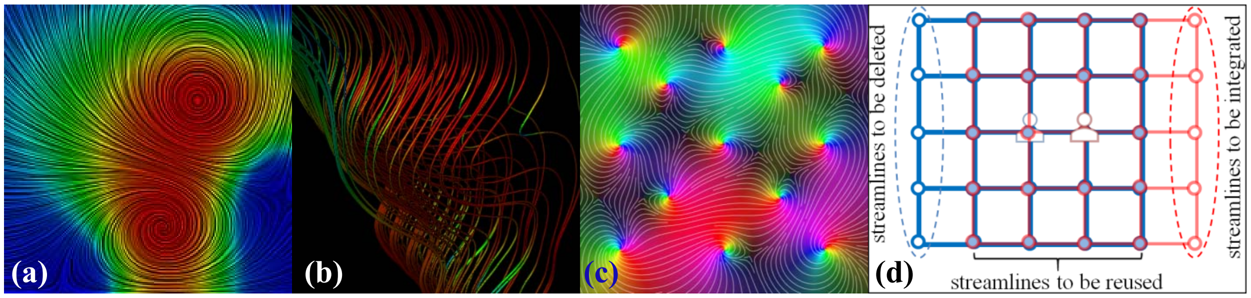

Figure 1. (a) 2D Line Integral Convolution (LIC). (b) 3D illuminated streamlines. (c) *Ad*vanced evenly-spaced streamlines (ADVESS). (d) 2D profile of a 3D seeder moving from an old position (*blue*, for previous frames) to a new position (*red*, for the current frame), with most streamlines reused and others integrated from none.

To fulfill a user-centric seeding mechanism, we design a “machine” called *seeder* which maintains a 3D rectangular lattice of seed emitters evenly spaced along each dimension, with the central emitter attached all the time to the “driver” (i.e., the user). This uniform distribution of seeds, with each dimension exactly aligned with an axis of the spatial (global) coordinate system, can easily reflect convergence and divergence behaviors of the flow from which visual perception comes into play to support feature recognition and pattern reconstruction. Based on this observation, this 3D seeder exhibits advantages over a seeding plane (2D), a seeding curve (1.5D), and a seeding line (1D, i.e., a rake line) in capturing these two kinds of flow dynamics in the *volume*. Likewise, the cuboid shape outperforms other 3D geometric objects (e.g., sphere). There are three parameters that the user can adjust at run time to manipulate the seeder: *position*, *capacity*, and *resolution*, of which the first refers to the 3D location of the central seed emitter. The capacity denotes the number of emitters, i.e., the number of seeds that are planted simultaneously. An odd integer n (≥ 3), common to the three dimensions, is adopted to arrange emitters along each dimension and therefore there are n^3 emitters in total. The resolution specifies the interval (a floating-point value) between two neighboring emitters. By default, a single interval is employed in the three dimensions to manage a set of geometrically isotropic seeds. Wherever necessary, the three dimensions may differ in the interval, resulting in a geometrically anisotropic distribution of n^3 seeds. This configuration is good at accommodating some special volumes, e.g., an ocean or wind flow field of which the vertical layers are much fewer than the rows and columns.

Wherever the user virtually goes within the flow domain by 3D translation operations (left / right / up / down / forward / backward), the position of the central emitter is updated in a way to move the entire rigid-body seeder.

Then, one seed is released from each emitter for advecting a streamline in both positive and negative directions. This navigation mode produces a look and feel as if the user were "dragging" a bundle of streamlines to comb through a dense volume of flow. From the implementation perspective, the linkage between the user and the seeder is not as "hard" as theoretically described above or as perceptually experienced. Instead, it works in a deferred manner. In reality, the seeder position remains unchanged until the user movement in any dimension reaches (or exceeds by a little bit) the length designated by the seeder resolution. In other words, most streamlines can be simply reused from frame to frame, which not only reduces the amount of streamline integration but also, and more importantly, maximizes spatial coherence as visually conveyed by the distribution of streamlines. Figure 1.*d* shows two snapshots of a 2D profile (e.g., the *X*-*Y* projection) of a 3D seeder, with the blue for the old position taken in a sequence of previous frames and with the red for the new position that is being utilized in the current frame. Without loss of generality, the seeder needs to be shifted by one cell of the lattice to the *right* in this scenario. The two snapshots share many seeds at the lattice points of the seeder such that only a small subset of the existing streamlines needs to be replaced by integrating entirely new ones, whereas others are just reused for the common seeds. This scheme also applies to left/upward/downward/forward/backward translations. Despite the intermittency of seeder movement, an illusion holds as if the seeder were *continuously* driven by the user. To increase depth cueing and enhance visual effects, we use a special graphics model called co-dimensional illumination to render 3D streamlines in the volume.

In fact, the primary goal of intermittent seeder movement is to introduce spatial coherence between frames. The rationale is that considerable changes in the contents of a scene would distract and even disorient the user such that the user could not engage in flow exploration. The minimal update on the streamlines that takes place only for the seeds emitted on a boundary slice of the seeder, and even none at all if the user movement has not yet been accumulated enough to trigger an actual alteration to the seeder position, both contribute well to maintaining strong spatial coherence. On the other hand, the high computational performance of the RK4-ASSEC integrator itself (as reported in Section 2.2) is able to achieve real-time frame rates for generating hundreds of streamlines completely from none. Our strategy is to reuse as many streamlines as possible and then reserve the power of RK4-ASSEC for the user to instead expand the capacity of the seeder. This consideration is necessary because the user may choose to enlarge the seeder capacity but decrease the resolution on the fly for a more detailed examination once an ROI has been located. If the local region shows a laminar flow pattern after the user turns around in search of features, the position and resolution of the seeder can be kept while the capacity is increased to broaden the coverage of streamlines. This operation allows the user to briefly detect potential features in the distance for informed navigation. In addition, this placement of streamlines, possibly spanning across the entirety of two dimensions (e.g., *X*-*Y*, *X*-*Z*, *Y*-*Z* planes), can be displayed in another outside-in view to enrich our understanding of the flow pattern.

Interactive View-Aware Cutting Planes

Depth ambiguity, visual clutter, and view occlusion are three major obstacles to effective volume visualization, regardless of the data being scalar or vector. Cutting planes may help mitigate these problems to some extent by showing 2D patterns on certain slices embedded in the volume. Cutting planes expose 2D samples of the volume from multiple perspectives and, if arranged appropriately and equipped with smart user interaction (e.g., animation of a stack of evenly-spaced parallel image slices), lend themselves to cognitive reconstruction of 3D structures. To accompany seeder-based user-driven placement of 3D streamlines, a cutting plane orthogonal to the view direction can augment depth cueing by providing a reference, against which graphical elements in the scene are visually separated into two parts. Even curve segments in front of the cutting plane can further be distinguished more easily in the depth, especially if the cutting plane is translated back and forth. Both of these two effects may cognitively tackle the visual cluttering problem. Apart from the role of a reference of depth, a cutting plane positioned sufficiently near the user can highlight a 2D profile of the volume flow. In front of this 2D image is only a small part of 3D streamlines, which conquers view occlusion. After all, the user can look through these curves, a geometry-based sparse representation, to inspect the flow visualized on the cutting plane. In some sense, the cutting-plane visualization and 3D streamlines take each other as the context, improving our understanding of the volume flow.

Visualization of a volume flow projected on view-perpendicular cutting planes is a constituent functionality of ExActiveFLOVE. As the user navigates (via left / right / upward / downward / forward / backward translation operations) and turns around (the vertical axis by default) in the 3D domain based on the global coordinate system *X*-*Y*-*Z*, the view point and view direction both dynamically change. Once the focus point is determined after interactive adjustment back and forth along the view direction, a local coordinate system *U*-*V*-*W* can be established, with the origin anchored at the focus point and *W*-axis taking the opposite of the view direction. Then, a 2D rectangular vector field can be created exactly on the *U-V* plane, i.e., the cutting plane, to comprise a grid of uniformly distributed sample points. For each sample, the 3D *X-Y-Z* coordinate can be obtained from the 3D *U-V-W* coordinate by means of a space transformation. Next, the associated 3D *X-Y-Z* vector is evaluated in the global system through tri-linear interpolation before the 3D *U-V-W* vector is derived by an inverse space transformation. Finally, the *W*-component, perpendicular to the cutting plane, is discarded to yield a 2D *U-V* vector. In this way, a 2D vector field can be constructed on the cutting plane as a projection of the volume flow. Then, switching can be made at run time among arrows, ESS, and LIC to visualize this 2D flow, with the result mapped back onto the cutting plane in 3D settings. This entire pipeline is detailed in Figure 2.

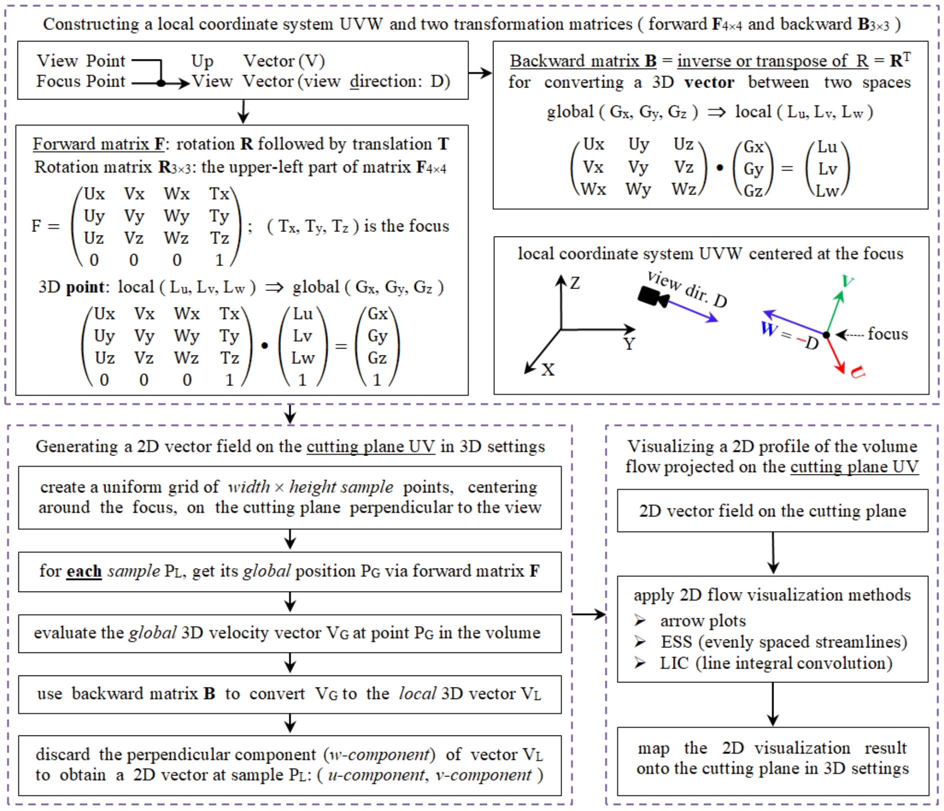
Constructing a local coordinate system UVW and two transformation matrices (forward $F_{4\times4}$ and backward $B_{3\times3}$)
View Point
Focus Point
Up Vector (V)
View Vector (view direction: D)
Backward matrix B = inverse or transpose of R = R^T
for converting a 3D vector between two spaces
global (Gx, Gy, Gz) ⇒ local (Lu, Lv, Lw)
$\begin{pmatrix} Ux & Uy & Uz \\ Vx & Vy & Vz \\ Wx & Wy & Wz \end{pmatrix} \bullet \begin{pmatrix} Gx \\ Gy \\ Gz \end{pmatrix} = \begin{pmatrix} Lu \\ Lv \\ Lw \end{pmatrix}$
Forward matrix F: rotation R followed by translation T
Rotation matrix $R_{3\times3}$: the upper-left part of matrix $F_{4\times4}$
$F = \begin{pmatrix} Ux & Vx & Wx & Tx \\ Uy & Vy & Wy & Ty \\ Uz & Vz & Wz & Tz \\ 0 & 0 & 0 & 1 \end{pmatrix}$; (Tx, Ty, Tz) is the focus
3D point: local (Lu, Lv, Lw) ⇒ global (Gx, Gy, Gz)
$\begin{pmatrix} Ux & Vx & Wx & Tx \\ Uy & Vy & Wy & Ty \\ Uz & Vz & Wz & Tz \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} Lu \\ Lv \\ Lw \\ 1 \end{pmatrix} = \begin{pmatrix} Gx \\ Gy \\ Gz \\ 1 \end{pmatrix}$
local coordinate system UVW centered at the focus
Z
Y
X
View dir. D
V
W = −D
focus
U
Generating a 2D vector field on the cutting plane UV in 3D settings
create a uniform grid of *width × height sample* points, centering around the focus, on the cutting plane perpendicular to the view
for each *sample* P_L, get its *global* position P_G via forward matrix F
evaluate the *global* 3D velocity vector V_G at point P_G in the volume
use backward matrix B to convert V_G to the *local* 3D vector V_L
discard the perpendicular component (*w-component*) of vector V_L to obtain a 2D vector at sample P_L: (*u-component*, *v-component*)
Visualizing a 2D profile of the volume flow projected on the cutting plane UV
2D vector field on the cutting plane
apply 2D flow visualization methods
➢ arrow plots
➢ ESS (evenly spaced streamlines)
➢ LIC (line integral convolution)
map the 2D visualization result onto the cutting plane in 3D settings

Figure 2. The pipeline of visualizing the projection of a volume flow on a view-perpendicular cutting plane.

In ExActiveFOVE, cutting-plane visualization can be toggled on/off dynamically as desired, running at real-time frame rates. Equipped with this utility like a handheld magnifying glass, the user is able to probe into the volume to gain insight from various angles, at different depths, and with multiple resolutions. In ExActiveFLOVE, multi-resolution visualization is attained by changing the uniform sampling interval and / or the size of the cutting plane. As for ESS, a third way is to adjust the density of evenly-spaced streamlines (Liu & Moorhead, 2006).

Pixel-Based Focus-Guided Content Culling

Visual clutter and view occlusion can be further addressed by culling part of the graphical contents in the scene. On one hand, this functionality prunes occluding, unimportant, or uninteresting elements (or those already examined) that may block or distract the user from investigating or emphasizing posterior, significant, or interesting items. On the other hand, this widget brings forth a new concept on *interacting* with the volume flow representation. While a streamline is considered a graphical *object*, we treat seeder-based user-driven placement of 3D streamlines (Section 3.1) as an *object-based compute-intensive back-end processing* mechanism since it governs what/where streamlines are generated. In comparison, we regard content culling as a *pixel-based focus-guided front-end processing* scheme in that part of a streamline, already generated and rasterized (i.e., converted into discrete pixels, or more precisely, fragments), is further manipulated by being retained or discarded in the display. In this sense, it fulfills low-level fine-granularity streamline “editing”. We believe that this capability is as instrumental as, and in parallel with, streamline placement. Both interactive placement (Section 3.1) and automated placement (e.g., ESS, Liu et al,,

2006) control the *generation* of streamlines, whereas they do not handle how (and what parts of) the streamlines are rendered. Content culling bridges this gap by providing great flexibility of what is selected for display in the view. In ExActiveFLOVE, it is deemed as a device for *image-oriented exploration*, which is seamlessly incorporated with seeder-based user-driven exploration (Section 3.1) to enhance volume flow visualization. Technically, our content culler is implemented by exploiting the stencil buffer of OpenGL. The scope of the culler is defined by either the inside or the outside of a rectangle (or of a circle) and the user can toggle between the two complementary areas.

It is worth emphasizing that a view-perpendicular cutting plane (Section 3.2) can be dynamically attached to (and detached from) the content culler at ease, which entirely avoids view occlusion for the 2D profile visualization. As the cutting-plane is animated back and back in the view direction, interior structures are then exposed to become the focus, with 3D streamlines beyond the cutting plane (also beyond the content culler) to make the context. Thus, the content culler is a smart real-time volume prober that can be translated (left / right / upward / downward / forward / backward) and toggled on/off. To our knowledge, the focus-guided content culler proposed here is the first in the literature for volume flow visualization. In fact, pixel-based front-end exploration has not yet attracted much attention in data visualization as most research has been revolving around object-based back-end exploration.

Prototypical Homogeneous VR Extension

The first room-sized Cave Automatic Virtual Environment (CAVE) designed by University of Illinois at Chicago in 1991 and the VR-Juggler library opened up opportunities for VR application development. The Virtual Wind Tunnel system developed at NASA Ames Research Center in 1993 erected a milestone for visualizing large CFD flows. In 2002, we also developed a CAVE-based system called TritonII-Flow (Liu et al, 2003) for interactive visualization of oceanographic-atmospheric volume flow data. VR facilities (e.g., CAVE) fueled by high-performance (parallel) computing are what professional-level serious visualization of *large (e.g., terabyte) scientific data* should resort to. Meanwhile, we might also consider mobile VR devices (e.g., HTC Vive, Oculus Rift), with very limited computing power, for use in testing, evaluation, education, and training. ActiveFLOVE and ExActiveFLOVE themselves are capable of delivering immersive (engaging) volume flow visualization, while we believe that VR can further augment immersive effects. VR is among the peripheral add-ons (Section 2.2) atop the ActiveFLOVE engine and therefore we select HTC Vive, coupled with Unreal, to facilitate a prototypical extension of ExActiveFLOVE to VR.

An overarching requirement is to avoid not only degradation of the high computational performance of the ActiveFLOVE engine (developed in C/C++) but also any obvious incompatibility issue (e.g., memory management), of which both are often incurred by C# (adopted in Unity, a VR development environment), Python, Java, etc. Thus, Unreal is our great choice of VR development because of the homogeneity with (Ex)ActiveFLOVE in C/C++. In the Unreal environment, linear splines (USplineComponent) are employed to "warp" cylinders (UStaticMesh) to create thin tubes around stream line segments and a rainbow color map (UMaterial) is utilized to encode the velocity vector magnitude. Besides this representation, strips (PersistentLineBatcher) can also be chosen for rendering streamlines.

RESULTS AND DISCUSSIONS

ActiveFLOVE is a toolkit made up of pure C++ classes, except for only several files with a tiny amount of code in support of platform-specific testing on visualization functionalities. It was developed using Microsoft Visual Studio (MS-VS) 1997 due to the algorithmic nature and the irrelevance to many new features of MS-VS. Recently, we switched to MS-VS 2008 to implement ExActiveFLOVE, without any extra dependence on MS Windows. Given the high computational performance (with statistics in Section 2.2) capable of real-time frame rates on an ordinary laptop for the proposed techniques without the need for any kind of parallelism, timing breakdowns are omitted herein for ExActiveFLOVE. These exploration-oriented techniques confine computing to only a subset of data for the local region, insensitive to large data sizes (Section 3). Thus, this section is focused on showing some visualization results of a wind volume flow (on a 360 × 181 × 13 Cartesian grid) to demonstrate the functionalities.

This rectangular volume of steady wind flow in the sky was derived from converting a spherically-coordinated dataset around the Earth. Thus, a map of the globe is displayed in Figure 3 to help with the spatial context of the "flattened" world. The settings of the 3D seeder common to the nine images (*a*-*i*) are capacity = 5 × 5 × 5 (for 125 seeds) and resolution = 2.0 (in cells of the grid). The *computational* speed for generating 125 streamlines ranges from 41.0 to 62.5 Frames Per Second (FPS) and the *scene rendering* speed varies from 6.4 to 21.3 FPS on a

Microsoft Surface laptop running Windows-10 with Intel i7-6600U CPU @ 2.60/2.81GHz and 16GB RAM. Except for a color-wheel map adopted for the backdrops in images *c* and *g-h*, a rainbow color map is used otherwise, with deep blue for the lowest velocity magnitude and full red for the highest. Carrying cone-shaped arrows aligned with the flow to indicate the positive direction, streamlines are rendered with co-dimensional illumination. In images *a-d*, the user is positioned at (46.3, 74.8, 7.7) , looking down and horizontally toward 16 degrees (counterclockwise from the east) and toggling on a view-perpendicular cutting plane after investigating flow structures in the baseline mode. The flow projected on this cutting plane is then visualized by 2D arrows, 2D evenly-spaced streamlines, and a 2D LIC image in order. Despite slight distractions from 3D streamlines in front of the plane, the pattern of the 2D profile can be well conveyed and easily recognized. In images *e-h*, the user is positioned at (66.3, 57.5, 6.8), looking down and horizontally toward 147 degrees (counterclockwise from the east), toggling on a rectangular content culler to entirely avoid not only visual clutter but also view occlusion from the outside and then from the inside of the rectangle, and “pasting” the culler (in the latter mode) on a view-orthogonal cutting plane that is shifted from near to far to probe 2D profiles of the flow by ESS. Image *i* is a snapshot of ExActiveFLOVE running in a VR environment, with an HTC Vive connected to a desktop PC, for a more immersive experience of volume flow exploration.

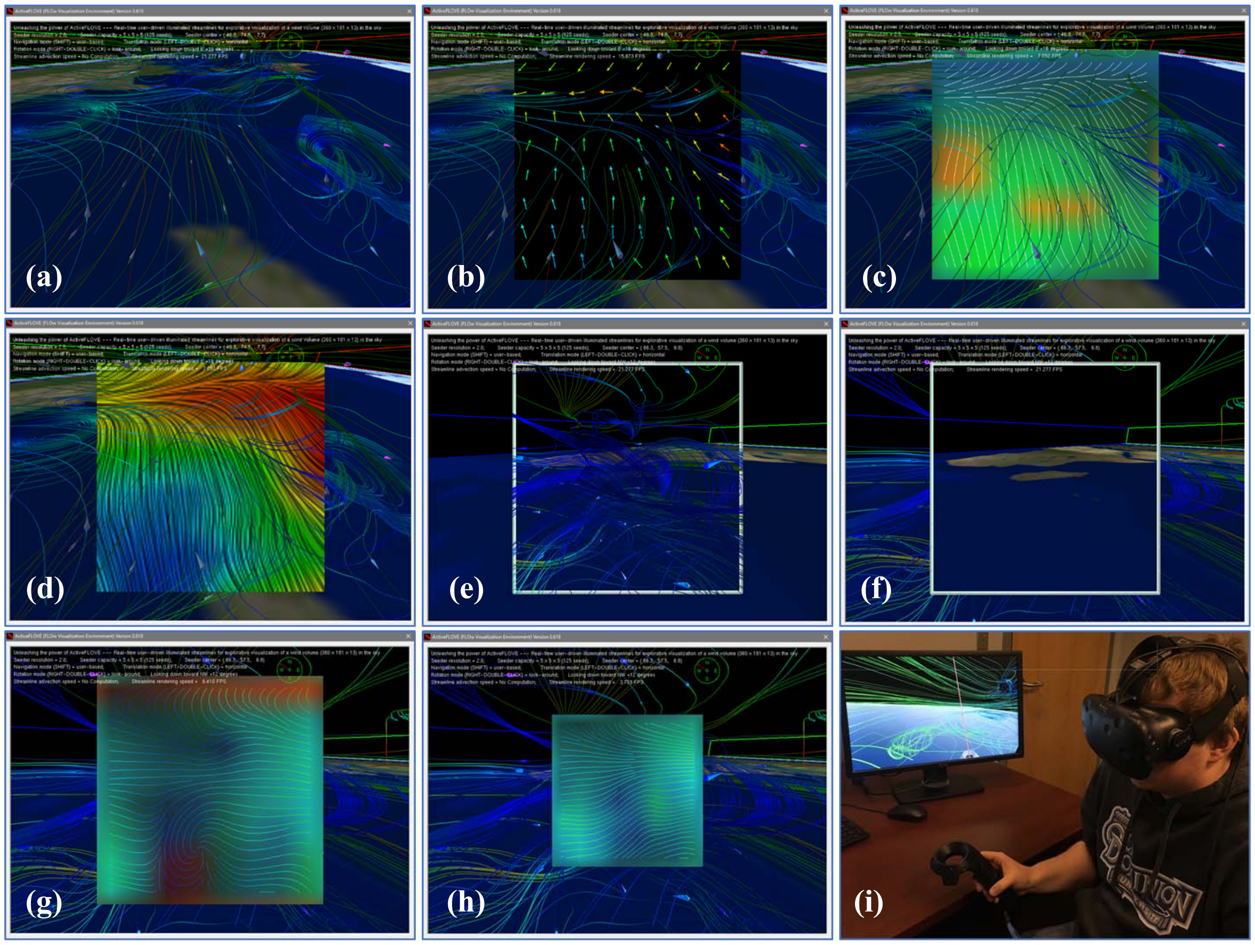

Figure 3. ExActiveFLOVE in execution for real-time visualization of a wind volume flow via seeder-based user-driven placement of 3D illuminated streamlines, with: (a) the baseline mode; (b, c, d) arrows, ESS, and LIC applied to a view-perpendicular cutting plane, respectively; (e, f) the contents outside and inside a square culled from the scene, respectively, (g, h) the culler stuck on a view-perpendicular cutting plane, shifted from near toward far to probe 2D profiles by ESS, (i) the VR mode in which an HTC Vive is attached to a desktop.

CONCLUSIONS AND FUTURE WORK

We have presented ExActiveFLOVE, which, built on top of our ActiveFLOVE engine, is able to attain real-time immersive explorative visualization of volume flows on an ordinary laptop without the need for any kind of parallel computing. The major contributions consist in reconfigurable seeder-based user-driven placement of 3D illuminated streamlines, dynamically maneuverable view-orthogonal cutting planes, a pixel-based focus-guided content culler, and a prototypical homogeneous extension to VR. Empowered by ultra-fast, highly accurate, very robust streamline integration and geared towards informed navigation plus ROI inspection, the four methods address view occlusion, visual clutter, and depth ambiguity in 3D settings. The explorative characteristic indicates that ExActiveFLOVE is insensitive to the data size and hence is well suited for interactive insightful visual analysis of big volume flow data.

As for future work, we plan to enrich ExActiveFLOVE by unleashing the combined power of available fundamental algorithms of ActiveFLOVE such as streamline integration on curvilinear/unstructured grids, pathlines, streak lines, and AUFLIC (Liu & Moorhead, 2005). We are also interested in adding a new layer for parallel visualization on top of ActiveFLOVE to incorporate a suite of algorithms that we have developed by multi-threading, GPU, and MPI.

ACKNOWLEDGEMENTS

This work was supported in part by the Program for Undergraduate Research and Scholarship (PURS) of Old Dominion University (ODU). The first two authors thank ODU-PURS for sponsoring the ExActiveFLOVE effort, except for seeder-based user-driven placement of streamlines (designed and implemented before the PURS project).

REFERENCES

Cabral, B., & Leedom, C. (1993). Imaging Vector Fields Using Line Integral Convolution. *Proceedings of ACM SigGraph '93*, 263-270.

Han, J., Tao, J., Zheng, H., Guo, H., Chen, D., & Wang, C. (2019). Flow Field Reduction via Reconstructing Vector Data from 3D Streamlines Using Deep Learning. *IEEE Computer Graphics and Applications, 39*(4), 54-67.

Interrante, V., & Grosch, C. (1997). Strategies for Effectively Visualizing 3D Flow with Volume LIC. *Proceedings of IEEE Visualization '97*, 421-424.

Jobard, B., & Lefer, W. (1997). Creating Evenly-Spaced Streamlines of Arbitrary Density. *Proceedings of Eurographics Workshop on Visualization in Scientific Computing*, 45-55.

Laramee, R., 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, 23*(2), 203-221.

Levoy, M. (1988). Display of Surfaces from Volume Data. *Computer Graphics and Applications*, *8*(3), 29-37.

Liu, Z. (2019). A Prototype Framework for Parallel Visualization of Large Flow Data. *Elsevier Advances in Engineering Software*, 130(April 2019), 14-23.

Liu, Z., Cai, S., Swan, E., Moorhead, R., Martin, J., & Jankun-Kelly, T. J. (2012). A 2D Flow Visualization User Study Using Explicit Flow Synthesis and Implicit Task Design. *IEEE Trans. Graphics & Vis., 18*(5), 783-796.

Liu, Z., & Moorhead, R. (2005). Accelerated Unsteady Flow Line Integral Convolution. *IEEE Transactions on Visualization and Computer Graphics, 11*(2), 113-125.

Liu, Z., & Moorhead, R. (2016). High-Performance Flow Visualization for Effective Data Analysis. *Journal of Flow Visualization and Image Processing, 23*(1-2), 41-57.

Liu, Z., Moorhead, R., & Groner, J. (2006). An Advanced Evenly-Spaced Streamline Placement Algorithm. *IEEE Transactions on Visualization and Computer Graphics, 12*(5), 965-972.

Liu, Z., Moorhead, R., & Ziegeler, S. (2003). Ocean Flow Visualization in Virtual Environments. *Technical Report MSSU-COE-ERC-03-03, Mississippi State University*, 1-50.

McLoughlin, T., Laramee, R., Peikert, R., Post, F. H., & Chen, M. (2009). Over Two Decades of Integration-Based Geometric Vector Field Visualization. *Proceedings of EuroGraphics '09*, 73-92.

Nguyen, D., Zhang, L., Monico, R., Thompson, D., Laramee, R., & Chen, G. (2019). Unsteady Flow Visualization via Physics Based Pathline Exploration. *IEEE Visualization '2019 (Short Paper Track)*, 1-5.

Shen, H.-W., Johnson, C. R., & Ma, K.-L. (1996). Visualizing Vector Fields using Line Integral Convolution and Dye Advection. *Proceedings of IEEE Symposium on Volume Visualization*, 63-70.

Stalling, D., & Hege, H.-C. (1995). Fast and Resolution Independent Line Integral Convolution. *Proceedings of ACM SigGraph '95*, 249-256.

Turk, G., & Banks, D. (1996). Image-Guided Streamline Placement, *Proceedings of ACM SigGraph '96*, 453-459.