Explorative Visualization of Volume Flow Data via Slice Textures

Ahmet Saglam
Old Dominion University
Norfolk, VA
asagl001@odu.edu

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

ABSTRACT

Scientific visualization is an important application of computer graphics to scientific computing, providing deep insight into the pattern underlying big data. Flow visualization plays a crucial role in oceanographic-atmospheric modeling, computational fluid dynamics simulation, and electro-magnetic field analysis in support of visual data exploration and interpretation toward scientific discovery. This paper presents a suite of techniques for effective visualization of volume flows by means of a texture-based representation. To address depth-cueing and view occlusion problems in 3D settings, we propose interactive extraction, explorative manipulation, and spatial correlation of 2D slices of the data volume that are visualized in a dense manner to reveal salient structures and intricate features. Specifically, the flow projected on each slice within a 3D region of interest is delineated through basic, oriented, and enhanced versions of line integral convolution in combination with grey-scale, rainbow, and color-wheel maps. By "sliding" an image slice along an axis, the user can visually reconstruct the 3D pattern, investigating local details while making sense of the global context. In addition, employing three such slices, one (dynamically toggled on / off) along each of three mutually orthogonal axes, improves the perception and understanding of complex structures embedded in the volume. Furthermore, the data volume can be explored via interactive retrieval of individual slices as well as automated animation of a stack of slices in succession and examined in depth by a smooth cyclic animation of the flow on any slice. The high computational performance of the proposed framework allows for explorative visualization of large volume flows on an ordinary laptop or desktop PC without dependency on graphics accelerators or GPUs.

ABOUT THE AUTHORS

Mr. Ahmet Saglam is a PhD student with the department of Modeling, Simulation, and Visualization Engineering at Old Dominion University (ODU-MSVE). He received the BS degree in Systems Engineering from Turkish Military Academy in 2008. After an MS study in Modeling, Virtual Environments, and Simulation at Naval Postgraduate School (September 2015 ~ December 2016), he transferred to ODU-MSVE (September 2017). Mr. Saglam performs research and development in scientific visualization, primarily in vector/flow data visualization and parallel visualization. His work on flow visualization is concerned with geometry-based methods (e.g., streamlines, pathlines, and streamline placement), texture-based approaches (e.g., line integral convolution and texture advection), vortex core detection, and immersive exploration of volume flows not only in VR environments but also on desktop PCs.

Dr. Zhanping Liu is an assistant professor of the department of Modeling, Simulation, and Visualization Engineering at Old Dominion University. He was a 2014-USRA summer visitor at NASA Ames Research Center and a 2013-FRPP summer visitor 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 School of Medicine at the University of Pennsylvania (2010 ~ 2011), Kitware, Inc. (2008 ~ 2010), and High-Performance Computing Collaboratory (formerly Engineering Research Center for Computational Field Simulation) at Mississippi State University (2001 ~ 2008) after a postdoctoral position with 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 mathematics from Nankai University (1992), both in P. R. China. His research interests remain in scientific data visualization, particularly flow/vector field visualization and recently parallel visualization. More information about his research and development is available at www.zhanpingliu.org.

Explorative Visualization of Volume Flow Data via Slice Textures

Ahmet Saglam
Old Dominion University
Norfolk, VA
asagl001@odu.edu

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

1. INTRODUCTION

As data grows at exponential rates by high-performance computing in a wide variety of science and engineering disciplines, it is necessary to turn to visual analysis to help with the understanding of big data. Given an array of data values acquired, generated, or defined on a 2D or 3D grid (i.e., mesh) of discrete sample points, scientific visualization either maps these values to intermediate geometric elements (e.g., points, lines, triangles, and tetrahedrons) followed by graphics rendering, or employs texture synthesis and image processing to create a visually continuous, dense depiction of the data field directly from the samples, both for intuitive perception and explorative analysis of the qualitative information. In general, scalar data visualization seeks to display the geometric structure or spatial distribution of a non-directional property, e.g., pressure, temperature, and height. A vector quantity is represented by two or three intrinsically coupled component values at each sample point and therefore vector (data) visualization or flow visualization differs from scalar data visualization in the extra requirement of conveying directional information. Flow visualization plays an important role in a broad range of areas such as oceanographic-atmospheric modeling, computational fluid dynamics simulation, and electro-magnetic field analysis by delivering deep insight into the pattern behind the scene.

While unsteady or time-varying flow visualization is well suited for investigating the spatial-temporal evolution of ocean currents (e.g., circulation of eddies), storm winds (e.g., hurricanes and tornados), air flows around delta wings, and fluid flows around submarines or in combustions, steady flow visualization is instrumental to examining the spatial structure of a flow at a time step, i.e., a snapshot. From the perspective of algorithm design, steady flow visualization paves the way for unsteady flow visualization. There has been considerable research on visualization of 2D (planar) flows, roughly categorized into sparse geometry-based (McLoughlin, Laramee, Peikert, Post, & Chen, 2009) and dense texture-based methods (Laramee et al., 2004). Among the most effective approaches are evenly-spaced streamlines (Liu, Moorhead, & Groner, 2006), Line Integral Convolution (LIC; Cabral & Leedom, 1993), and Image-Based Flow Visualization (IBFV; van Wijk, 2002). However, they exhibit some issues, when employed for (curved) surface flows, such as cluttering of streamlines, view occlusion, poor depth-cueing, and texture aliasing. In fact, these problems are exacerbated in visualizing volume flows. In particular, the latter three hinder texture-based methods from exposing interior structures, intricate features, and overall patterns in 3D settings.

Along the path of texture-based visualization of 3D steady flows, there has been a series of research efforts on volume LIC (Shen et al., 1996; Interrante et al., 1997; Salama et al., 1999). They spurred advances in flow texture synthesis and transfer function design, of which the latter is critical for Direct Volume Rendering (DVR; Levoy, 1988). On the other hand, this family of work also allows us to realize some intrinsic pitfalls. First, performing LIC on an entire 3D vector field incurs a large computational cost. In addition, the use of volumetric elements (voxels) for rasterizing 3D streamlines, as shown by the essence of the process of 3D LIC, introduces much inaccuracy to the representation of the flow direction. The subsequent low-pass filtering operation along each streamline adds to the inaccuracy by a "smearing" effect. After these steps, the directional information, originally in the form of continuous geometric curves (i.e., streamlines), has degenerated into an array of scalar data values (falling within a range of [0, 255]) generated on a rectangular lattice of discrete sample points (i.e., voxels). Furthermore, transfer function design remains a daunting task of DVR even for revealing internal structures out of scalar volume datasets (e.g., those resulting from ultrasound, CT scan, and Magnetic Resonance Imaging [MRI]). It is particularly more difficult for transfer functions to "extract" or "reconstruct" the directional information from a dense volume of 3D LIC. As a consequence, volume LIC tends to yield either a dense cloud-like volume with obscure directional cueing or only a small number of coarse thick curved bundles without the ability to capture rapidly changing flow directions. Last

but not least, the compute-intensive procedure of DVR significantly increases the overhead of the entire pipeline of volume LIC, leaving it heavily dependent on graphics accelerators or Graphical Processing Units (GPUs).
Motivated by the aforementioned observations, we propose to adopt 2D slices, textured by LIC images, for explorative visualization of volume flow data. As ground truth is usually not guaranteed by a single visualization result for an unknown complex volume flow, interactivity is the key to fruitful exploration and iterative visual analysis. LIC for an individual 2D slice outperforms that for the entire volume in response time, which enables spatial exploration to probe regions of interest. It also makes it possible to retrieve a stack of LIC slice images back and forth along an axis in search of potential features, in an interactive or automated fashion. Thus, domain experts may exploit visual memory retention to reconstruct the 3D flow pattern by perceptually concatenating the profiles in the longitudinal direction, like many radiologists even nowadays diagnose 3D organs from CT slices. Pattern recognition is further augmented by manipulating three mutually perpendicular intersecting slices. Smooth cyclic animation of flow trajectories on a slice of interest, user interaction with the data volume (e.g., translation, rotation, and scaling), and user navigation within the volume all help gain an improved understanding of the interior flow behavior. By dimensionality reduction, flow directions approximated by pixelization on a 2D plane are more accurate than those by voxelization in the 3D space. Likewise, flow path deviation caused by the "smearing" operation of LIC is less in 2D than in 3D. Without the need for transfer function design and DVR, 2D LIC maintains high computational performance as well as high-resolution high-fidelity flow streaks and supports diverse visual styles by LIC variants and color maps.

The remainder of this paper is organized as follows. Section 2 begins with a concise introduction to previous work, followed by a treatise of LIC as the foundation of sliced flow textures. In section 3, we present a suite of techniques for explorative visualization of volume flows through slices textured by LIC images. Results and discussions are provided in section 4 to demonstrate the capabilities and effectiveness of these visualization techniques. Section 5 concludes the paper with a brief summary and outlook on future work.

2. PREVIOUS WORK

Our work is dedicated to *texture*-based visualization of *steady volume* flows. Geometry-based methods (e.g., arrows, streamlines, pathlines, time lines, streak lines, stream arrows, stream ribbons, stream tubes, stream surfaces, and stream volumes) covered in a comprehensive survey (McLoughlin et al., 2009) are beyond our discussion below. Topology-based approaches (e.g., critical points, topological curves, and topological surfaces) enumerated in a thorough literature review (Laramee, Hauser, Zhao, & Post, 2007), which we feel might also fall within a macro version of the geometry-based family when considering the form of *graphical representation*, are also omitted in this section. Even for texture-based methods, our introduction treats surface flows and unsteady flows in a brief manner. Instead, this section is primarily focused on those related the most to, and those directly involved in, our work.

2.1 Texture-Based Techniques

The pioneering work in texture-based flow visualization began with spot noise proposed by van Wijk (1991). This method stretches and distorts, for a number of successive steps along the spatially varying vector directions, a collection of 2D oval texture *splats* (i.e., spots, of which each carries an intensity value) randomly placed in and continuously driven by the flow. These *advection* operations are implemented by warping the Cartesian (or regular) mesh underlying the flow and the visualization is then fulfilled by blending the intensity values that each pixel receives. Due to the excessive spot size and the large advection step size, both relative to the cells of the grid or the pixels of the image, this method is susceptible to missing small or turbulent flow features. In other words, the resolution of delineation and the accuracy of flow paths are insufficient.

In light of the limitations of spot noise, Cabral and Leedom (1993) presented LIC for high-resolution high-fidelity depiction of 2D flows. Each pixel of the output image, i.e., a target pixel, is taken as a seed position to generate a streamline in the flow domain. By indexing white noise, a low-pass filter is applied to a set of pixels hit by this curve to perform convolution to determine the intensity value for the target pixel. The use of white noise increases the granularity of "texture splats" to a level of pixel. A considerable number of samples, with the interval between any two consecutive ones no greater than a pixel size, are chosen along each streamline to enhance the accuracy of flow advection. The convolution mechanism allows for a variety of low-pass filters, by which phase shifting can be

further exploited to produce a sequence of animated frames of LIC. With these advantages, LIC is able to achieve high-quality dense flow visualization, showing the overall pattern and exposing local features in a crispy image. However, by the nature of pixel-oriented texture synthesis, LIC is a computationally expensive process.

The subsequent one decade was tinted mainly with research in texture-based flow visualization. Among the follow-up work of LIC are fast LIC (Stalling & Hege, 1995) for one order-of-magnitude speedup by reusing streamlines and for multi-resolution support, acceleration of LIC by parallel computing (Zockler, Stalling, & Hege, 1996), extension of LIC to curvilinear grids (Forssell & Cohen, 1995) and triangulated surfaces (mounted on 2.5D unstructured grids; Teitzel, Grosso, & Ertl, 1997), encoding of the velocity vector magnitude by multi-frequency noise design (Kiu & Banks, 1996), *oriented LIC* for adding the cue of positive flow direction (Wegenkittl, Groller, & Purgathofer, 1997), and accentuation of "embedded" flow streaks by *enhanced LIC* (Okada & Kao, 1997). By means of an image-space oriented (or pixel-oriented), Eulerian-based, value-collecting scheme, LIC is well suited for visualizing steady flows. However, it cannot be directly employed to handle unsteady flows. Otherwise, flickering or abruption artifacts would emerge in the resulting animation of LIC frames due to the temporal coherence (i.e., smooth transition between consecutive frames) being ignored at all. To deal with this issue, Shen and Kao (1998) proposed Unsteady Flow LIC (UFLIC) by adapting LIC for effective visualization of 2D time-varying flows. UFLIC takes an object-space oriented (or particle-oriented), Lagrangian-based, value-scattering strategy to address not only spatial coherence but also temporal coherence. Thus, great image contrast is exhibited in individual frames and strong temporal coherence is also maintained in the animation. Then, Liu and Moorhead (2005) presented Accelerated UFLIC (AUFLIC) that, by reusing pathlines, is one order-of-magnitude faster than UFLIC yet with the same image and animation quality.

Motivated by spot noise and LIC was a flurry of research in the incorporation of texture advection and frame blending. Texture advection is intended to establish temporal coherence through particle-oriented Lagrangian-based pathlines, while frame blending seeks to construct (or "insert") spatial coherence by performing low-pass filtering between a number of successive frames. Thus, most algorithms of this type, e.g., LEA (Jobard, Erlebacher, & Hussaini, 2002) and UFAC (Weiskopf, Erlebacher, & Ertl, 2003), are capable of visualizing both steady and unsteady flows by considering the entire spatiotemporal domain. One drawback is insufficient spatial coherence, in the form of blurry appearance and obscure flow directions, sacrificed for temporal coherence. Behind the operation of frame blending, in essence, is an exponentially-decaying low-pass filter, which is not good at introducing sufficient similarity between pixels along the vector direction to convey flow streaks. Also under the same category is Image-Based Flow Visualization (IBFV) proposed by van Wijk (2002). IBFV is a well-known, easy-to-implement, super-fast, and versatile algorithm, with an ability to emulate many geometry-based and texture-based techniques such as arrow plots, mesh warping, flow topology, particle tracing, noise smearing, and time lines yet without dependency on graphics accelerators or GPUs.

Given the issue in spatial coherence, the combination of texture advection and frame blending is not as good as LIC at visualizing steady volume flows. Volume LIC is a straightforward extension of LIC to steady volume flows by replacing a 2D noise texture, pixels, and 2D streamlines with a 3D noise texture, voxels, and 3D streamlines, respectively. The major difference is that the intermediate output is a 3D flow texture, i.e., a scalar data volume, which needs to be shown using direct volume rendering or DVR. There are several methods for DVR, e.g., ray casting (Levoy, 1988), shear-warp, splatting, 2D texture mapping, and 3D texture mapping. Ray casting is a famous widely-used algorithm for displaying scalar volume data acquired from ultrasound, CT, and MRI. As for the use in volume LIC (Figure 1), the most stubborn task is to design an effective transfer function to map flow texture values to R-G-B colors and the associated opacity (or transparency) values.

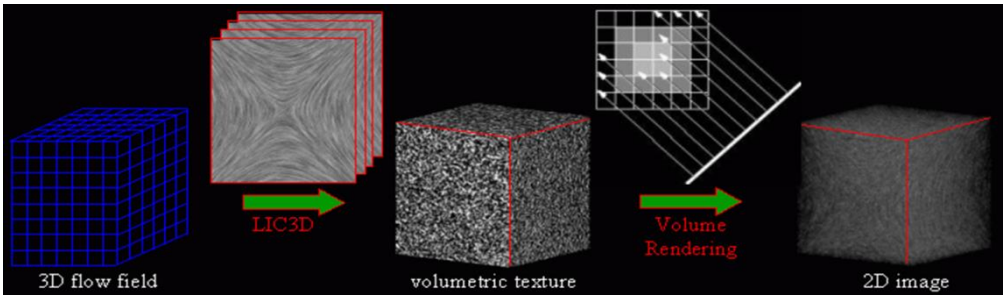

Figure 1. The pipeline of volume LIC consisting of 3D LIC and direct volume rendering to yield a 2D image.

There has been quite some work on volume LIC in the literature. Three algorithms reflect some typical weaknesses of this method. In one case, volume LIC delivers a translucent cloud-like volume (Shen et al., 1996), with view occlusion, ambiguous depth cueing, and indiscernible flow directions. In a second case, depth cueing and flow directions may be enhanced, though the granularity of depiction is decreased such that only coarse thick curved bundles are present in the result image and they are insufficient to cope with small or turbulent features densely distributed in a volume (Interrante & Grosch, 1997). These two cases both incur heavy computational costs. A third scenario (Rezk-Salama et al., 1999) involves graphics hardware. Like in the first two cases, the flow streaks are not accurate because of voxelization of 3D streamlines, flow path deviation caused by convolution along a voxelized curve, and ad-hoc transfer function design. We believe that the fundamental problem with volume LIC is the degradation of directional information from a continuous geometric representation (a 3D streamline) to a jaggy chain of discretized voxels of which the associated scalar values do not explicitly mean directional information any more or even any other physical property. Sophisticated transfer function design in volume LIC is an awkward remedy for this degradation or loss of information.

Direct display of 2D slices, each textured by an LIC image, allows us to avoid many problems of volume LIC. Then the primary challenge is how to facilitate the user to visually reconstruct and perceptually assimilate both the 3D domain and volumetric structures from textured slices.

2.2 Line Integral Convolution

Originating from a seed position (where a particle is released), a *streamline* (McLoughlin et al., 2009; Laramee et al., 2004; Liu et al., 2006) is a field line (i.e., the trajectory of the particle) that is point-wise tangent to the local velocity vectors. This curve is generated by connecting a sequence of sample points as the successive "snapshots" of the particle driven by and moving along the flow, while the samples are calculated, step by step from the seed, via numerical integration in a way to solve an ordinary differential equation. Despite a geometry-based method, streamlines serve as a prerequisite for the texture-based LIC method. As a dense representation, LIC was proposed by Cabral and Leedom (1993) in ACM SigGraph'93 for visualizing 2D steady flows. The basic idea is to low-pass filter white noise along streamlines, with each traversing through a chain of pixels. It is a 1.5-dimensional curve-aligned anisotropic "image" convolution process, emulating what happens when a rectangular area of fine sand is blown by a gust of strong wind (Figure 2). LIC exploits spatial correlation in the tangential direction to expose flow streaks but suppresses spatial correlation in the orthogonal direction to create intensity "gaps" between the streaks, resulting in a "smearing" effect.

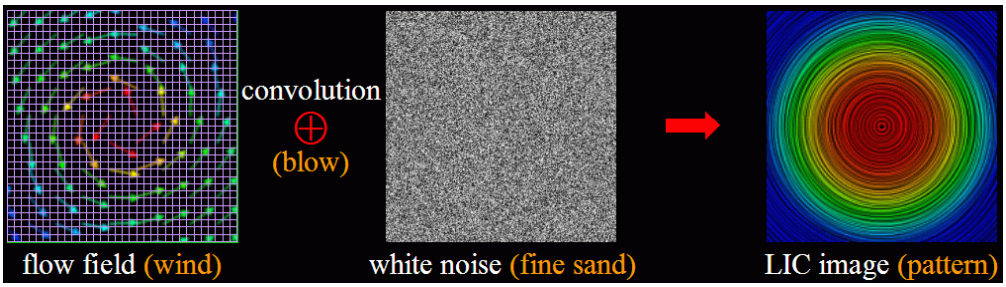

Figure 2. The basic idea of LIC analogous to the wind blowing across a patch of fine-grained sand.

LIC is usually implemented using an image-space oriented (pixel-oriented), Eulerian-based, value-collecting pipeline (Figure 3). First, a streamline is integrated from the center of each pixel of the output image in both directions but with the same length. Then, the correlated pixels are located along the streamline. Next, white noise is accessed to *collect* their initial (noise) values. Finally, convolution (i.e., normalized weighted averaging) is applied to these values to determine the output pixel value. The pattern of the LIC image is depicted by the "embedded" or "embossed" flow streaks, of which each comes into shape by taking similar pixel values along the streamline.

The quality of an LIC image is influenced by the number of correlated pixels along each streamline participating in the noise texture convolution. It is the number of times that a low-pass filter function is sampled to evaluate a weight value for each participatory pixel. Thus, it is also denoted as the filter length, analogous to the strength of wind force (Figure 2) and / or the wind duration. Within a certain range, the larger the filter length is, the smoother the white noise texture is smeared or "combed" and hence the more outstanding the flow streaks are out of the image. The LIC

image quality is also concerned with the differentiability between neighboring streaks. This factor, closely related to image contrast, governs how easily and how well the tangential flow direction is visually discerned.

This *basic LIC* algorithm takes white noise as the input texture and many low-pass filters can be employed for the convolution, e.g., those based on box, ramp, and Hanning kernels. In addition, other types of noise may be designed to meet specific needs. The purpose of a noise texture is to impose an amount of randomness, more or less, to avoid artifacts that would otherwise show up because of the regular lattice on which the Cartesian flow field, the input texture, and the output image all are mounted.

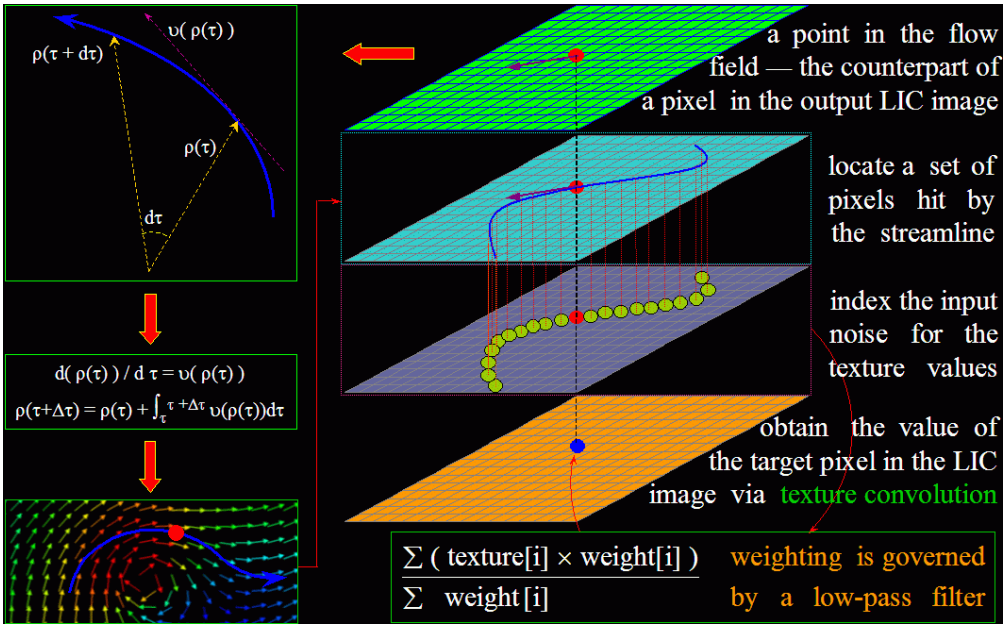

Figure 3. The pipeline of 2D LIC by image-space oriented Eulerian-based value-collecting implementation.

3. EVV-FLOST

In this section, we present Explorative Visualization of Volume Flow data via Slice Textures (**EVV-FLOST**). We begin with extraction of regions of interest, followed by slice visualization including LIC variants, color maps, and animation of LIC frames. Then, we address depth awareness, visual reconstruction of the flow volume, and perceptual recognition of internal structures by adding support for interactive retrieval of slices as well as automated animation of slices. Also considered for explorative visualization are data volume manipulation and user navigation.

3.1 Slice Visualization

Given a volume flow defined on a Cartesian grid, a Region of Interest (ROI) may be specified to reduce the domain of visual investigation. For big data, ROI selection is a good practice to save memory footprint, computational cost, and graphics rendering by excluding unimportant portions that may be detected in a lightweight pre-processing stage, e.g., calculation of local entropy values. For medium- or small-sized data, ROI selection allows the user to focus on a sub-region of the volume for thorough inspection, e.g., with a high resolution. This case is well aligned with and part of explorative visualization of volume flows. Thus, an ROI may be chosen by user interaction, possibly along with a scaling factor for vector data super-sampling. To obtain a super-sampled ROI, trilinear interpolation is usually adopted

Once a volume ROI is ready, a rectangular slice of flow data can be extracted along (and perpendicular to) one of three axes by discarding the vector component in that direction while retaining the other two components. In general, such a slice directly comes from a plane of original voxels. ROI super-sampling enables a sufficient resolution in each dimension and hence obviates the need to probe an arbitrary plane other than a “raw” slice of the

volume. In fact, EVV-FLOST maintains three mutually orthogonal stacks of slices (i.e., XY slices, YZ slices, and ZX slices) in memory, with each stack perpendicular to an axis.

For the 2D vector field acquired on each slice, or called a *slice flow* for short, three variants of LIC are available for visualizing the pattern. The Basic LIC (**BLIC**) algorithm proposed by Cabral and Leedom (1993) produces a blurry appearance despite the use of a sufficient filter length. This is because white noise is too "dense" for the vector field to curve out clear flow trajectories by only one pass of convolution. An analogy for this situation is that an area of sand is too compact for a single gust of wind to leave a path of tangible footprints. It implies that executing two passes of LIC may strengthen flow streaks to help them stand out of a rectangular array of pixels, as is the case with Enhanced LIC (**ELIC**; Okada & Kao, 1997). In more detail, ELIC applies a high-pass filter to the output image of a normal pass of LIC (i.e., the first pass) to increase image contrast so as to sharpen flow streaks. Then, this intermediate result is fed forward to the second pass of LIC to serve as the input texture in place of white noise. ELIC is able to synthesize a crispy image with accentuated flow paths.

BLIC and ELIC show only the tangential flow direction, without any cue for the positive direction, i.e., *orientation*. This lack is attributed to a symmetric low-pass filter. An *a*symmetric filter, e.g., one designed from a ramp kernel, discriminates between the two sides of the seed of a streamline regarding the weighting values in convolution. As a result, the negative and positive directions of the flow are visually distinguished by introducing, to the flow streak, an intensity-tapering effect, as exhibited by the trace of a comet. However, intensity tapering requires sufficient length in the tangential direction and sufficient gaps in the orthogonal direction to demonstrate an orientation metaphor by an illuminated "ribbon". Thus, sparse noise needs to be designed to achieve Oriented LIC (**OLIC**; Wegenkittl et al., 1997).

Apart from direction and orientation, velocity vector magnitude is another important aspect, though a scalar quantity, that flow visualization usually takes into account. In general, there is some degree of spatial correlation between flow direction and velocity magnitude and therefore the latter lends itself to the understanding of the former. As a channel in parallel with geometry and texture, color mapping is often utilized to encode velocity magnitude. A rainbow map (lower part, Figure 4) linearly maps the lowest magnitude to deep blue and the highest to full red. To deal with a highly non-uniform distribution of velocity magnitude values within a vast range (e.g., a range spanning many orders of magnitude), they are converted by a non-linear transformation prior to color mapping. Otherwise, direct map would not make full use of the limited spectrum of visually differentiable colors, making many major colors absent from the visualization result.

A color wheel (upper right, Figure 4) maps orientation and magnitude of the flow to the three components of HSV color space (hue, saturation, and value or brightness), e.g., with vector [1.0, 0.0] or the eastward orientation indicated by the red hue, vector [−0.5, 0.866] by the blue hue, and the velocity magnitude increasing from the center (the darkest point of the color wheel) outwards to the circular boundary (the brightest edge) along each radial line. By establishing a one-to-one correspondence between the hue and the flow orientation, this color map creates a bright "dot" at the core of each center or focus, which facilitates identification of topological features.

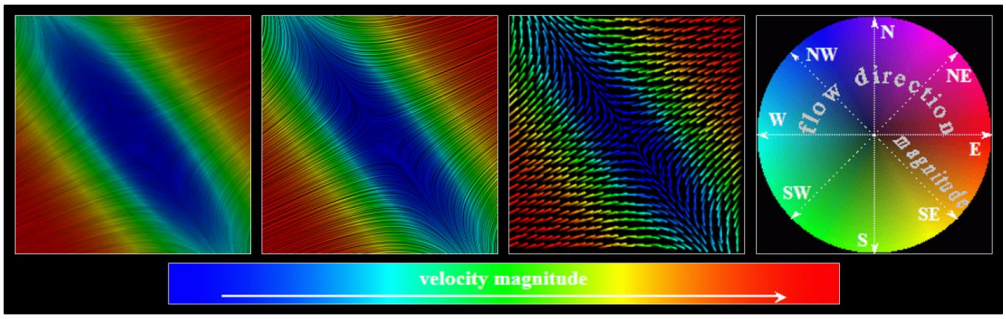

Figure 4. A slice flow visualized by using BLIC, ELIC, and OLIC (in succession, upper left) with a rainbow color map illustrated at the lower part. Also shown is the legend of a color wheel map (upper right).

Given an ROI volume flow, the initial velocity vector magnitude values are transformed, in EVV-FLOST, to [1.0, 1,000.0] by histogram equalization to guarantee consistent, hue-effective, and aesthetic color map of slice images

besides a grey-scale version. The color wheel mode linearly maps the magnitude to brightness within [0.2, 1.0] and to saturation within [0.4, 1.0]. For any slice image, the user can interactively switch between three schemes: grey-scale, rainbow map, and color wheel. BLIC and ELIC adopt a 30-pixel-wide box kernel coupled with white noise. In particular, ELIC applies a 3×3 Laplace high-pass filter between two passes of LIC. OLIC employs a 20-pixel-wide ramp kernel and sparse noise, while the latter is synthesized by jittering a 3×3 white cross, i.e., a textured splat, within each of a set of 9×9 black quads uniformly distributed in the rectangular domain of the slice flow. The purse of jittering splats is to implement a semi-random placement by preventing them from overlapping with one another. Figure 4 shows three results of visualizing a slice flow by BLIC, ELIC, and OLIC (the first three in the upper row), respectively, with the same rainbow color map to qualitatively represent the velocity magnitude. Both ELIC and OLIC can display a saddle point, very well, which is though hard to notice in the BLIC image. The ELIC result shows sharp flow paths in a dense fashion. The OLIC image conveys the flow orientation, though at the cost of the resolution of flow delineation, leaving some areas uncovered in the image.

LIC is intended to visualize a 2D steady flow, of which the velocity vectors do not vary over time. In other words, the flow pattern is fixed to a single time step of vector data. However, if a filter function is based on (i.e., an integral form of) a periodic kernel, a sequence of animated LIC frames can be created by increasing the phase of the kernel from 0 to 2π in an evenly-angled manner. This effect of visual motion without either physical movement or shape change, produced by phase shifting, is similar to the motion of a sine or cosine curve function along the X axis. With 45 degrees as the uniform interval in EVV-FLOST, eight frames are generated using a Hanning kernel to support smooth cyclic animation of a slice flow that the user selects for examination.

3.2 Volume Exploration

DVR of CT/MRI data or 3D LIC flow textures can be visually appealing, though the blending operation performed on a collection of interpolated planes, perpendicular to and along the view direction, sacrifices the depth information. As a consequence, difficulties arise as to proper assimilation of the relative spatial relationships between objects (possibly tiny, small, large, thin, thick, regular, or irregular) in 3D settings. This issue hinders the practical applicability of DVR to medical diagnosis and that of volume LIC to 3D visualization of Computational Fluid Dynamics (CFD) flows. On the other hand, the position of a slice, textured by the associated LIC image, along the axis of reference can be well perceived. The depth of one slice relative to another parallel slice is intuitive. By arranging several parallel slices, either evenly-spaced or not along the axis of reference, and dynamically toggling them on/off as needed, the flow volume can be visually reconstructed in the longitudinal direction. Meanwhile, the user can also make sense of internal structures by correlating the flow patterns shown on multiple parallel slices. These multiple parallel slices are denoted in EVV-FLOST as a *group*.

Incorporating three such groups of parallel slices along X, Y, and Z axes, respectively, helps correlate the pieces of 3D information obtained initially and separately along three longitudinal directions. This kind of spatial and visual correlation allows three channels of cognitive work to compensate for one another in a way to provide a spatial registration mechanism. By adjusting the position of mutually orthogonal intersecting slices, the user can refine visual reconstruction of the flow volume and perceptual recognition of internal structures. It is worth mentioning that, essentially, each group of slices is not necessarily perpendicular to X, Y, or Z axis. Instead, each can be oriented toward an arbitrary direction that the user designates. In this case, such a group turns into a flexible set of *cutting planes* embedded in the data volume, while these parallel cutting planes may or may not differ in the size, depending on the initial configuration or dynamic adjustment by the user. Then there can be more than three such arbitrarily oriented stacks of (mutually parallel) slices. By interactively tweaking the orientation of each group and the size of each individual cutting plane, the user can have a much improved perception, cognition, and understanding of the volume and flow structures. This extended exploration strategy works by means of a synergistic *prober* composed of multiple planar data viewers. We believe that this is a promising solution to gain insight into complex volume flows and will be implemented in the next version of EVV-FLOST.

Apart from the aforementioned concept of *group*, *stack* is another one in EVV-FLOST. A stack refers to an array of *all* original slices, textured by LIC images, of the flow volume along one axis. EVV-FLOST maintains three stacks of LIC slices in support of interactive retrieval of individual slices and automated animation of all or part of the slices of a stack. Given an arbitrary slice, only a rectangular array of single-channel intensity values (falling within a range of [0, 255]) needs to be generated and stored in memory to represent the *LIC Pattern* of the associated 2D vector field, or *LicPat*, for each of the three LIC *variant*s (i.e., BLIC, ELIC, and OLIC). Thus, there are three

LicPats for the same slice and each LicPat depicts the *geometric structure* (e.g., the position, shape, and length of a flow path) and *topological structure* (i.e., the position, type, and size of a salient topological element such as node, focus, center, or saddle and the inter-connectivity between such elements) of the flow by one of the three styles of LIC synthesis, independent of color information. Then, the grey-scale image of each LIC variant is assembled on the fly by duplicating every element of the corresponding LicPat twice to make up R-G-B tuples. The other two *version*s, i.e., images, of the same LIC variant are created, upon the display, by rainbow and color wheel maps, respectively. The total number of LicPats for the entire flow volume is (1 LicPat *per* LIC variant *per* slice) × (3 LIC variants) × $\sum_{\boldsymbol{i}}$ (number of slices along axis ***i***).

Although there are multiple and even many LIC slices simultaneously shown in the scene, only a single image block of memory, with the size determined by the largest slice among three stacks, is allocated as a buffer to temporarily hold the slice image, synthesized on the fly, which is immediately submitted for texture mapping by the OpenGL graphics rendering engine. Please note that a LicPat is not produced until the target variant of the very slice is accessed for the first time, by either interactive retrieval or automated animation. This on-demand deferred LIC scheme avoids computing resources wasted on slices that may not be actually accessed, which, equipped with our high-performance implementation of the LIC engine, ensures decent frame rates for EVV-FLOST.

Interactive retrieval of slices out of a stack enables random probing into and preliminary check of the flow volume during the course of identifying an ROI. Dynamically toggling on/off some slices wherever necessary addresses occlusion issues. Automated animation of a stack of slices in succession offers another way for probing the volume and detecting potential features. In addition, the rapid smooth replay process significantly facilitates visual reconstruction of the flow volume and perceptual cognition of internal structures through visual memory retention.

Explorative visualization in EVV-FLOST also includes volume manipulation and user navigation. The user can interact with the entire data volume by rotation to get the look and feel from a different perspective, by scaling to see more detail, and by translation to approach or leave the data field. The latter case is particularly instrumental to volume flow visualization. Explorative visualization might begin with slice-based operations, usually with the user "placed" outside the volume. This high-level, relatively "distant" viewing experience is good at and possibly necessary for attaining an understanding of the global context and overall pattern of the flow. Then, putting the user into the data volume, with immersive navigation, helps capture and examine interior flow features in depth.

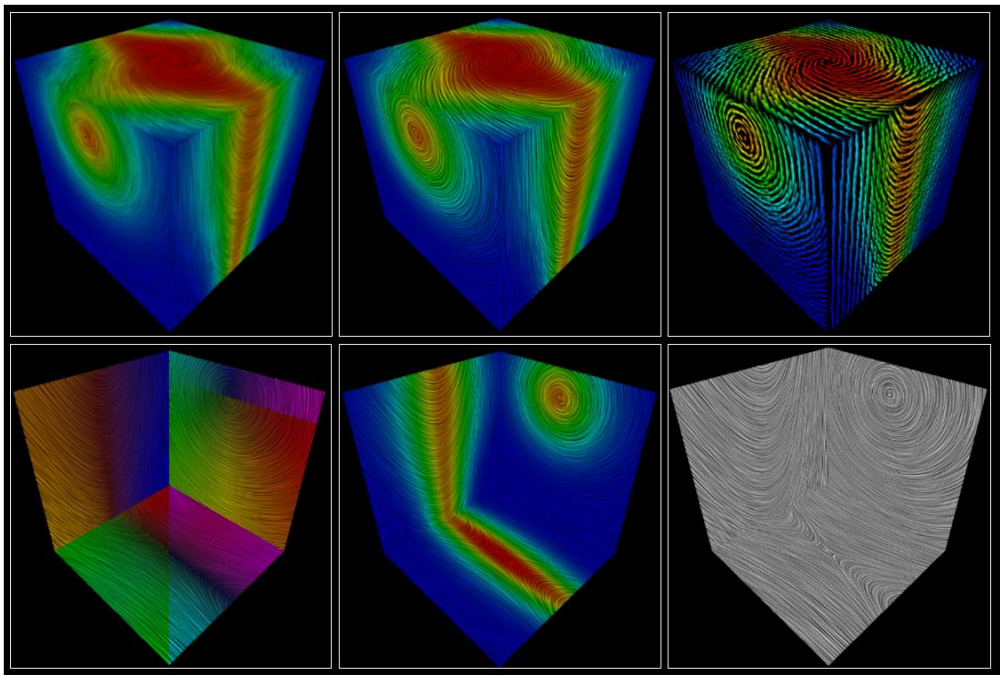

Figure 5. Visualization of the ROI volume flow by (a) BLIC, (b) ELIC, and (c) OLIC with a rainbow color map for three mutually perpendicular slices (all slice indices: 255), and by ELIC with (d) a color wheel, (e) a rainbow map, and (f) a grey-scale depiction for three mutually orthogonal slices (all slice indices: 0).

4. RESULTS AND DISCUSSIONS

We have developed a prototype system of EVV-FLOST by using Visual C++ and OpenGL on a Dell Latitude 5580 laptop (Intel Core i7-7820HQ 2.90GHz, 16GB RAM) with Windows 10. A 3D Lorenz vector field, generated by mathematical equations on a 512×512×512 Cartesian grid, serves as a test dataset for EVV-FLOST. The sub-volume in the middle part, with a size of 86×86×86, is extracted as an ROI and then scaled by a factor of 3 to get a volume flow defined on a 256x256x256 Cartesian grid.

The upper row of Figure 5 shows three mutually orthogonal boundary slices of the ROI volume flow visualized by BLIC, ELIC, and OLIC with a rainbow color map. Each slice displays a 2D profile of the 3D flow on the boundary plane, indicative of a similar structure deeper into the volume. The three slices in combination give a basic idea of the 3D flow behavior. The consistency across the edges, in the velocity magnitude exhibited by the rainbow map, lends itself to visual reconstruction of the 3D pattern. The lower row of Figure 5 shows the opposite set of mutually perpendicular boundary slices visualized by ELIC but with different color modes. Although the color wheel introduces inconsistency across the edges, it reveals flow orientations in a dense representation other than OLIC.

Figure 6 demonstrates a tri-slice (or even multi-slice) prober initially positioned at the exact center and then moved within the data volume to aid in visual reconstruction of 3D structures. Fixing any two slices but sliding the rest along its axis back and forth augments the understanding of the interior flow pattern. Figure 7 illustrates a typical process of flow exploration by volume manipulation and user navigation (Section 3.2). The user begins by looking down upon three mutually orthogonal slices to get an impression of the flow. Then, the data volume is rotated by 180 degrees for the user to see the other side. An alternative choice is that the user may go into the volume from the original direction to check more detail. The user may slide the slices to enrich his / her knowledge of the flow.

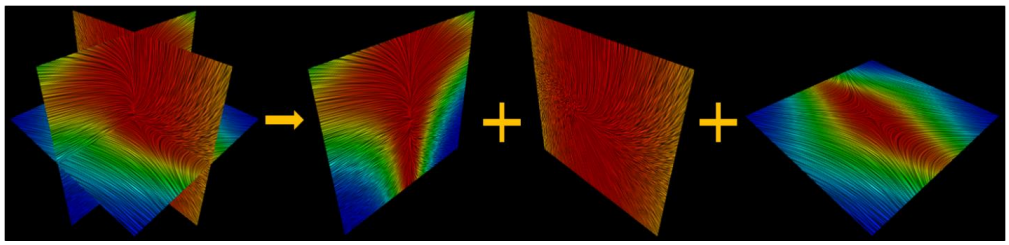

Figure 6. Visualization of the ROI volume flow by positioning three mutually perpendicular slices exactly at the center of the domain (all slice indices: 127). This tri-slice prober correlates multiple views to enhance visual reconstruction and perceptual recognition of the 3D structure.

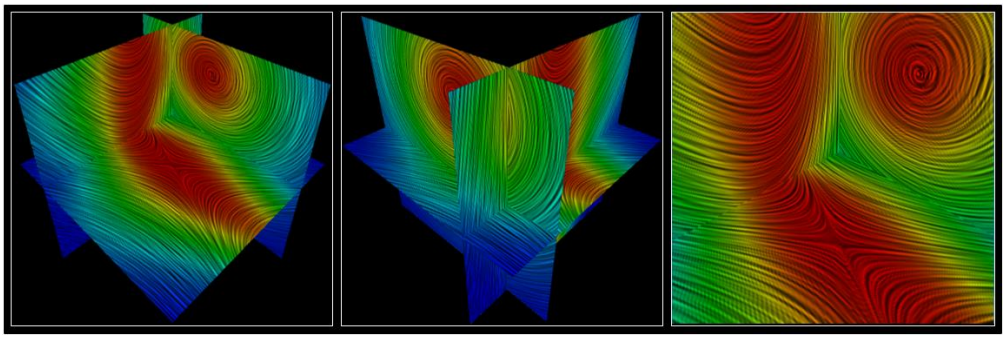

Figure 7. Volume rotation (from the left to the middle) and user navigation (from the left to the right).

5. CONCLUSIONS AND FUTURE WORK

We have presented EVV-FLOST for *explorative visualization of volume flow data via slice textures*. This work is based on our observation that 3D texture synthesis, regardless of volume LIC or texture advection, is not well suited for visualizing volume flows because of depth cueing, view occlusion, obscure flow direction, insufficient path

accuracy, low delineation resolution, high computational costs, and other issues that DVR may not resolve by transfer function design. We propose to exploit 2D slices, textured by three LIC variants with two color map modes, to cope with these challenges by volume exploration techniques such as manipulation of mutually perpendicular slices (e.g., toggling on/off and sliding), interactive retrieval of slices, and automated animation of slices. A preliminary test with a prototype implementation of EVV-FLOST demonstrates the effectiveness, reflective of a promising direction for further research efforts. As for future work, we plan to investigate arbitrarily oriented slices, interactive manipulation of a multi-slice prober, addition of illuminated streamlines, and extension to VR / AR.

REFERENCES

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

Forssell, L. K., & Cohen, S. D. (1995). Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation, and Unsteady Flows. *IEEE TVCG,1*(2), 133-141.

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

Jobard, B., Erlebacher, G., & Hussaini, M. (2002). Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization. *IEEE TVCG, 8*(3), 211-222.

Kiu, M. H., & Banks, D. C. (1996). Multi-frequency Noise for LIC. *IEEE Visualization*, 121-126.

Laramee, R. S., Hauser, H., Doleisch, H., Vrolijk, B., Post, F. H., & Weiskopf, D. (2004). The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum, 23*(2), 203-221.

Laramee, R. S., Hauser, H., Zhao, L., & Post, F. H. (2007). Topology-Based Flow Visualization, The State of the Art. *Mathematics and Visualization Topology-Based Methods in Visualization,* 1-19.

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

Liu, Z., & Moorhead, R. (2005). Accelerated Unsteady Flow Line Integral Convolution. *IEEE TVCG, 11*(2), 113-125.

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

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

Okada, A., & Kao, D. L. (1997). Enhanced Line Integral Convolution with Flow Feature Detection. *Proceedings of IS & T / SPIE Electronics Imaging*, Vol. 3017, 206-217.

Rezk-Salama, C., Hastreiter, P., Teitzel, C., & Ertl, T. (1999). Interactive Exploration of Volume Line Integral Convolution Based on 3D-Texture Mapping. *Proceedings of IEEE Visualization*, 233-240.

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.

Shen, H. W., & Kao, D. (1998). A New Line Integral Convolution Algorithm for Visualizing Time-Varying Flow Fields. *IEEE TVCG, 4*(2), 98-108.

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

Teitzel, C., Grosso, R., & Ertl, T. (1997). Line Integral Convolution on Triangulated Surfaces. *Proceedings of the Conference in Central Europe on Computer Graphics and Visualization*, 572-581.

Wegenkittl, R., Groller, E., & Purgathofer, W. (1997). Animating Flow Fields: Rendering of Oriented Line Integral Convolution. *Proceedings of IEEE Computer Animation*, 15-21.

Van Wijk, J. J. (1991). Spot Noise Texture Synthesis for Data Visualization. *ACM SIGGRAPH Computer Graphics, 25*(4), 309-318.

Van Wijk, J. J. (2002). Image Based Flow Visualization. *ACM Transactions on Graphics, 21(3),* 745-754.

Weiskopf, D., Erlebacher, G., & Ertl, T. (2003). A Texture-Based Framework for Spacetime-Coherent Visualization of Time-Dependent Vector Fields. *Proceedings of IEEE Visualization*, 107-114.

Zöckler, M., Stalling, D., & Hege, H. C. (1997). Parallel Line Integral Convolution. *Parallel Computing*, *23*(7), 975-989.