VISUALIZING TIME-VARYING THREE-DIMENSIONAL FLOW FIELDS USING ACCELERATED UFLIC

Zhanping Liu, Robert J. Moorhead II
Mississippi State University

Keywords: *unsteady 3D flow visualization, LIC, UFLIC, texture synthesis, volume rendering*

ABSTRACT

Most texture-based 3D flow visualization techniques such as volume Line Integral Convolution (LIC) are limited to steady fields. Time-varying dense volume flow visualization is still an open problem due to intensive computation, demanding temporal-spatial coherence, and ad-hoc volume rendering. Our work is based on an enhanced Unsteady Flow LIC (UFLIC) algorithm, Accelerated UFLIC (AUFLIC), which uses a flow-driven seeding strategy and a dynamic seeding controller to reuse pathlines in the value scattering process to achieve fast time-dependent flow visualization. We first extend AUFLIC to time-varying volume flow fields for accelerated texture generation. To address occlusion, lack of depth cueing, and poor perception of flow directions within a dense volumetric texture, we employ a magnitude-based color-opacity mapping scheme in ray casting to clearly show flow structures in each frame and the flow evolution by means of a smooth animation.

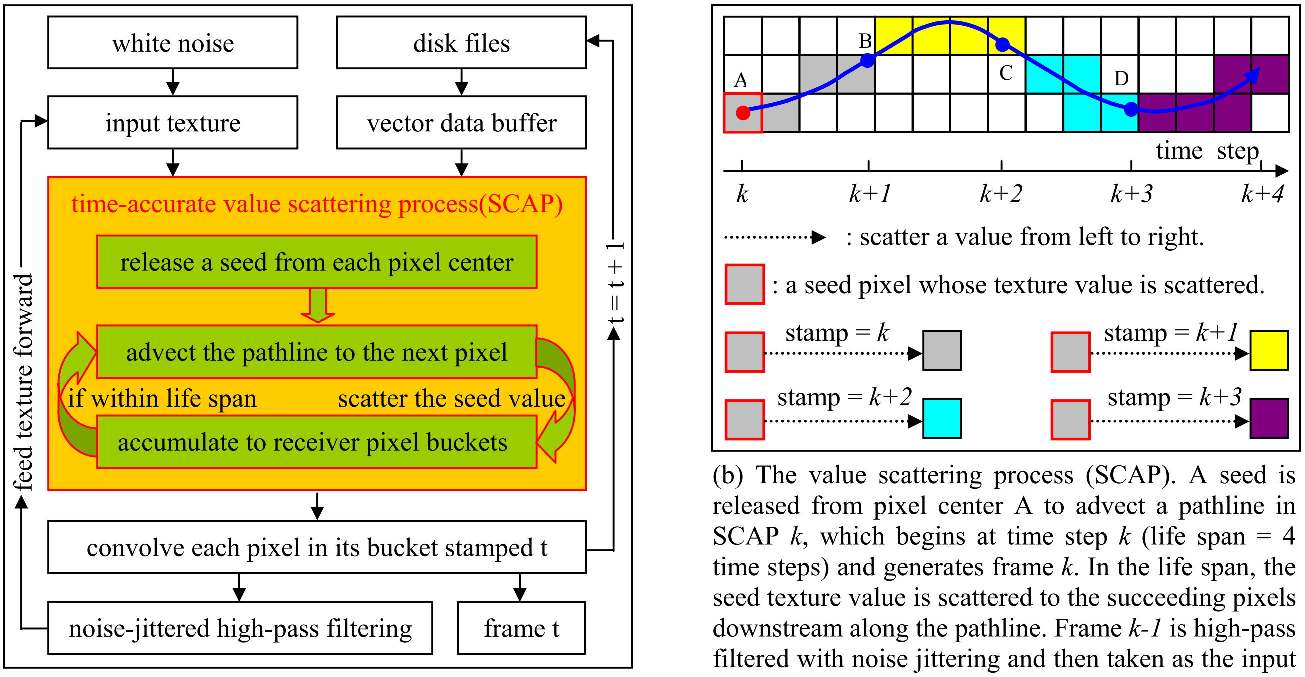
1 INTRODUCTION

Texture-based dense flow visualization techniques are increasingly important in investigating computational fluid dynamics due to their increased resolution representation and due to the decreased mental reconstruction required compared to traditional graphical-icon based methods such as arrow plots, streamlines, and stream surfaces. The traditional methods achieve either a local, discrete, coarse view or a cluttered image of the flow field. Van Wijk presented a texture synthesis technique called spot noise [1] that visualizes flow data by stretching in the flow direction a collection of oval texture splats placed within the field. Later Cabral and Leedom [2] proposed Line Integral Convolution (LIC) that convolves an input noise texture using a low-pass filter along pixel-centered symmetrically bi-directional streamlines to exploit spatial correlation in the flow direction. LIC synthesizes an image that provides a global dense representation of the flow, analogous to the resulting pattern of wind-blown sand. There have been many optimizations and extensions to LIC such as fast LIC [3], parallel LIC [4], LIC on curvilinear grids [5], LIC on triangulated surfaces [6], multi-frequency LIC [7], oriented LIC [8], enhanced LIC [9], dyed LIC [10], volume LIC [11], [12], and HyperLIC [13]. Heidrich *et al*. [14] incorporated indirect pixel texture addressing and additive / subtractive blending to accelerate streamline integration and texture convolution in LIC.

In recent years many texture based methods have been proposed to visualize time-varying flow fields using intuitively more understandable representations of the flow evolution than pathlines, streaklines [15], and flow volumes [16]. Max and Becker [17] employed texture mapping in either forward grid warping or backward texture coordinates advection for time-dependent flow visualization. Forssell and Cohen [5] used pathlines as convolution paths in LIC to visualize unsteady flow fields. Verma *et al*. [18] presented Pseudo LIC (PLIC) in which pre-synthesized template textures are mapped on sparsely placed pathline ribbons to emulate a dense flow representation. More and more methods take advantage of hardware acceleration to achieve high performance visualization. Jobard *et al*. [19], [20] developed an efficient rendering pipeline based on indirect pixel texture addressing [14] for fast texture and dye advections. Weiskopf *et al*. [21] applied pixel texture to visualize time-varying 3D vector fields. Van Wijk proposed IBFV [22] to advect a sequence of temporally-spatially low-pass filtered noise textures via forward texture mapping on warped meshes in combination with blending of successive frames to emulate many visualization techniques at interactive frame rates. IBFV was then used to visualize flow on arbitrary surfaces and enhance surface shape cueing by flow-aligned textures [23]. IBFV was even extended to 3D flows by decomposing the 3D advection to planar and longitudinal advections [24]. Independent of hardware acceleration [25], the LEA algorithm by Jobard *et al*. [26] employs backward integration to search the previous frame for the contributing particle, which scatters the texture value to the target pixel of the current frame. Temporal coherence along pathlines is created using successive texture blending while spatial coherence is improved using short-length directional low-pass filtering. Recently Laramee *et al.* [27] combined IBFV and LEA to address unsteady flow on large arbitrary triangular surfaces by projecting surface geometries to image space to which backward mesh advection and successive texture blending are applied. Weiskopf *et al.* proposed UFAC [28], which, based on a generic spacetime-coherent framework, establishes temporal coherence by property advection along pathlines while building spatial correlation by texture convolution along instantaneous streamlines.

Among the most effective algorithms, Unsteady Flow LIC (UFLIC) proposed by Shen and Kao [29] uses a time-accurate value scattering scheme and a successive texture feed-forward strategy in the pipeline (Figure 1.a) to achieve both very high temporal coherence and very strong spatial coherence. At each time step a value scattering process (*SCAP*, Figure 1.b) begins by releasing a seed from each pixel and ranges through several time steps (i.e., *life span*), in which a pathline is advected for each seed to scatter its texture value to the downstream pixels along the trace. The values received by each pixel at a time step are accumulated and convolved to synthesize the corresponding frame, which, after noise-jittered high-pass filtering, is forwarded as the input texture to the next scattering process. The inconsistency between temporal and spatial patterns in IBFV [22], LEA [26], and UFAC [28] is successfully resolved by scattering fed-forward texture values in UFLIC. Value scattering along a long pathline over several time steps not only correlates a considerable number of intra-frame pixels to establish strong spatial coherence, but also correlates sufficient inter-frame pixels to build tight temporal coherence. Successive texture feed-forward constructs an even closer correlation between consecutive frames to enhance temporal coherence. Flow directions can be clearly depicted in individual images for instantaneous flow investigation while the animation is also quite smooth. To address the low performance of UFLIC, Liu and Moorhead [30] presented Accelerated UFLIC (AUFLIC) that is an order-of-magnitude faster than UFLIC and achieves interactive unsteady flow visualization by reusing pathlines in the time-consuming value scattering process.

Several texture-based techniques have been used to visualize flow on 3D curved surfaces [5], [6], [23], [27], [31] and volume flows. However, existing methods for dense volume flow visualization such as volume LIC [10], [11], [12], [32] and 3D IBFV [24] are limited to steady fields except for 3D hardware-accelerated texture advection [21] which, unfortunately, depends on programmable texture fetch and per-pixel blending only supported by nVidia GeForce 3 graphics cards. Thus time-varying dense volume flow visualization is still an open problem primarily due to intensive computation, demanding temporal-spatial coherence, and ad-hoc volume rendering. In this paper we address this problem by extending AUFLIC [30] to 3D scenarios for fast volumetric flow texture generation and employing a magnitude-based color-opacity mapping scheme in ray casting to clearly show flow structures in each frame and the flow evolution in a smooth animation.


white noise
disk files
input texture
vector data buffer
time-accurate value scattering process(SCAP)
release a seed from each pixel center
advect the pathline to the next pixel
if within life span
scatter the seed value
accumulate to receiver pixel buckets
convolve each pixel in its bucket stamped t
noise-jittered high-pass filtering
frame t
feed texture forward
t = t + 1
A
B
C
D
time step
k
k+1
k+2
k+3
k+4
: scatter a value from left to right.
: a seed pixel whose texture value is scattered.
stamp = k
stamp = k+1
stamp = k+2
stamp = k+3

(a) The UFLIC pipeline.

(b) The value scattering process (SCAP). A seed is released from pixel center A to advect a pathline in SCAP *k*, which begins at time step *k* (life span = 4 time steps) and generates frame *k*. In the life span, the seed texture value is scattered to the succeeding pixels downstream along the pathline. Frame *k-1* is high-pass filtered with noise jittering and then taken as the input texture for SCAP *k* (white noise is used for *k* = 0).

Figure 1. The UFLIC pipeline and the value scattering process.

The remaining parts of the paper are organized as follows. In section 2 we revisit the accelerated UFLIC algorithm. In section 3 we extend the algorithm to 3D flows and employ magnitude-based color-opacity mapping to ray cast dense volumetric flow textures. Results are given to show the effectiveness and efficiency. The paper concludes with a summary and directions for future work.

2 ACCELERATED UFLIC

In this section, we briefly describe the flow-driven seeding strategy and the dynamic seeding controller used in the 2D accelerated UFLIC algorithm, i.e., AUFLIC [30].

The time-accurate value-scattering process is the bottleneck of the UFLIC pipeline while a large amount of Euler pathline integration is computationally expensive due to intensive temporal-spatial vector interpolation and line segment clamping against pixels. In order to obtain a dense scattering coverage to exploit sufficient temporal-spatial coherence, UFLIC uses a conservative seeding scheme by which a seed is released from each pixel center at the first integer time step of each SCAP. Such a regular seed pattern does not take into account flow structures and particles are too dense in converging regions. Also, a pathline is always advected from scratch and is deleted when the life span expires. This simplistic use of pathlines ignores intra-SCAP and inter-SCAP correlations, which induces severe pathline redundancy. AUFLIC employs a fourth-order Runge-Kutta integrator with adaptive step size and error control in combination with cubic Hermite polynomial curve interpolation [33] to achieve faster, more accurate pathline advection, and faster line convolution (due to evenly sampling the texture) than the Euler method. The temporally and spatially flexible seeding scheme used in AUFLIC allows for a flow-aligned seed placement which substantially reduces pathline integration by copying and truncating pathlines in the same SCAP and reusing and extending pathlines between SCAPs. The dynamic seeding controller effectively implements the flow-driven seeding strategy to achieve dense value scattering by adaptively determine whether to advect, copy, or reuse a pathline for a given pixel in the SCAP. Table 1 gives a comparison of UFLIC and AUFLIC.

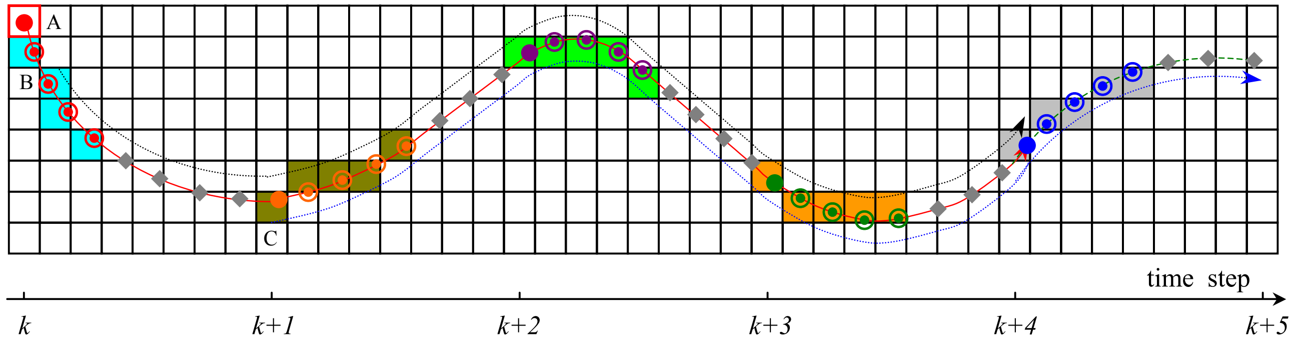
Items **Methods**		**UFLIC**	**AUFLIC**
Pathline Integration	pathline integrator	Euler (first-order)	Fourth-order Runge-Kutta (RK4)
	step size	line-segment clamp against pixels	adaptive step size
	error control	none	embedded Runge-Kutta formulae
	numerical accuracy	first-order	second-order *
	overall efficiency	slow and inaccurate	fast and accurate
Flow Driven Seeding Strategy	temporal flexibility	none (always at an integer time step)	within a fraction past a time step
	spatial flexibility	none (always from a pixel center)	an arbitrary position within a pixel
	flow-aligned	no (a lattice pattern)	yes (release seeds along pathlines)
SCAP Correlation	intra-SCAP	ignored	copy and truncate pathlines
	inter-SCAP	ignored	reuse and extend pathlines
Dynamic Seeeding Controller	seeding order	left-to-right, top-down	roughly left-to-right, top-down
	flow structure	ignored	adapt seeding to flow structures
	dense coverage	yes	yes
Line Convolution	evenly line sample	none	by cubic Hermite polynomial
	accumulated	line-segment lengths as weights	scattering hits for averaging
Result	pathline integration	a large amount	substantially reduced
	performance	low	high (1 order-of-magnitude faster)
	quality	high temporal-spatial coherence	high temporal-spatial coherence

*: As a multi-stage integrator, the RK4 method for unsteady flow advection achieves only second-order accuracy when temporal interpolation is based on the assumption that the flow varies linearly between two time steps [15].

Table 1. Comparison of UFLIC and AUFLIC.

2.1 Flow-Driven Seeding Strategy

The flow-driven seeding strategy is based on temporal and spatial seeding flexibilities which slightly decouple seed distribution with pixel-based image space and strict frame timing, but instead are geared toward physical space where the flow structure is exhibited by flow-aligned particles and toward a more continuous seed release than the intermittent mode used in UFLIC. Spatial flexibility allows a seed to be released at an arbitrary position within a pixel rather than exactly at the pixel center. Temporal flexibility allows a seed to be released within a small fractional time past the beginning of a time step (i.e., the first time step of a SCAP) rather than strictly at the very beginning of the time step point. Given the two flexibilities, seeds are released along known pathlines, if available, so that only a few of them need to actually advect pathlines; the rest can simply extract their pathlines using pathline copying and pathline reuse. *Pathline copying* is an intra-SCAP operation by which a seed's trace in the life span is used, with only minor corrections, for those of other seeds successively released at several positions downstream along the same trace at fractional times shortly after the SCAP begins as long as they are released at the same time as the initial seed travels through their seeding positions. Each of these seeds travels through a different-length part of the same curve during the first time step of the SCAP, but they synchronously run though the same trace over the remaining time steps. *Pathline reuse* is an inter-SCAP operation by which the position a pathline passes through within a fractional time into the second time step of the previous SCAP is used to release a new seed at exactly the same global time, but in the first time step of the current SCAP. This seed's trace is obtained by reusing the latter part of the known pathline from the previous SCAP, appended with integration over an additional time step. Pathline copying applies whether a pathline is obtained by reuse or brute-force integration. Figure 2 illustrates the flow-driven seeding strategy.



ξ : an arbitrary position (not necessarily a pixel center). $n+\delta$: a small fractional time past time step n.

◆ a regular sample point used to receive scattered seed values. Life span = 4 time steps.

● a seed released from a pixel center at time step k in SCAP k to advect a pathline from scratch for value scattering.

◉ a seed released from ξ at $k+\delta$ in SCAP k along the known pathline to copy and truncate the pathline.

● ● ● ● seeds released from ξ at $k+1+\delta$, $k+2+\delta$, $k+3+\delta$, $k+4+\delta$ in SCAP $k+1$, $k+2$, $k+3$, $k+4$ along the known pathline saved in SCAP k, $k+1$, $k+2$, $k+3$ respectively to reuse and extend the pathline.

◉ ◉ ◉ ◉ seeds released from ξ at $k+1+\delta$, $k+2+\delta$, $k+3+\delta$, $k+4+\delta$ in SCAP $k+1$, $k+2$, $k+3$, $k+4$ along the known pathline reused in SCAP $k+1$, $k+2$, $k+3$, $k+4$ respectively to copy and truncate the pathline.

········➤ the pathline (offset) of seed B in SCAP k that is obtained by copying and truncating that of seed A.

········➤ the pathline (offset) of seed C in SCAP $k+1$ that is obtained by reusing and extending that of seed A.

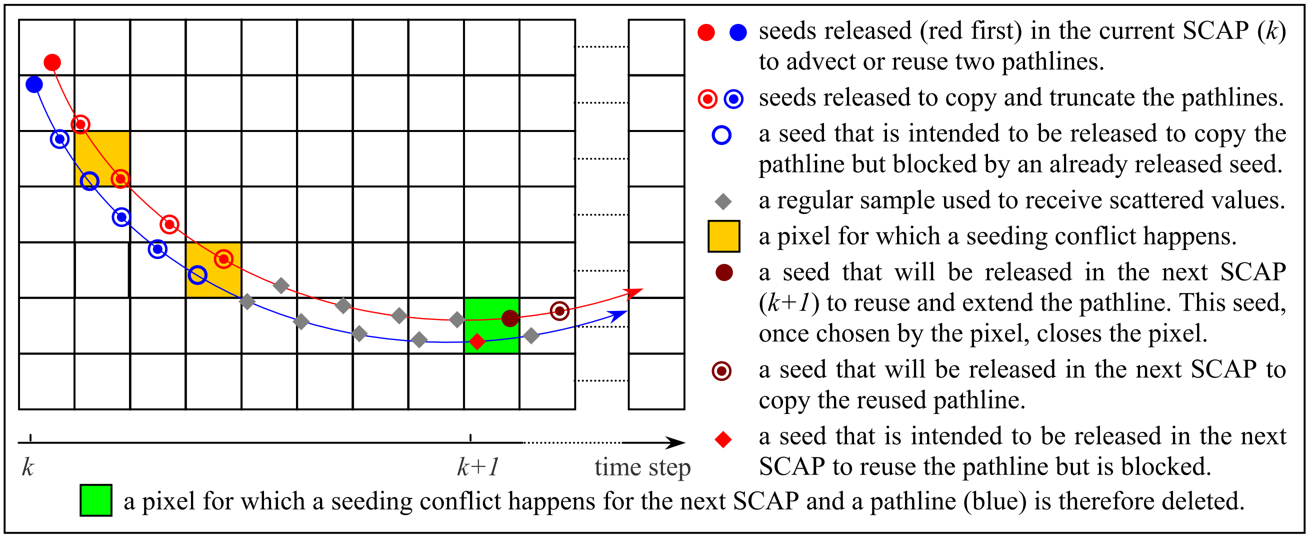
- - - - - - the extended part of the pathline that is advected in SCAP $k+1$ by seed C.

■ ■ ■ ■ ■ pixels, of which the centers are not necessarily used as in UFLIC to release seeds exactly at time step k, $k+1$, $k+2$, $k+3$, $k+4$ in SCAP k, $k+1$, $k+2$, $k+3$, $k+4$ respectively to advect new pathlines.

Figure 2. The flow-driven seeding strategy. Given the temporal and spatial seeding flexibilities, seeds released along a known pathline can copy and truncate in the same SCAP, and reuse and extend between consecutive SCAPs the pathline for fast value scattering without pathline advection.

2.2 Dynamic Seeding Controller

To implement the flow-driven seeding strategy, AUFLIC adopts a dynamic seeding controller to govern the seed distribution in a SCAP to determine for each pixel whether a pathline is advected from scratch, reused and extended, copied and truncated, saved for the next SCAP, or deleted. Without an effective controller, many seeds might be released from the same pixel in a SCAP as more and more pathlines are copied and reused while there might be no seeds released from other pixels in diverging regions. The dynamic seeding controller is used provide an adaptive, global, organized control over seed placement that prevents redundant pathline copying or reuse while maintaining a dense value scattering coverage. There is a trade off between pathline reuse and advection: the more pathlines reused, the fewer pathlines advected in the current SCAP; the fewer pathlines reused, the more pathlines advected in the next SCAP. This situation implies the computational cost could severely fluctuate over SCAPs. To obtain a nearly constant frame rate for optimal overall performance, the dynamic seeding controller seeks to balance pathline reuse and advection in each SCAP. The controller works by dynamically updating a seeding state for each pixel that either allows for a seed release (when the pixel is *open*, i.e., there is not yet a seed release) or blocks further releases (when the pixel is *close*, i.e., there is already a seed release) to ensure no more than one seed is released from the same pixel in a SCAP. Figure 3 illustrates the dynamic seeding controller.



● ● seeds released (red first) in the current SCAP (*k*) to advect or reuse two pathlines.

◉ ◉ seeds released to copy and truncate the pathlines.

○ a seed that is intended to be released to copy the pathline but blocked by an already released seed.

◆ a regular sample used to receive scattered values.

🟨 a pixel for which a seeding conflict happens.

● a seed that will be released in the next SCAP (*k+1*) to reuse and extend the pathline. This seed, once chosen by the pixel, closes the pixel.

◉ a seed that will be released in the next SCAP to copy the reused pathline.

◆ a seed that is intended to be released in the next SCAP to reuse the pathline but is blocked.

🟩 a pixel for which a seeding conflict happens for the next SCAP and a pathline (blue) is therefore deleted.

Figure 3. The seeding controller dynamically updates the states of pixels by which to determine whether a seed is released (when the pixel is *open*) or blocked (when the pixel is *close*) and whether a pathline used in the current SCAP (*k*) is saved (e.g., the red pathline) for the next SCAP (*k+1*) or just deleted (e.g., the blue pathline).

Governed by the seeding controller, a SCAP begins by reusing pathlines saved in the previous SCAP, followed by scanning the image in a left-to-right, top-to-bottom order to advect new pathlines for open pixels. Pathline copying is applied each time a pathline is reused or advected while several pixels in the opposite flow direction are reserved to enable pathline copying for right-to-left or bottom-to-top flows. The pathline, if blocked, is deleted; otherwise it is saved for use in the next SCAP.

3 VISUALIZING TIME-VARYING VOLUME FLOWS USING AUFLIC

In this section we first extend the AUFLIC algorithm to time-varying volume flows. We then adopt magnitude-based color-opacity mapping in ray casting to render dense volumetric textures to show interior flow structures.

3.1 Three-Dimensional AUFLIC

The extension of AUFLIC to volume flows is very straightforward. The flow-driven seeding strategy and the dynamic seeding controller work the same way in the volume field as in 2D scenarios. The small memory footprint of the algorithm allows AUFLIC to be used for large unsteady volume flow visualization. The major difference is the introduction of volume rendering in the pipeline to display generated volumetric textures using 2D images (Section 3.2). A white noise volume is typically used as the initial input texture. Sparse noise textures effectively used in steady volume flow visualization [32] can not be directly used in 3D AUFLIC for time-varying flows since the iteratively advected "empty" space may not follow the evolution of the flow, resulting in discontinuous flow lines. We implemented 3D UFLIC, 3D AUFLIC, and compared them on Dell Inspiron 2650 notebook PC running Windows XP (Mobile Pentium IV 1.70 GHz, 512MB RAM). Table 2 shows the time breakdown of the two algorithms for generating 41 volumetric textures from a time-varying volume flow dataset (144 × 73 × 81). The test shows 3D AUFLIC is nearly 5 times faster than 3D UFLIC.

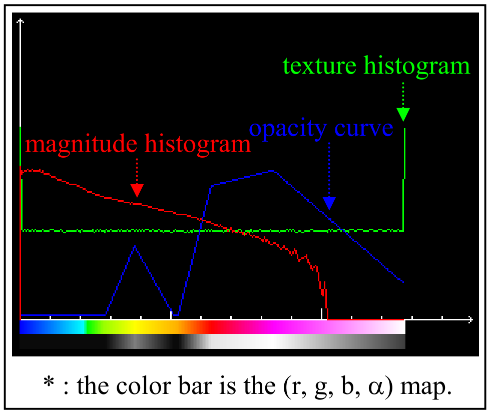
Items Methods	3D UFLIC	3D AUFLIC
Data loading	41.57	40.96
Value scattering	2830.47	552.58
Convolution	4.83	4.84
Noise-jittered high pass filtering	19.12	18.83
Texture output	5.01	4.96
Total	2901.01	622.17
Acceleration ratio	4.70	

Table 2. The time breakdown of 3D UFLIC and 3D AUFLIC in comparison (in second).

3.2 Magnitude-Based Color-Opacity Mapping

Volumetric textures generated by 3D AUFLIC need to be displayed using volume rendering. Volume rendering techniques such as ray casting [34], ray tracing [35], splatting [36], shear-warp [37], and hardware-based texture mapping [38] eliminate the construction of intermediate geometric representations that is needed in iso-surface extraction [39], but instead operate directly on volumetric elements (voxels) by using a light absorption-transmission model and a transfer function to assign colors and opacities to voxels that are then composited along the view direction. Volume rendering has been extensively and successfully used in medical data visualization, though its application in rendering volume flows, particularly dense LIC texture volumes, is still limited due to occlusion, poor depth cueing, and most importantly lack of any physical meaning of a texture value. Without an effective transfer function, the interior flow structure and spatial orientation could not be revealed to provide an insight into the dense volume.

The histogram of a volume AUFLIC texture (Figure 4) offers no guidance to transfer function design since it does not convey information provided by, e.g., that of a volume medical data which can be used to distinguish between different components such as bones and soft tissues. Fortunately, AUFLIC texture values exhibit temporal-spatial correlation along flow lines and hence can be used to compute gradients for use as normals in Phong shading. On the other hand, velocity magnitude, an important flow attribute, can be used to enhance or suppress certain parts of the flow. Thus we employ magnitude-based color-opacity mapping (Figure 4) to design transfer functions for volume rendering. It is similar to but different from the bi-variate volume rendering model [10] which is used to render the mixture of two volumes. Figure 5 shows the rendering pipeline of 3D AUFLIC textures. Figure 6 shows two images generated using this method. Interior flow structures can be clearly revealed by tuning the magnitude-based transfer function while the flow evolution is shown by means of a smooth animation (see accompanying movies).



* : the color bar is the (r, g, b, α) map.

Figure 4. Magnitude-based mapping.

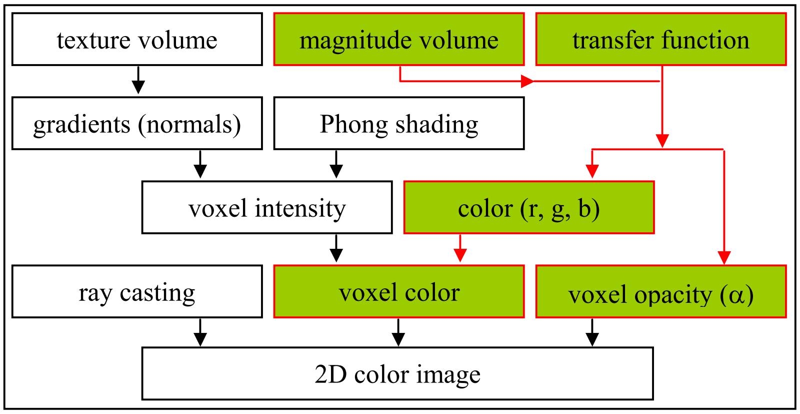


Figure 5. The rendering pipeline of 3D AUFLIC textures.

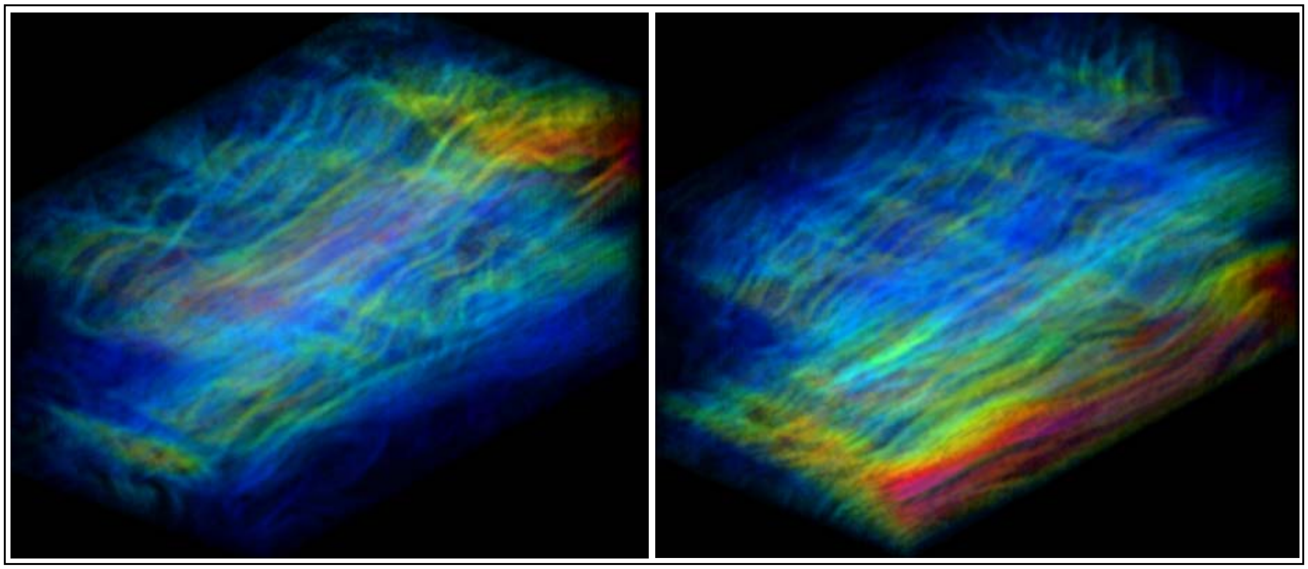


Figure 6. Two images of a time-varying volume flow generated by using magnitude-based color-opacity mapping in ray casting.

4 CONCLUSIONS AND FUTURE WORK

Accelerated UFLIC uses a flow-driven seeding strategy and a dynamic seeding controller to reuse pathlines in the value scattering process to achieve fast time-dependent flow visualization with very high temporal-spatial coherence. We have extended this algorithm for time-varying volume flows and employed magnitude-based color-opacity mapping to ray cast dense volumetric textures. Interior flow structures can be investigated by adjusting the mapping function and the flow evolution can be shown in the smooth animation. Future work includes further accelerating AUFLIC by using shorter pathlines while maintaining high temporal-spatial coherence. Another direction is to improve volumetric flow texture rendering by using, e.g., ROI masking, geometric clipping, and visibility-impeding 3D halos [32].

ACKNOWLEDGMENTS

This work was supported by DoD HPCMP and NSF grant EPS-0132618.

REFERENCES

1. van Wijk, J. J. Spot Noise: Texture Synthesis for Data Visualization. *Computer Graphics*, Vol. 25, No. 4, pp.309-318, 1991.
2. Cabral, B. and Leedom, L. C. Imaging Vector Fields Using Line Integral Convolution. *Proceedings of ACM SIGGRAPH 93*, Anaheim, California, Aug 2-6, pp.263-270, 1993.
3. Stalling, D. and Hege, H.-C. Fast and Resolution Independent Line Integral Convolution. *Proceedings of ACM SIGGRAPH 95*, Los Angeles, California, Aug 6-11, pp.249-256, 1995.
4. Stalling, D., Zockler, M., and Hege, H.-C. Parallel Line Integral Convolution. *Proceedings of the First Eurographics Workshop on Parallel Graphics and Visualization*, Bristol, U.K., Sept 26-27, pp.111-128, 1996.
5. Forssell, L. K. and Cohen, S. D. Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation, and Unsteady Flows. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1, No. 2, pp.133-141, 1995.
6. Teitzel, C., Grosso, R., and Ertl, T. Line Integral Convolution on Triangulated Surfaces. *Proceedings of the Fifth International Conference in Central Europe on Computer Graphics and Visualization (WSCG97)*, Plzen-Bory, Czech Republic, Feb 10-14, pp.572-581, 1997.
7. Kiu, M.-H. and Banks, D. C. Multi-Frequency Noise for LIC. *Proceedings of IEEE Visualization 96*, San Francisco, California, Oct 27–Nov 1, pp.121-126, 1996.
8. Wegenkittl, R., Groller, E., and Purgathofer, W. Animating Flow Fields: Rendering of Oriented Line Integral Convolution. *Proceedings of Computer Animation 97*, Geneva, Switzerland, June 5-6, pp.15–21, 1997.
9. Okada, A. and Kao, D. L. Enhanced Line Integral Convolution with Flow Feature Detection. *Proceedings of IS & T / SPIE Electronics Imaging 97*, San Jose, California, Feb 8-14, Vol. 3017, pp.206-217, 1997.
10. Shen, H.-W., Johnson, C., and Ma, K.-L. Visualizing Vector Fields using Line Integral Convolution and Dye Advection. *Proceedings of IEEE Symposium on Volume Visualization 96*, San Francisco, California, Oct 28-29, pp.63-70, 1996.
11. Interrante, V. and Grosch, C. Strategies for Effectively Visualizing 3D Flow with Volume LIC. *Proceedings of IEEE Visualization 97*, Phoenix, Arizona, Oct 19–24, pp.421-424, 1997.
12. Rezk-Salama, C., Hastreiter, P., Teitzel, C., and Ertl, T. Interactive Exploration of Volume Line Integral Convolution Based on 3D-Texture Mapping. *Proceedings of IEEE Visualization 99*, San Francisco, California, Oct 24-29, pp.233-240, 1999.
13. Zheng, X. and Pang, A. HyperLIC. *Proceedings of IEEE Visualization 03*, Seattle, Washington, Oct 19-24, pp.249-256, 2003.
14. Heidrich, W., Westermann, R., Seidel, H.-P., and Ertl, T. Applications of Pixel Textures in Visualization and Realistic Image Synthesis. *Proceedings of ACM Symposium on Interactive 3D Graphics*, Atlanta, Georgia, April 26-28, pp.127-134, 1999.
15. Lane, D. A. Scientific Visualization of Large-Scale Unsteady Fluid Flows. *Scientific* Visualization (Editors: Nielson, G. M., Hagen, H., and Muller, H.), IEEE Computer Society Press, Los Alamitos, California, pp125-145, 1997.
16. Becker, B. G., Lane, D. A., and Max, N. L. Unsteady Flow Volumes. *Proceedings of IEEE Visualization 95*, Atlanta, Georgia, Oct 29-Nov 3, pp.329-335, 1995.
17. Max, N. L. and Becker, B. Flow Visualization Using Moving Textures. *Data Visualization Techniques* (Editor: Bajaj, C.), John Wiley and Sons Ltd., pp.99-105, 1999.
18. Verma, V., Kao, D. L., and Pang, A. PLIC: Bridging the Gap Between Streamlines and LIC. *Proceedings of IEEE Visualization 99*, San Francisco, California, Oct 24-29, pp.341-348, 1999.

19. Jobard, B., Erlebacher, G., and Hussaini, M. Y. Hardware-Assisted Texture Advection for Unsteady Flow Visualization. *Proceedings of IEEE Visualization 00*, Salt Lake City, Utah, Oct 8–13, pp.155-162, 2000.
20. Jobard, B., Erlebacher, G., and Hussaini, M. Y. Tiled Hardware-Accelerated Texture Advection for Unsteady Flow Visualization. *Proceedings of the 10th International Conference on Computer Graphics & Vision (Graphicon 2000)*, Moscow, Russia, Aug 28-Sept 2, pp.189-196, 2000.
21. Weiskopf, D., Hopf, M., and Ertl, T. Hardware-Accelerated Visualization of Time-Varying 2D and 3D Vector Fields by Texture Advection via Programmable Per-pixel Operations. *Proceedings of the 6th International Fall Workshop on Vision, Modeling, and Visualization (VMV 2001)*, Stuttgart, Germany, Nov 21-23, pp439-446, 2001.
22. van Wijk, J. J. Image Based Flow Visualization. *Proceedings of ACM SIGGRAPH 02*, San Antonio, Texas, July 21-26, pp.745-754, 2002.
23. van Wijk, J. J. Image Based Flow Visualization for Curved Surfaces. *Proceedings of IEEE Visualization 03*, Seattle, Washington, Oct 19-24, pp.123-130, 2003.
24. Telea, A. and van Wijk, J. J. 3D IBFV: Hardware-Accelerated 3D Flow Visualization. *Proceedings of IEEE Visualization 03*, Seattle, Washington, Oct 19-24, pp.233-240, 2003.
25. Weiskopf, D., Erlebacher, G., Hopf, M., and Ertl, T. Hardware-Accelerated Lagrangian-Eulerian Texture Advection for 2D Flow Visualization. *Proceedings of the 7th International Fall Workshop on Vision, Modeling, and Visualization (VMV 2002)*, Erlangen, Germany, Nov 20-22, pp.77-84, 2002.
26. Jobard, B., Erlebacher, G., and Hussaini, M. Y. Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 8, No. 3, pp.211-222, 2002.
27. Laramee, R. S., Jobard, B., and Hauser, H. Image Space Based Visualization of Unsteady Flow on Surfaces. *Proceedings of IEEE Visualization 03*, Seattle, Washington, Oct 19-24, pp.131-138, 2003.
28. Weiskopf, D., Erlebacher, G., and Ertl, T. A Texture-Based Framework for Spacetime-Coherent Visualization of Time-Dependent Vector Fields. *Proceedings of IEEE Visualization 03*, Seattle, Washington, Oct 19-24, pp.107-114, 2003.
29. Shen, H.-W. and Kao, D. L. New Line Integral Convolution Algorithm for Visualizing Time-Varying Flow Fields. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 4, No. 2, pp.98-108, 1998.
30. Liu, Z. and Moorhead, R. J. Accelerated Unsteady Flow Line Integral Convolution. *IEEE Transactions on Visualization and Computer Graphics*, 2004 (to appear).
31. Mao, X., Kikukawa, M., Fujita, N., and Imamiya, A. Line Integral Convolution for Arbitrary 3D Surfaces Through Solid Texturing. *Proceedings of the 8th EuroGraphics Workshop on Visualization in Scientific Computing*, Boulogne-Sur-Mer, France, April 28-30, pp.57-69, 1997.
32. Interrante, V. and Grosch, C. Visualizing 3D Flow. IEEE Computer Graphics and Applications. Vol. 18, No. 4, pp.49-53, 1998.
33. Hege, H.-C. and Stalling, D. LIC: Acceleration, Animation, and Zoom. *Texture Synthesis with Line Integral Convolution (Course #8), ACM SIGGRAPH 97 Conference*, Los Angeles, California, Aug 3-8, pp.17-49, 1997.
34. Levoy, M. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, Vol. 8, No. 5, pp.29-37, 1988.
35. Levoy, M. Efficient Ray Tracing of Volume Data. *ACM Transactions on Graphics*, Vol. 9, No. 3, pp.245-261, 1990.
36. Westover, L. Footprint Evaluation for Volume Rendering. *Computer Graphics*, Vol. 24, No. 4, pp.367-376, 1990.
37. Lacroute, P. and Levoy, M. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. *Proceedings of ACM SIGGRAPH 94*, Orlando, Florida, July 24-28, pp.451-458, 1994.
38. Cabral, B., Cam, N., and Foran, J. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. *Proceedings of ACM / IEEE Symposium on Volume Visualization 94*, Washington DC, Oct, pp.91-98, 1994.
39. Lorensen, W. E. and Cline H. E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, Vol. 21, No. 4, pp.163-169, 1987.