Dense Geometric Flow Visualization

Sung W. Park* Oliver Kreylos* Bernd Hamann*

1 Introduction

Flow visualization has a long tradition in scientific data visualization. Approaches for 3D vector fields however have only recently experienced a boost due to the introduction of programmable graphics hardware with large texture memory. Consequently, volumetric flow visualization has entered many disciplines of science and engineering including mechanics, physics, chemistry, meteorology, geology, and medicine. Many applications are concerned with steady or unsteady flow over 2D or 3D domains. By adopting ideas from texture-based techniques and taking advantage of parallelism and programmability of contemporary graphics hardware, we introduce a dense geometric flow visualization technique based on streamlines and pathlines addressing both steady and unsteady flow. The lines are computed using a trajectory-based particle-advection method. We achieve high numerical accuracy by enforcing short particle lifetimes and employing a fourth-order integration method. Along with line integration, we apply rendering techniques such as streamline illumination, mutli-dimensional transfer functions (MDTFs)[Park et al. 2004], and haloing all in the graphics hardware to enhance visualization and achieve interactivity.

2 Related Work

Flow visualization approaches have been well documented in recent surveys [Laramee et al. 2004; Post et al. 2002; Post et al. 2003] and can be generally categorized into direct, geometric, texture-based, and feature-based approaches.

Early attempts such as arrow and hedgehog plots or color coding fall into the category of direct flow visualization [Post et al. 2002]. They provide an intuitive image of local flow properties. For a better understanding of global flow dynamics with respect to "long-term" behavior, integration-based approaches have been introduced. These integrate flow data leading to trajectories of no-mass particles moving over time. Geometric flow visualization approaches render the integrated flow using geometric objects such as lines, tubes, ribbons, or surfaces [Post et al. 2002].

In texture-based flow visualization, a texture is used for a dense representation of a flow field. The texture is filtered according to the local flow vectors leading to a notion of overall flow direction [Laramee et al. 2004]. Feature-based flow visualization is concerned with the extraction of specific patterns of interest, or features. Various features such as vortices, shock waves, or separatrices have been considered. Once a feature has been extracted, standard visualization techniques are applied for rendering [Post et al. 2003].

All these approaches work well for 2D vector fields. While geometric flow visualization approaches generalize to volume data they are computationally expensive to achieve dense, interactive visualization. Direct and texture-based approaches have occlusion issues in 3D, an inherent problem for dense representations. Dense representations are desirable, as they provide information concerning overall flow behavior and serve as a context for chosen visualization methods.

*Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, University of California, Davis;{parksw—kreylos—hamann}@cs.ucdavis.edu

3 Dense Geometric Flow Visualization

The concept of dense geometric flow visualization is adopted from texture-based flow visualization approaches. A rendering primitive is advected over time under the influence of the underlying flow field. In each time frame, a flow integration step is applied to determine the position of the primitive in the subsequent frame. The motion of the primitive indicates local flow behavior. The motion of a large number of primitives distributed densely across the domain visualizes the behavior of the entire flow field.

While for texture-based approaches a rendering primitive is represented by color information, for example in form of noise, our approach uses geometry. Instead of advected texels, we advect particles and connect them over subsequent frames to form streamlines or pathlines.

Let $\mathbf{f} : D \rightarrow R$ be the function of a steady flow field with a 2D or 3D domain D and a vector-valued range R. To visualize steady flow we render a large number n of streamlines distributed densely and nearly uniformly over the domain D. At each point in time (after a start-up phase), exactly n streamlines exist. Each streamline has a constant lifetime of k cycles. The lifetime of a streamline is divided into three phases: (i) seeding, (ii) advection, and (iii) expiration. The seeding and the expiration phases last only one cycle each, such that for the major part of its lifetime, the other $k-2$ cycles, a streamline is in its advection phase.

The streamlines are grouped into k groups of size $\frac{n}{k}$, where n is chosen to be a multiple of k. All streamlines of each group start their life simultaneously. The computation of all streamlines can be processed in a pipeline with k cycles. Figure 1 shows the processing pipeline.

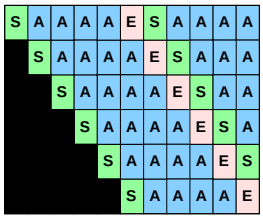

Figure 1: Processing pipeline for streamline rendering. Lifecycle of streamlines is divided into phases seeding (S), advection (A), and expiration (E).

4 Implementation

We implement the entire processing pipeline in the programmable graphics hardware. For particle seeding, we have implemented a Halton sequence generator in the vertex shader to not reduce the efficiency of the pipelining process using the fragment shader. As input, we stream a set of vertices, where each vertex holds a 2D index corresponding to a texel in a 2D texture. Each vertex streamed

into the vertex shader generates two or three Halton numbers for 2D or 3D flow, respectively, and passes the fragment shader value. At the end of this stage, the rendered texture contains the location of a seed particle in each texel.

The advection component of the algorithm takes as input a set of particle positions, and generates a corresponding set of advected particles. A single advection is performed by first sampling each texel of the texture that holds the vector field at time t in the fragment shader. For each particle's position the vector field texture is looked up. The vector field texture is sampled three more times at different locations based on the particle's position according to the fourth-order Runge-Kutta integration method. Finally, the advected particles are written to a new texture, which is used for both rendering and further advection in the following time steps.

Rendering is performed by connecting particle positions over time using line primitives. We use a buffer that holds positions of each seed in its k cycles of life. The rendering component uses as input the result of the last particle advection. The advected particle positions are first extracted from the texture. Then, the particles are added into the buffer in its proper cycle. The line segments are then sorted based on its depth from the viewer using bitonic sort algorithm and finally rendered with enhancements such as line-illumination, haloing, and MDTFs.

5 Results

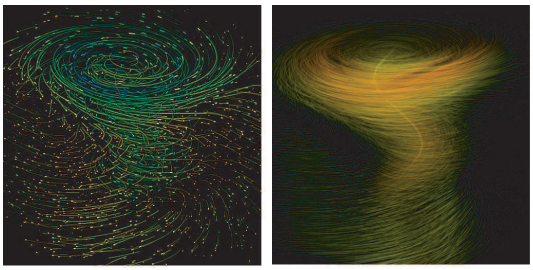

Figure 2: Tornado: Streamlines are rendered using (left) haloing (right) depth sorting and illumination in conjuction with depth-based attenuation.

We examined a tornado data set of size 128^3 and a CFD simulation of five jets consisting of 2000 timesteps of 128^3 vector field data. Figure 2 shows a dense geometric flow visualization for the tornado data set using streamlines. In Figure 2 (left), we have rendered polylines using flow orientation, depth-based attenuation, and haloing. Figure 2 (right) shows the importance of using illumination and depth sorting. The visual perception has increased significantly. In Figure 3, we have used an MDTF that extracts regions of high curl (or vorticity) magnitude. The streamlines are only drawn within the vortex area.

The computation times for our rendering depend on the rendering techniques used and on the number of streamlines or pathlines, respectively. For steady flow, we have achieved frame rates up to 66 and for unsteady data when rendering pathlines, we have achieved frame rates up to 10 frames per second. Contrary to texture-based flow visualization approaches, the frame rates achieved when using dense geometric flow visualization are (almost) independent of the size of the data set. Thus, for larger data sets that still fit in the texture memory, frame rates are close to the ones listed above. For unsteady data, data transfer, i. e. copying the individual time steps to the GPU, becomes a bottle-neck.

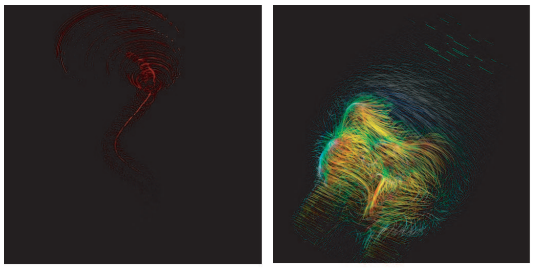

Figure 3: Tornado (left) and Time step 1500 of jets data set(right): MDTFs extract regions of high curl magnitude and high velocity magnitude.

6 Conlclusion and Future Work

We have presented a flow visualization approach based on rendering geometry in a dense, uniform distribution. We have integrated flow using particle advection to generate streamlines (for steady flow) and pathlines (for unsteady flow). Pipelining is used to manage seeding, advection, and death of streamlines/pathlines with constant lifetime. Our method uses a fourth-order Runge-Kutta method, which has been efficiently implemented in hardware by exploiting parallelism and programmability of graphics hardware. Geometry is rendered using several techniques to enhance visual perception. We have applied our approach to steady and unsteady 3D flow fields achieving interactive frame rates.

Future research efforts will be directed at employing a depth-sorted haloing strategy, using adaptive time steps for flow integration, experimenting with different types of geometry (requires programmability of the graphics card's geometry engine), and extending the approach to unstructured and irregularly grid-structured data.

References

LARAMEE, R. S., HAUSER, H., DOLEISCH, H., VROLIJK, B., POST, F. H., AND WEISKOPF, D. 2004. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum 23*.

PARK, S., BUDGE, B., LINSEN, L., HAMANN, B., AND JOY, K. I. 2004. Multi-dimensional transfer functions for interactive 3d flow visualization. In *Proceedings of The 12th Pacific Conference on Computer Graphics and Applications - Pacific Graphics 2004*, D. Cohen-Or, H.-S. Ko, D. Terzopoulos, and J. Warren, Eds.

POST, F. H., LARAMEE, R. S., VROLIJK, B., HAUSER, H., AND DOLEISCH, H. 2002. Feature extraction and visualisation of flow fields. In *Eurographics 2002, State of the Art Reports*, IEEE Computer, D. Fellner and R. Scopigno, Eds., The Eurographics Association, 69–100.

POST, F. H., VROLIJK, B., HAUSER, H., LARAMEE, R. S., AND DOLEISCH, H. 2003. The state of the art in flow visualization: Feature extraction and tracking. *Computer Graphics Forum 22*, 4, 775–792.