High-quality Texture-based Flow Visualization on Surfaces

Dibin Zhou, Kangjian Wang, Yao Zheng, Yanni Wang
College of Computer Science and Center for Engineering and Scientific Computation, Zhejiang University, Hangzhou, Zhejiang, 310027, China
dibin.z@163.com

Abstract

In this paper, we present a high-quality texture-based approach for the visualization of vector fields on surface. It adopts a scheme of streamline enhancement by applying a 1D high-pass filter to flow textures along the orthogonal flow direction to enhance the intensity contrast among streamlines and improve the image quality of flow textures. The scheme of edge detection and supplement is used in our approach to modulate the characteristic of particles flow. Our algorithm is image space based and can achieve a real-time rendering performance on a personal computer by using parallel processing ability of current graphics hardware.

1. Introduction

Flow visualization is one of the fundamental tasks in scientific visualization, especially in hydrodynamics, aerodynamics, weather forecast, oceanography and computational mechanics. Traditional techniques, such as displaying particle traces (streamlines, pathlines, timelines and streaklines), can only reveal flow phenomena in preselected local regions, and thus they are unable to describe the global flow features. Critical flow features will be missed if these important regions are not covered by any particle path. This problem can be overcome by a dense representation, i.e., by covering the domain densely with particle traces.

Interactive texture-based flow visualization has become an active field of research in the latest years, which has the advantage of revealing the entire flow field, unlike many velocity probes that provide information at only one point or along one line. In early days, the texture-based algorithms are mainly implemented on CPU and heavy convolution computation makes early research work focus on convolution optimization. To achieve real-time rendering performance, they often recur to parallel systems or special graphics hardware. Recently, with the rapid development of commodity graphics hardware and the advent of programmable processors, some algorithms can be implemented on GPU and easily achieve real-time rendering performance on a personal computer.

Although rendering performance is remarkably improved for current texture-based methods, some of them are not high spatial-correlated due to unclear flow lines in the convolution texture and low intensity contrast of streamlines. The intensity distribution of the resulting representation concentrates in a very narrow region. Some algorithms utilize 2D high-pass filter to enhance flow lines but ignore the directional characteristic of the resulting representation. In this paper, we present a high-quality texture-based approach to visualize vector fields on surfaces which can produce animations with high spatial-temporal coherence at interactive rates. It adopts a scheme of streamline enhancement by 1D high-pass filtering flow textures along the orthogonal flow direction, which enhances intensity contrast among streamlines and improves the image quality. Compared with 2D high-pass filter, 1D filter is more efficient due to fewer texture lookups and doesn’t blur the flow image along streamlines. The scheme of edge detection and supplement is used in our approach to modulate the characteristic of particles flow.

The remainder of this paper is organized as follows. In Section 2, we relate our work to previous research and focus on GPU-based dense visualization methods; Section 3 describes our high-quality algorithm and emphasizes particles advection and streamline enhancement. In Section 4, high-quality images

Supported by the NSFC (National Natural Science Foundation of China), for the National Science Fund for Distinguished Young Scholars under grant Number 60225009, and the Major Research Plan under grant Number 90405003.

978-1-4244-1579-3/07/$25.00 © 2007 IEEE
generated by the algorithm are provided, and the real-time interactivity of the algorithm is demonstrated on a PC. The paper is concluded in Section 5, where some proposals for future work are also presented.

2. Related work

Nowadays, many texture-based flow visualization algorithms can be accelerated by using of parallel processing ability of graphics hardware. Heidrich et al. [1] exploit pixel textures to accelerate LIC computation, but their method only supports 2D steady vector fields. Jobard [2] et al. introduced a GPU-assisted texture advection technique for the dense visualization of 2D unsteady flow. Van Wijk presented a new technique of flow visualization called Image Based Flow Visualization (IBFV [3]). It can achieve a high performance by using standard features of graphics hardware such as texture mapping and fragment blending, and simulate other techniques, such as arrow plots, streamlines, particles and topological images. Later IBFV was extended to visualize flow on 3D curved surfaces and enhance surface shape cuing by means of flow-aligned textures (IBFVS [4]). It was even extended to 3D flows by decomposing the 3D advection to planar and longitudinal advections 3D space (3D IBFV [5]). Jobard et al. presented a LEA [6] algorithm to visualize unsteady flow fields at high frame rates despite its independence of hardware acceleration.

Recently, Laramee et al. presented a novel method ISA [7] (Image Space Advection) based on IBFV and LEA for a direct dense representation of unsteady flows on arbitrary triangular surfaces to address large, unstructured, dynamic meshes by projecting the surface geometry to image space to which backward mesh advection and successive texture blending are applied. Grabner and Laramee [8] described ISA implementation on GPU, including both the advantages and disadvantages of implementing ISA on the GPU. Weiskopf et al. introduced a generic texture-based framework for visualizing 2D unsteady vector fields. They proposed Unsteady Flow Advection Convolution (UFAC [9]) as an application of the framework for visualizing unsteady flow. Shen and Kao presented a hardware-independent algorithm called UFLIC (Unsteady Flow LIC) which uses a time-accurate value scattering scheme and a successive texture feed-forward strategy to achieve very high temporal and spatial coherence. Later, Liu and Moorhead [10] presented an accelerated UFLIC algorithm (AUFLIC) where a dynamic seeding controller was designed to intelligently distribute seeds for dense coverage and better overall performance at a very low memory cost. Later, Li [11] presented an efficient and accurate implementation of UFLIC on GPU and called it GPUFLIC.

3. Our algorithm

Fig. 1 shows the workflow of high-quality texture-based flow visualization on surfaces. Firstly, vectors on a surface are projected onto screen space and then mapped into a 2D vector texture generated by graphics hardware. Secondly, particles advection is implemented on fragment processors by applying LIC convolution on the 2D property texture of particles (gray image). Thirdly, noise is injected and blended to the convolution texture and 1D high-pass filter is applied to the convolution texture to enhance flow texture and improve its image quality. The enhanced texture can be both used as input property texture for next iteration and for the final display. The final display can combine surface shading or scalar properties of the flow, such as velocity magnitude, orientation and curvature.

The input property texture for the first convolution is a 2D random noise texture which can be looked upon as massless particles with some properties. The high-quality characteristic of our algorithm lies on the spatial-temporal coherence. Spatial coherence within a single image frame reveals image patterns indicating the flow structure of a vector field. It can be achieved by LIC convolution on property texture and enhanced by a 1D high-pass filtering. Temporal coherence allows the user to identify a consistent motion of these patterns. It is achieved by iterative convolution on property texture. Streamline enhancement is introduced as post-processing of LIC to improve the
image quality. These parts of our algorithm are detailed as follows, including surface vector projection, streamline enhancement (the 1D high-pass filtering), texture convolution, edge detection and supplement.

### 3.1. Vector projection on surfaces

The projection of surface vectors consists of two parts. One is projection vectors on a surface to avoid influence of the changed viewpoint; the other is projection vectors on screen space to produce a 2D texture of vector field supported by the graphics hardware. The first part can be pre-calculated only once before rendering while the second need be calculated each time with the changed viewpoint.

[Diagram showing projection vectors on a surface]

Fig. 2. Projection vectors on a surface

Fig. 2 shows projection vectors on a surface. \( N \) is an unit normal at point \( O \) of the surface, \( V \) is a vector at point \( O \) and \( V_p \) is its projection on the surface. If the projection process is neglected, a problem occurs when viewpoint is slight changed from point \( A \) to \( B \). The vector \( V \) is projected to \( OA' \) for the viewpoint at point \( A \) while projected to \( OB' \) for the viewpoint at point \( B \). The vector \( OA' \) and \( OB' \) are with the reverse direction, thus it is hard to identify the \( V \) direction on the surface. We can solve this problem by projection \( V \) to \( V_p \) at first. The \( V_p \) can be calculated as follows:

\[
V_p = V - (N \cdot V) N
\]  

### 3.2. Streamline enhancement

The streamline enhancement is important for appropriate particles advection and therefore introduced at first, particles advection including edge detection and supplement will be introduced in the next section.

At present, some algorithms display a line integral convolution texture without post-processing, such as ISA and IBFV, and result in low quality texture images. Other algorithms adopt a 2D high-pass filter to extrude flow lines, such as UFLIC. Considering that the 2D high-pass filter is non-directional and may enhance intensity contrast of the interior streamlines to a certain extent, we adopt the 1D high-pass filtering as the post-processing of our algorithm.

Fig. 3 shows the filtering procedure in a vector field. \( V \) denotes the tangent direction of a streamline at point \( P_c \), \( P_s1P_s2 \) is perpendicular to \( V \), and point \( P_s1 \) and \( P_s2 \) are two sample points. The filter kernel is a 1D matrix of \([-1, 3, -1]\). The sample distance \( P_s1P_c \) or \( P_s2P_c \) depends on the magnitude of the vector \( V \), as shown below.

\[
Len = (1 - \alpha) ||V|| + \alpha
\]  

Where, \( ||V|| \) denotes the vector magnitude, \( Len \) is sample distance, and \( \alpha \) is a parameter used to adjust the sample distance and range from 0.0 to 1.0. It can be seen that sample distance increases with the vector magnitude increases from formula (2). The coordinates of the two sample points \( P_s1 \) and \( P_s2 \) can be calculated by formula (3).

\[
P_s = P_c \pm Len \cdot V \perp
\]

Where, \( V \perp \) denotes a unit vector perpendicular to the streamline, \( P_c \) is the position of the center point and \( P_s \) is the position of a sample point.

[Diagram showing the illustration of 1D high-pass filtering]

Fig. 3. The illustration of 1D high-pass filtering

Fig. 4 shows two convolution results. The frame (a) is the result of FastLIC [12], where the shading intensity is more even because LIC is a low-pass filter along streamlines in fact which reduces the intensity...
contrast of streamlines. The frame (b) is a result of our algorithm with the post-processing of 1D high-pass filtering and a backward particles advection. It can be seen that the frame (b) have more clear streamlines, the contrast of the intensity is evident among streamlines. Thus the vector field is much easier to be observed.

Fig. 5 shows the influence of different alpha value on the LIC textures and interior regions in the black rectangle are zoomed in on the bottom frames. When the sample distance parameter alpha is 0.0, sample distance is linear with magnitude. In the frame (a), the center region A is blurred without clear streamlines and it may be due to the small magnitudes of vectors and the low sample distance, while the region B is slight blurred and dim. When the alpha is 0.3, the texture is better than (a), with clear streamlines and high intensity contrast as shown in the frame (b), but some regions are still not good, such as region C with uneven streamlines and slight blurred region D. When the alpha is 0.5, the image quality is improved than before as shown in the frame (c) except little uneven streamlines in region E. When the alpha is 0.8, the image quality somewhat falls as shown in the frame (d), and region F becomes blurred and streamlines of region G becomes uneven and discontinuous. When the alpha is 1.0, the image quality falls quickly as shown in the frame (e). The image in region H is more blurred and the granularity of the streamline in region I is more uneven. Usually, the alpha value should be adjusted for different vector fields. The appropriate alpha value is in the range of 0.3-0.5 in which the streamlines are most clear and image quality is highest.

3.3. Particles advection

As seen in Section 3.2, the 1D high-pass filtering can remarkably enhance the intensity contrast among streamlines and improve image quality. In this section, we will discuss appropriate particles advection method. LIC is a widely used convolution method to represent particles advection, which can introduce spatial coherence among these particles. The convolution formula is shown as follows.

\[ I(x_0) = \int_{s_0}^{s_0+L} k(s-s_0)T(\sigma(s)) \, ds \]  

(4)

Where, \( K(s) \) is the convolution kernel, \( T(x) \) denotes the property texture (particles), \( \sigma(S) \) denotes the streamline, initial condition is \( x_0=\sigma(S_0) \). The convolution can be calculated by backward and forward symmetrical convolution with length \( L \) along the streamline \( \sigma(S) \).

Considering that streamlines and path-lines aren’t coincident for unsteady vector fields, we adopt the backward LIC convolution as shown in formula (5).

\[ I(x_0) = \int_{s_0-L}^{s_0} k(s-s_0)T(\sigma(s)) \, ds \]  

(5)

The most common convolution kernels are box, triangle and exponential etc. The box kernel is suitable to be implemented on GPU while other kernels require many texture lookups which will increase the burden on fragment processors. The formula (6) illustrates the discrete convolution.

\[ I(x_i) = k \sum_{j=-n}^{n} T(x_j) \quad \chi_i = \sigma(x_0 + ih_i) \]  

(6)

Where, \( k = I/(m + 1) \) and \( h_i \) denotes the sample distance.
Fig. 6. Texture convolution with the different convolution length. From left to right, the convolution length is 5, 10, 20, 30 and 40 respectively.

Fig. 6 shows the results with the different convolution length and the scheme of the streamline enhancement is used as post-processing. When the convolution length is not more than 10, the streamlines are not continuous enough and the image quality of flow rendering is low. With the increase of the convolution length, the more smooth streamlines are generated and the more computational burden is on our system. The appropriate convolution length is usually at a range of 20-30 in which the streamlines are smooth enough.

3.4. Edge detection and supplement

As for the result of particles advection in image space, a visual flow continuity may be introduced where it is not desired in some cases. These discontinuous regions are either boundary edges or edges between a backward and forward facing polygon. Fig. 7 shows these edges, such as the black silhouette and the blue lines. Considering that the advected particles in image space cannot traverse these edges, we incorporate a geometric edge detection procedure into the algorithm. An edge can be identified by comparing spatially adjacent depth values during one integration and advection step. By a given threshold \( t \), if formula (7) can be satisfied, then an edge is identified. The paper [7] has detailed this edge detection procedure.

\[
|Z_{k-1} - Z_k| > t \cdot |P_{k-1} - P_k|
\]

(7)

Where, \( Z_{k-1} \) and \( Z_k \), \( P_{k-1} \) and \( P_k \) are Z depth values and coordinates of two adjacent points. Because the sample points or particles near the inflow edges is few, and the convolution length is usually not enough, thus result in a trivial change of intensity in the edge regions, and the streamlines may be not smooth enough. With the continuous particles advection, it may weaken the fluidness of particles in whole flow fields and therefore make it is hard to observe these particles advection. In this situation, we propose a strategy of edge supplement to enhance differences among the property and accelerate the particles flow.

Fig. 7. Edges on a surface

Fig. 8 shows the edge supplement, the blue line denotes an edge and black lines denote the inflow streamlines. These two textures are required for convolution computation, that is the property texture and a random noise texture. As for inflow streamline \( L \), convolution computation is implemented on the property texture and the convolution length is not more than 5 for these black points. Virtual streamline \( L' \) is the extended part of \( L \) with the same direction as \( L \) at edge point. Edge supplement is implemented on the noise texture by convolution along \( L' \) and these red points are the additional sample points. In fact, the edge supplement is regarded as the inflow of outside particles which can be used to modulate the characteristic of a flow field along the streamlines.

4. Experiments and results

We tested our implementation on a Windows XP PC with an Intel Pentium IV (3.0 GHz), 512 M RAM and a NVIDIA Geforce7600 GT graphics card with 256MB.
of video memory. Our implementation takes advantage of the programmable ability provided by graphics hardware with OpenGL specification and Cg shader Language. The particles advection and noise injection and blending can be performed in one rendering pass in our algorithm. By using of the parallel processing ability of current graphics hardware, our algorithm can achieve a rendering speed of over 30 frames per second.

Fig. 9 shows rendering results for a frame buffer resolution in 512x512, the fame (a) adopts a velocity magnitude map. By combining the mapping of scalar properties, our method can not only clearly visualize flow, but also reveals its scalar properties. The frame (b) is combined with surface shading. The accompanying videos can be downloaded from the website: http://saa.zju.edu.cn/dibinz/index.htm.

(a)                                (b)
Fig. 9. Flow visualization results

5. Conclusion and future work

In this paper, we present an enhanced flow visualization algorithm to visualize flow on surfaces. A 1D high-pass filter is used in this algorithm to enhance the intensity contrast among streamlines while has no influence on the convolution texture along streamlines. It is image space based and with good spatial-temporal coherence.

We analyze the influence of the sample distance parameter alpha on flow visualization and argue that the appropriate alpha value is in a range of 0.3-0.5. This range may be adjusted for different vector fields. The reasonable convolution length is in a range of 20-30. The flow image quality may be lowered using a shorter convolution length while the convolution intensity is increased on fragment processors using a longer one.

In future, we try to apply our method to the unsteady 3D vector fields and continue to improve the flow image quality and enhance the temporal coherence by dynamically modulating the impact parameters of convolution texture.

References