Image-Space Texture-Based Output-Coherent Surface Flow Visualization

Jin Huang, *Member*, *IEEE*, Zherong Pan, Guoning Chen, *Member*, *IEEE*, Wei Chen, *Member*, *IEEE*, and Hujun Bao, *Member*, *IEEE*

Abstract—Image-space line integral convolution (LIC) is a popular scheme for visualizing surface vector fields due to its simplicity and high efficiency. To avoid inconsistencies or color blur during the user interactions, existing approaches employ surface parameterization or 3D volume texture schemes. However, they often require expensive computation or memory cost, and cannot achieve consistent results in terms of both the granularity and color distribution on different scales. This paper introduces a novel image-space surface flow visualization approach that preserves the coherence during user interactions. To make the noise texture under different viewpoints coherent, we propose to precompute a sequence of mipmap noise textures in a coarse-to-fine manner for consistent transition, and map the textures onto each triangle with randomly assigned and constant texture coordinates. Further, a standard image-space LIC is performed to generate the flow texture. The proposed approach is simple and GPU-friendly, and can be easily combined with various texture-based flow visualization techniques. By leveraging viewpoint-dependent backward tracing and mipmap noise phase, our method can be incorporated with the image-based flow visualization (IBFV) technique for coherent visualization of unsteady flows. We demonstrate consistent and highly efficient flow visualization on a variety of data sets.

Index Terms—Flow visualization, mipmap, LIC, IBFV, surface flows, unsteady flows

1 Introduction

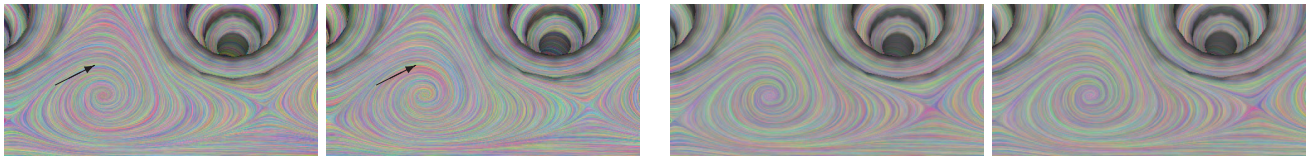
Visualizing vector fields is of great importance in many applications like engineering design, computational fluid dynamics, and climate research. Many approaches for vector field visualization have been studied, among which line integral convolution (LIC) [1], image-based flow visualization (IBFV) [2] and their variants are most popular. Reasons for that include their superior capability of space-filling, and the high efficiency when implemented on modern graphics hardware. More importantly, such a texture-based method can be applied to surfaces to study the flow in 3D space. This paper concentrates on the LIC visualization of surface vector fields. An extension on unsteady flow visualization is also provided based on the framework of IBFV to achieve coherent texture appearance during user interactions.

Generally speaking, existing texture-based visualization approaches for surface vector fields need to compute the flow texture in the 2D parametric space. According to the selected parameterization scheme, they can be classified into two categories: global surface parameterization methods and image-space methods. The first category generates the LIC textures in a parametric space, which covers the entire surface and naturally preserves the texture continuity over the surface even under user interactions. However, this scheme is difficult to produce high-quality results, especially for large-scale surfaces with complex geometry and topology. Alternatively, an image-space approach generates LIC images for only the visible portion of the surface under a given viewpoint. Specifically, it projects the vector fields and surface geometry into the viewing screen, and then applies 2D LIC or texture advection in the image space. This provides higher performance than the parameterization-dependent approaches, and is easy to implement with the aid of modern graphics hardware. However, this scheme may suffer from artifacts of inconsistency (see Fig. 1) when the viewer is rotating or zooming the model. In other words, the LIC result is not consistent among consecutive frames during user interactions when the noise texture is generated independently on the image space for each frame, or say, the noise texture on the surface is changing.

In this paper, we present a novel image-space visualization technique for surface vector fields that addresses the texture inconsistency problem. Our approach takes a similar pipeline as the conventional image-space surface LIC visualization, while modifies it with a simple texture mapping technique and an additional preprocess stage. Concerning the first one, we propose to fix the texture coordinates of each vertex with a simple triangle-texture matching technique, making the noise textures under different viewpoints coherent. In contrast, conventional image-space approaches cannot ensure this by mapping the entire model to the texture space. Our scheme is feasible because convoluting a white noise with a vector field yields a result that is not sensitive to the texture coordinates. In this way, no surface parameterization is required, and the underlying surface model can be arbitrarily complex and large.

- *J. Huang, Z. Pan, W. Chen, and H. Bao are with State Key Lab of CAD&CG, Zhejiang University, Zhejiang, China 321000. E-mail: {hj, panzherong, chenwei, bao}@cad.zju.edu.cn.*
- *G. Chen is with the Department of Computer Science, University of Houston, Philip G. Hoffman Hall, Room 566, Houston, TX 77204-3010. E-mail: chengu@cs.uh.edu.*

Manuscript received 31 May 2012; revised 10 Nov. 2013; accepted 22 Feb. 2013; published online 28 Feb. 2013.
Recommended for acceptance by H. Hauser, S. Kobourov, and H. Qu.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2012-05-0092.
Digital Object Identifier no. 10.1109/TVCG.2013.62.



(a) (b)

Fig. 1. Two consecutive frames during the zooming-in operation. Randomly generating the noise in the image space [3] causes popping of color as shown in (a), especially around singularities. Our method generates a consistent streamline rendering result (b). Please see the accompanying video for a better illustration.

Even with a consistent noise texture, the result can still exhibit popping artifacts caused by texture aliasing. Meanwhile, the LIC streamlines may greatly vary with the zooming operations, which makes the result unclear or flickering. Our solution is to precompute a customized noise texture pyramid that simultaneously characterizes the consistency and variance of the noise texture. The mipmap level decision is made on-the-fly. This scheme ensures consistent LIC results because all noise textures have similar appearances and differ only in the granularity. Several enhancement techniques are also proposed to control the effects like the contrast, the density, and the length of streamlines.

For unsteady flow visualization, we extend our approach by leveraging the framework of IBFV, whose kernel is the blending of each frame and the time-variant phase noise injected at its subsequent frame. In particular, we employ the mipmap noise texture as the initial phase which provides coherent screen resolution for the pathline. It is apparent that fixing the texture coordinates of each vertex cannot fully prevent the pathlines from drifting on the surface while the viewpoint is changing. This is because a frame should be properly warped into its consecutive frame with respect to the viewpoint change. Accordingly, an additional step is performed to transform the point in one frame to its previous frame by using the transformations from the object-space to the screen-space. Two types of artifacts that appear near the boundary or in the high velocity region with the conventional IBFV approaches, can be greatly alleviated by locally reverting flow direction and applying a low-pass filter along the pathlines.

In summary, this paper presents an efficient surface LIC visualization approach with the following contributions:

1. A parameterization-free image-space surface LIC generation scheme that works for arbitrarily complex (manifold or nonmanifold) and large mesh models with $O(N^2)$ memory complexity (assuming the image is at the size of $N \times N$).
2. A novel mipmap-based noise texture generation technique that is output coherent, and leads to smooth transition of LIC streamlines and consistency in different scales; to further improve the consistency around the silhouette, i.e., the surface boundary, the streamlines are extended to the back face of the surface. This mode is quite different to conventional image-space approaches that only employ the visible portion of the surface.
3. An extended unsteady flow visualization scheme that is consistent in both resolution and position of the pathlines during the viewpoint change. Methods to alleviate the artifacts near boundaries and in high velocity regions are proposed.

The remainder of this paper is organized as follows: We first review related work in Section 2 and then present our approach in Section 3, and extend it to unsteady flow visualization in Section 4. Next, we show the results in Section 5. We summarize this paper and highlight the future work in Section 6.

2 Related Work

Reviewing all work on vector field visualization is beyond the scope of this paper. We will cover the most relevant work here (i.e., LIC-based methods) and refer readers to [4], [5] for a comprehensive survey.

Van Wijk [6] introduced the first texture-based vector field visualization method that distributes a large number of spots on the spatial domain, and generates an illustrative texture, called spot noise. Inspired by that, Cabral and Leedom [1] proposed to locally smooth an input noise texture by convoluting the texture with a filter kernel derived from the vector fields. The so-called LIC technique leads to a high correlation along streamlines, and generates a dense texture representation for vector fields. Simply speaking, both spot noise and the LIC-based methods employ a space-filling scheme, and are amenable for parallelization.

Subsequently, the LIC-based vector field visualization approach has been improved in various aspects [7], [8], [9], and been extended to surface [10], unsteady flows [11], and 3D flows [12], respectively. The efficiency of LIC is also improved by a fast LIC approach [13], of which the speed acceleration is gained from the minimization of the total number of streamlines.

In particular, Forssell [10] first applied the LIC-based visualization approach to parametric surfaces by transforming the vector fields into the parameter space, and generating LIC in this parameter space. In the last step, the LIC result is mapped back to the surface. The main problem with this scheme is that it is often difficult to get a global parameterization with low distortion, which is almost impossible for nonmanifold or curved (i.e., nonflat) surface models. Alternatively, the technique proposed by Battke et al. [14] tessellates a surface with triangles and packs triangles into texture memory. A local LIC texture is computed for each triangle based on its local euclidean coordinates, avoiding the global parameterization. Its main drawback is that triangle packing demands the model to have a good mesh quality. It should be noticed that our

approach basically takes a local parameterization means. The difference from the parameterization-dependent methods [15] lies in that our approach does not fulfill the LIC in the parameterization domain, and thus removes the requirement of low distortion parameterization.

On the contrary, image space scheme directly generates the screen-space and has no requirement of parameterization. Since the work of Heidrich et al. [16], many image space LIC algorithms were proposed on GPU, yielding excellent performance. Many of them focus on properly visualizing the features of the vector field. For example, Li et al. [3] accurately represented higher order singularities by a novel vector field interpolation scheme. Recently, Zhang et al. [17] presented a visualization technique for planar and surface tensor fields based on the image-based flow visualization. Palacios and Zhang [18] advanced the image-space LIC visualization approaches to allow for interactive visualization of rotational symmetry fields both in the plane and on surfaces.

Image-based methods have also been extended to unsteady visualization. UFLIC [11] and IBFV [2] are two classical methods. In UFLIC, the color of a pixel is the weighted average of randomly colorized seed particles which pass it in a time window. IBFV is regarded as the fastest algorithm for texture-based visualization of 2D unsteady vector fields. It differs from other solutions in that it advects the color of the previous frame into the current one, and then blends it with a periodic noise texture. Based on this technique, two dense texture-based methods were proposed for visualizing unsteady vector fields on surfaces: image-based flow visualization for curved surfaces (IBFVS) [19] and image-space advection (ISA) [20]. Both approaches mentioned in [21] compute the LIC in the image space by leveraging a projection-and-advection pipeline. In this way, no parameterization of surfaces is needed, and the entire procedure can be accelerated using modern graphics hardware. However, without the consideration of moving viewpoint, the path-line pattern drifts over the surface instead of naturally sticking to the surface. To address such incoherency, object space texture mapping and tracing is required. A singe resolution object space noise texture leads to incoherent granularity in image space of different zooming levels, and thus we propose the strategy of using mipmap texture in a recent work [22]. In addition, we analyze the incoherency in IBFV and extend the coherent LIC method in [22] for unsteady flow visualization in this paper.

Although image-based methods have great advantages in terms of efficiency, they typically suffer from inconsistencies or color blur during user interaction because the noise texture is often independent from the object in each frame and has inadequate resolution. Weiskopf and Ertl [23] proposed an excellent method to address these issues: embedding the surface in an object-space 3D solid noise texture leads to consistent LIC result, and blending multiple noise textures in different resolutions makes the granularity consistent. Our approach differs from their approach in two aspects. First, our approach achieves similar results with less memory requirements. Second, our approach can yield similar color distribution in different scales with better consistency, especially for LIC visualization (see the comparison in Fig. 2 concerning the inconsistency).

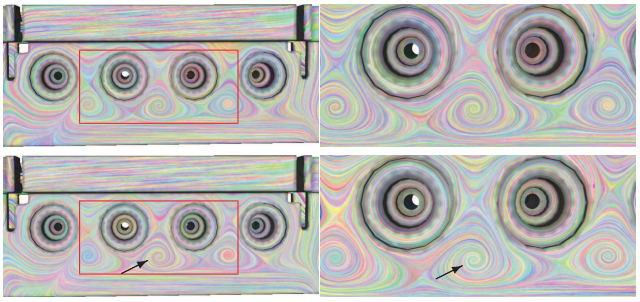


Fig. 2. The images on the right column depict the regions in red in the left images by moving the camera closer to the surface. Our method utilizes a set of correlated noise textures in a mipmap texture pyramid, and yields results with similar color distributions in different scales (upper row). Using uncorrelated multiresolution noise textures (lower row) cannot achieve such consistency.

3 Output-Coherent LIC for Steady Flows

We adopt LIC for steady flow visualization because it produces better image quality than IBFV. The goal of our work is to enable consistent visualization of vector fields on surfaces. That is, the LIC results share similar color distribution and granularity whenever the surface is rotated, translated, or scaled. Throughout this paper, we describe our algorithm in terms of a triangular surface model, and it is easy to adapt for other surface representations.

Our approach can be separated into two stages: preprocessing and visualization. The pipeline (Algorithm 1) is highlighted in Fig. 3. In the preprocessing stage, we generate a consistent noise texture pyramid and compute the texture coordinates for each triangle, which is the focus of our work (see Section 3.2). When the viewpoint is changed, we project the vector field and noise texture to the image-space, and perform the LIC integration (backward Euler method) on the image-space with the consideration of silhouettes (see Section 3.4). In the stage of vector field projection, we use a similar scheme to ISA [20] for a triangle mesh: for each vertex q with vector u, a vertex shader is used to calculate the image-space coordinates q' and $(q+u)'$ of q and $q+u$, respectively. Then, the projected image-space vector is evaluated by normalizing $(q+u)'-q'$. Given the image-space vector field defined on each vertex, the vector on each pixel can be interpolated in the downstream fragment shaders.

Algorithm 1. The pipeline of applying our approach to LIC visualization.

Input : A triangle mesh with a vector field on it
Output: Interactive visualization result

preprocessing stage:
Generate a consistent noise texture pyramid.

visualization stage:
for *each frame* **do**
  Project the vector field to the image-space.
  Generate texture coordinates in a geometry shader.
  Map the noise texture.
  Perform the LIC in the image-space.
end

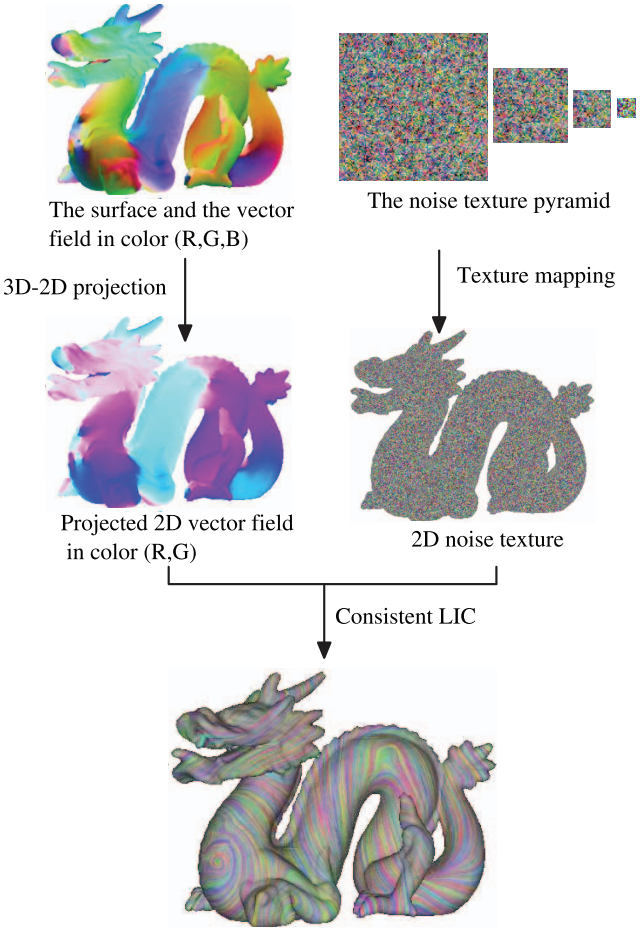


Fig. 3. The pipeline of applying our approach to LIC visualization.

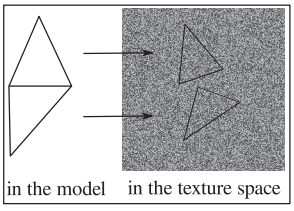
3.1 Mapping Noise Texture to the Mesh

In previous image-space surface LIC approaches, the inconsistency appears because the noise texture in the image-space is generated independently frame by frame. Our solution for that is to attach the model with a consistent noise texture by means of a simple texture mapping process:

1. Generate a random white noise 2D texture map. The size of the texture is determined by the projected size of triangles in the image-space. If it is too small, the noise texture will wrap and influence the white noise property.
2. Generate texture coordinates for each triangle by randomly projecting it into the texture space with the same scale. To avoid unnecessary memory cost, the texture coordinates can be generated on-the-fly in geometry shader. For each triangle with vertex positions $q_i, i = 1, 2, 3$, we construct a local 2D frame by two orthogonal directions $\mathbf{u} = (q_2 - q_1)/\|q_2 - q_1\|$ and $\mathbf{v} = \mathbf{n} \times \mathbf{u}$, where $\mathbf{n}$ is the unit normal of the triangle. Then the texture coordinates for the ith vertex can be generated as

$$R(\theta)\begin{pmatrix}\mathbf{u}^T\\ \mathbf{v}^T\end{pmatrix}q_i, \tag{1}$$

where $R(\theta)$ stands for a 2D rotation in a "random" angle θ. In our implementation, θ is set to $\|q_1 + q_2 + q_3\|$, so that θ is a constant value for a triangle during the visualization.



This process can be regarded as a simple local (per triangle) parameterization, and has no requirement on the continuity and distortion of the parameterization. Satisfying results are yielded because the texture image contains only white noise. The texture coordinates of each triangle remain the same during interaction as well as the noise textures. This means that each triangle will be constantly mapped to the same portion of the texture at each frame, yielding consistent output.

3. Perform texture mapping with the given texture coordinates and generate texture noise in image space.

The described technique leads to consistent noise in the image-space when the viewpoint is changed. Because no global parameterization is required, it is naturally suitable for complex (manifold or nonmanifold) and large-sized models. Compared to [23] that uses 3D solid texture with the $O(N^3)$ complexity, our approach uses only 2D texture images and requires less memory consumption at the complexity of $O(N^2)$.

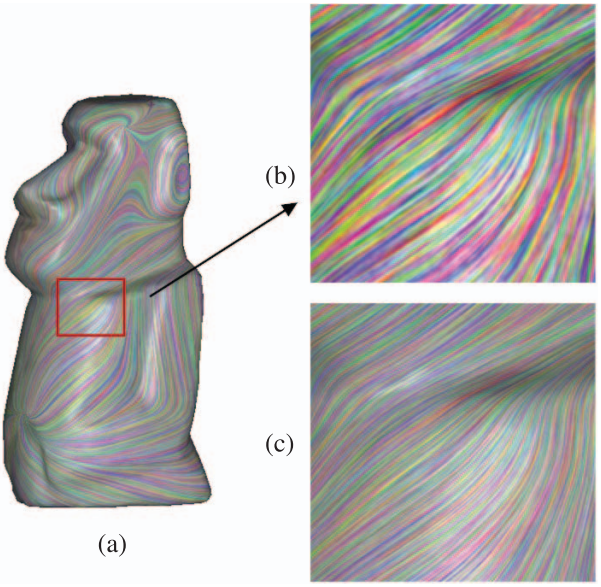


Fig. 4. (a) An LIC result for the Moai model. When the model is zoomed in, the result (b) with the noise texture used in (a) presents color blur, while the result (c) with a noise texture selected from the precomputed texture pyramid presents pleasing effect.

3.2 Generating Consistent Noise Texture Pyramid

Even though the noise is consistent, the LIC result will still present popping artifacts caused by the texture aliasing, especially when the model is zoomed in (please see the accompanying video). In addition, the streamlines will become thicker (see Fig. 4b) than desired (see Fig. 4c) when

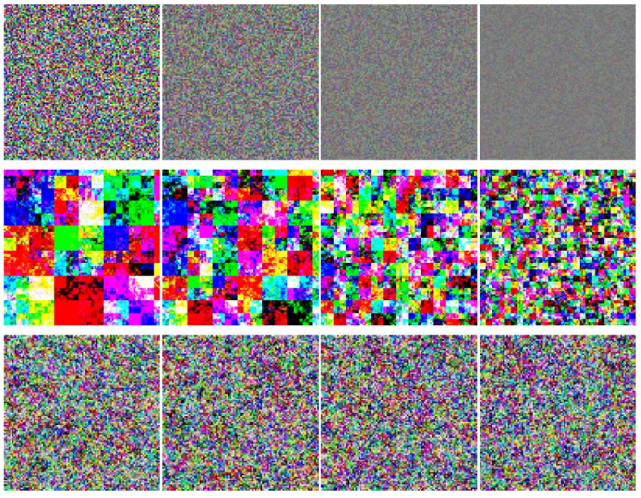


Fig. 5. From left to right, the images show a *small portion* of the texture image in the pyramid in the order of resolution gradually decreasing. The first row shows automatic-generated noise mipmap. As the resolution decreases, the maps become blurred. The second row shows the customized maps without the scaling operation in (3). It can be seen that the noise maps have color blocks. The maps in the last row are generated using our algorithm (4).

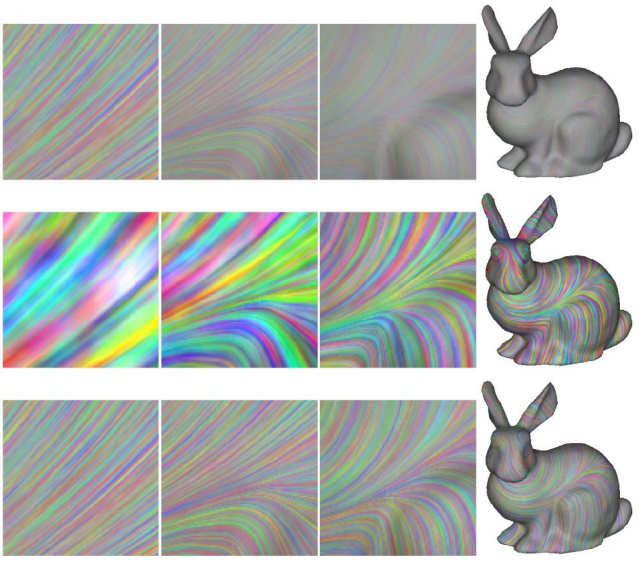


Fig. 6. From left to right, the model is zoomed out. The first row shows the results at different resolutions with the automatic-generated mipmap. The results become blurred when the model is zoomed out (from left to right). The second row shows the results by employing (3), where big color blocks appear when the model is zoomed in (from right to left). The last row presents the best quality with (4).

the model is zoomed in. In this result, a 256×256 texture image is used.

3.2.1 Automatic Mipmap Generation

A straightforward solution would be using the mipmap technique (the automatic-generated texture pyramid) provided in standard graphics libraries like OpenGL. The texture pyramid is composed of a sequence of noise textures, of which the $n^{th} \in [n_{min}, n_{max}]$ image has the resolution of $2^n \times 2^n$. In this way, the popping artifacts can be alleviated during the zooming operation, and the resolution of streamlines is appropriate as long as the zooming factor does not exceed the finest level in the pyramid. However, this scheme may fail because the contrast of the LIC streamlines decreases greatly when the model is zoomed out. This problem also exists where the projected surface part is largely sheared, yielding overblurred results (see Figs. 5 and 6).

Let us briefly analyze the process of automatic mipmap generation. Both the random colors (see [18]) and gray-level noise ([23]) can be used to generate the LIC result. Our approach employs the random colors by generating white noise for each channel of RGB independently. When computing the image in the $(n-1)$th level from the nth level, the value of each RGB channel $I_{n-1}(x, y) \in [0, 1]$ at (x, y) in the $(n-1)$th level is derived from the corresponding four pixels in the nth level:

$$I_{n-1}(x, y) = \frac{\sum_{\Delta x, \Delta y \in \{0,1\}} I_n(2x + \Delta x, 2y + \Delta y)}{4}. \quad (2)$$

According to (2), the variance of the $(n-1)$th level is smaller than that of the nth level. As a result, the $(n-1)$th level is more blurred than the nth level (see the first row of Figs. 5 and 6). This explains the reason of the blurred effect when the model is zoomed out, that is, the continuous diminution of the variance of higher level textures. In other words, the automatic mipmap generation mode recursively employs low-pass filtering and generates the textures from fine to coarse. To ensure that each texture in the mipmap has an adequate variance, a fine texture that contains spot noises in different resolutions/frequencies is needed. However, this task is difficult.

3.2.2 Customized Noise Texture Pyramid

To address the aforementioned problem, two requirements should be met: the maps of the adjacent levels have adequate correlation to make the result consistent and popping-free, and meanwhile, the variance of the map at each level is stable to avoid blurred effect.

For the first goal, the noise texture pyramid should be generated from coarse to fine instead of using a fine-to-coarse process as in standard mipmap generation techniques. Such a coarse-to-fine strategy is also used to generate stroke-based texture in [24] for preserving the stroke resolution in different scales. Beginning from the coarsest map at the resolution of 1×1, a finer map can be successively generated by means of the following rule:

$$I_{n+1}(2x + \Delta x, 2y + \Delta y) = I_n(x, y) + \sigma(2\gamma_{\Delta x, \Delta y} - 1), \quad \Delta x, \Delta y \in \{0, 1\}, \quad (3)$$

where $\gamma_{\Delta x, \Delta y}$ is a random number uniformly distributed between $[0, 1]$, and $\sigma \in R$ is an adjustable number to tune the variance.

Applying (3) can produce a consistent texture pyramid. However, the value will be out of $[0, 1]$ when $I_n(x, y)$ is close to 0 or 1. Clamping it into $[0, 1]$ yields obvious color blocks as shown in the second row of Figs. 5 and 6. These block artifacts are actually caused by inappropriate local variances.

Thus, a better solution would take both the correlation and variance into account, which is essentially an

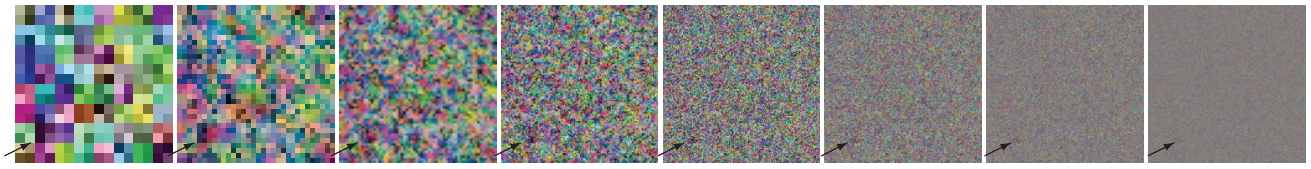


Fig. 7. The sequence of the mipmap texture images (from 16×16 to $2{,}048 \times 2{,}048$). The arrows point to the relatively dark regions.

optimization problem. We design a simple and efficient scheme: $I_n(x, y)$ is linearly mapped from $[0, 1]$ into $[\eta, 1-\eta], \eta \in [0, 0.5]$, and then add four random numbers $\xi_{\Delta x,\Delta y}$ to generate $I_{n+1}(2x + \Delta x, 2y + \Delta y)$ in the $(n+1)$th level. This can be expressed as

$$I_{n+1}(2x + \Delta x, 2y + \Delta y) = ((1 - 2\eta)I_n(x, y) + \eta) + \xi_{\Delta x,\Delta y} := I'_n(x, y) + \xi_{\Delta x,\Delta y}. \quad (4)$$

To ensure $I_{n+1}(2x + \Delta x, 2y + \Delta y) \in [0, 1]$, $\xi_{\Delta x,\Delta y}$ should be in $[-I'_n(x, y), 1 - I'_n(x, y)]$. The difference between the mean of four values $I_{n+1}(2x + \Delta x, 2y + \Delta y)$ and $I_n(x, y)$ is $\eta(1 - 2I_n(x, y)) + \alpha$, where α stands for the mean of the four random numbers $\xi_{\Delta x,\Delta y}$. Setting $\alpha = \eta(2I_n(x, y) - 1)$ will increase the correlation, but will decrease the variance especially when $I_n(x, y)$ is close to 0 or 1. Thus, we choose to generate four random numbers with the zero expectation, and use the following equation to compute the required random numbers in the range of $[-I'_n(x, y), 1 - I'_n(x, y)]$ with zero expectation:

$$\xi_{\Delta x,\Delta y} = \text{power}\left(\gamma_{\Delta x,\Delta y}, \frac{1 - I'_n(x, y)}{I'_n(x, y)}\right) - I'_n(x, y). \quad (5)$$

Finally, it turns out that the values of the four pixels in level $n + 1$ are as follows:

$$I_{n+1}(2x + \Delta x, 2y + \Delta y) = \text{power}\left(\gamma_{\Delta x,\Delta y}, \frac{1 - I'_n(x, y)}{I'_n(x, y)}\right). \quad (6)$$

In other words, four random numbers in the range $[0, 1]$ with expectation $I'_n(x, y)$.

Note that η should be small to ensure high correlation, but a small η will yield small variance, i.e., blurred or low contrast result. Experimental results (e.g., Figs. 5, 6, and 7) indicate that 0.25 is a good choice. In all coherent results, a mipmap texture pyramid from 1×1 to $2{,}048 \times 2{,}048$ is used.

3.3 Addressing Surface Silhouettes

For image-space surface LIC approaches, special care must be taken on the object-space geometric discontinuities around the surface boundaries, i.e., the silhouette in the image-space. The reason is that the integration of a streamline in the image-space stops at the silhouette while in the object-space its counterpart will continue onto the back face of the surface.

The silhouette is composed of points that meet the following condition [20]:

$$|z_{i+1} - z_i| > \varepsilon |p_{i+1} - p_i|, \quad (7)$$

where ε is an adjustable threshold. p_{i+1} and p_i are two consecutive points along the integral path in the image-space, and z_{i+1} and z_i are their depth values in the object-space.

Traditional methods [19], [20], [21] solve the issue by blending a silhouette mask over the image to diminish the artificial continuity on the silhouette. Feeding fake noise value in the invisible part of the integration path are also adopted in some methods [3], [19], [23]. Such treatment may lead to popping artifacts near the silhouette between two consecutive frames. To achieve constant gray value (or color) for a pixel for coherent LIC visualization, the integration path (a segment of streamline), and the noise in the path should be the same for different viewpoints, even if the pixel is near the silhouette.

We adopt the idea of object-space LIC integration, and propose a two-stage scheme to drive the integration of the streamlines onto the back face of the model. First, the front and back faces of the underlying surface are extracted by rendering the surface twice, i.e., with the back-face culling and front-face culling operations, respectively. In this process, the projected vector field and the noise texture in the image-space that correspond to both faces are generated. Second, the conventional LIC is performed in the image space for the front face part. When the integration of a certain streamline meets the silhouette, its value is computed by considering the back-face parts of the vector field and the noise texture.

4 Output-Coherent IBFV for Unsteady Flows

Although LIC is a successful technique for steady flow visualization, it cannot be directly applied to unsteady flows because the generated streamlines lack temporal coherence among frames. IBFV overcomes this issue by replacing the streamlines with pathlines. However, traditional IBFV approaches may lead to incoherency when the viewpoint is changing. As mentioned in [21], using ISA and IBFV takes a short time to achieve a stable visualization when the viewpoint is changed. In addition, the results may be incoherent between different viewpoints even when the individual frames can properly depict the flow. Before introducing our solution, here we first briefly review the framework of IBFV.

IBFV is a 2D flow visualization method that generates a new frame by blending the previous frame with a time-variant noise texture. In the kth frame (at time $k\Delta t$), the position of a 2D point p on the object is denoted as p_k. The point p_{k-1} moves to $p_k = p_{k-1} + v(p_{k-1}; k-1)\Delta t$ by the 2D unsteady flow v. Then the color of the point in the kth frame is evaluated as

$$F(p_k; k) = (1 - \alpha)F(p_{k-1}; k-1) + \alpha G(p_k; k), \quad (8)$$

where $F(p_k; k)$ and $G(p_k; k)$ stand for color and the injected noise at p_k in the kth frame, respectively. A typical scheme for noise injection with period M is

$$G(p_k; k) = g(\phi_p; k) = w((k/M + \phi_p) \bmod 1), \quad (9)$$

where ϕ_p is the phase of p at the first frame, and is typically a uniformly distributed white noise over the object. $w(t) = 1$ for $0 < t < 1/2$ and 0 elsewhere.

Unlike LIC, the causes of incoherency in IBFV are improper advection and periodical noise injection. As such, we propose the following solution, whose pipeline is listed in Algorithm 2.

Algorithm 2. The pipeline of applying our approach to IBFV visualization.

Input : A triangle mesh with N frames of vector field on it.
Output: Interactive visualization result

preprocessing stage:
Generate a consistent noise texture pyramid as the phase noise.

visualization stage:
$k \leftarrow 1$
for $k < N + 1$ **do**
  Get the object-space to image-space transformation T_{k-1} of previous frames. If $k = 1$, $T_{k-1} \leftarrow T_k$.
  Render buffer $Q(\cdot; k)$.
  Generate texture coordinates in the geometry shader.
  for *Each pixel (Fragment Shader)* **do**
    Back trace the previous point according to (11).
    Inject noise to the warped image of previous frame according to (13) and (14).
  end
  $k \leftarrow k + 1$
end

4.1 Viewpoint Dependent Backward Tracing

Several methods [19], [21] have been proposed to extend the standard IBFV to handle curved surface with the assumption of fixed viewpoint. This makes the location of p_k fully determined by the velocity in the image-space. When the viewpoint changes, p_k and p_{k-1} may not correspond to the same particle advected by the flow. In other words, two points are not on a pathline. Thus, using p_{k-1} to fetch the color in the previous frame will lead to serious artifacts, such as pathline drifting, artificial flow, and even high frequency noise (see the top row of Fig. 8).

To solve this issue, we trace the image-space projection of a particle q of the flow $\dot{q}$ by considering the moving viewpoint. At the kth frame, we fill two three-channel buffers $q(\cdot; k)$ and $v(\cdot; k)$ with values of the object-space coordinates of the surface and the underlying flow, respectively. Given an image-space point p_k, we can efficiently fetch its object-space coordinates and the associated flow by means of $q(p_k; k)$ and $v(p_k; k)$. The particle position in the previous frame can be evaluated by the backward Euler technique:

$$q_{k-1} = q(p_k; k) - v(p_k; k)\Delta t. \tag{10}$$

The proper image-space position p_{k-1} is determined as

$$\begin{pmatrix} p'_{k-1} \\ z'_{k-1} \\ h_p \end{pmatrix} = T_{k-1} \begin{pmatrix} q_{k-1} \\ 1 \end{pmatrix}, \quad p_{k-1} = p'_{k-1}/h_p, \tag{11}$$

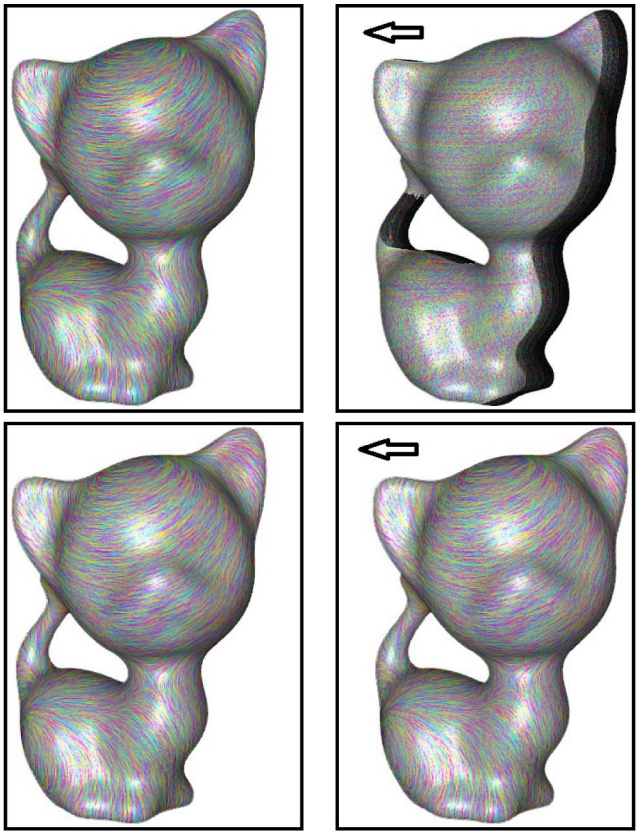


Fig. 8. The consecutive two frames during the interactive visualization of the unsteady flow. The arrow shows the moving direction of the viewpoint. Results with conventional approaches (top row) show incoherence and pathlines with serious artifacts. Our results (bottom row) exhibit better quality.

where T_{k-1} is the transformation from the object space to the image space in the previous frame. To accelerate the process, we can fill a single buffer $Q(\cdot; k)$ with the values of $Q(p_k; k) = q(p_k; k) - v(p_k; k)\Delta t$ so that a single texture lookup is adequate to get q_{k-1}. This scheme produces coherent pathlines during the user interaction, as shown in Fig. 8 and the accompanying video.

Although only p_{k-1} is required to fetch $F(p_{k-1}; k-1)$ with (8), z'_{k-1}/h_p should also be used to check whether p_k and p_{k-1} are associated with the same particle on the model. In each frame, we render the screen depth into a buffer z. If $|z'_{k-1}/h_p - z(p_{k-1}; k-1)|$ is larger than a user-defined threshold, q_{k-1} is regarded as invisible in the previous frame, i.e., the position of p_{k-1} is out of the screen or occupied by other parts of the model. This degeneracy often occurs on the screen border or silhouette. A simple treatment is to set $F(p_{k-1}; k-1)$ to be black. But this will cause severe illuminance loss (left of Fig. 9). Another option is to set $F(p_{k-1}; k-1) = F(p_k; k-1)$ by assuming that the flow field is zero locally, as used in [2]. However, this introduces artifacts near the boundary (middle of Fig. 9). Instead, we propose to reverse the flow field locally and trace the opposite direction. This scheme improves the possibility that a bright pixel is fetched and the average illuminance is maintained. Moreover, the artifacts are alleviated because the reversed pathline is unlikely to cross the boundary, as shown in the right of Fig. 9.

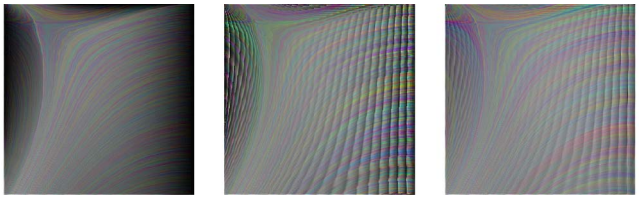


Fig. 9. On the boundary regions of the screen, special cares need to be taken. Simple treatment of setting to black (left) causes illuminance loss. Assuming zero flow field leads to artifacts (middle). Reversing the velocity field alleviates these problems (right).

4.2 The Mipmap Noise Injection

In conventional IBFV approaches, noise is periodically injected to each pixel according to its associated phase. Similar to LIC, the phase should be associated with the object-space model for coherency. Mapping a simple noisy phase texture onto the model leads to incoherent screen resolution. Thus, we build a mipmap texture for the noisy phase texture ϕ_p. To ensure that the noises are evenly injected onto the model, the phase should be uniformly distributed between 0 and 1. Applying a linear or anisotropic filter to the phase texture will distort the uniform distribution in ϕ_p because they cannot properly deal with the phase in a circle topology. A typical artifact appears when the average illuminance varies. One possible solution is to use the nearest filter. However, it may make the pathlines popping and modify their granularity abruptly when the viewpoint is changed.

We introduce an interpolation scheme to address this problem. For p_k, we calculate the noise to be injected as

$$G(p_k;k) = (1-\tau)g\big(\phi_p^l;k\big) + \tau g\big(\phi_p^{l+1};k\big), \tag{12}$$

where τ is the coefficient to interpolate two values at p_k in two neighboring textures (the lth and $(l+1)$th levels) of the mipmap textures. This simple technique yields satisfying results, as shown in Fig. 10.

To compute the appropriate mipmap level τ, we create an auxiliary mipmap texture which has the same size as the noise texture and employs a linear interpolation filter between the mipmap levels and texels. In this way, τ can be simply fetched by means of a single texture lookup from the auxiliary texture.

In regions of high flow velocity, particles may pass several pixels in the screen-space within a single timestep. This will cause severe artifacts (periodic patterns), especially when the model is zoomed in. To tackle this problem, conventional

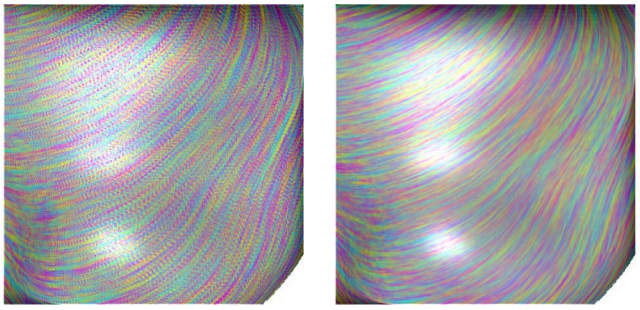


Fig. 11. (Left) When the model is zoomed, the flow velocity in the screen is large, and high-frequency periodic patterns appear along the pathline. By performing a low-pass filter along the pathline, our method eliminates the artifacts (right).

IBFV approaches clamp the flow velocity by a certain threshold. Alternatively, we propose to perform an LIC-like convolution between p_k and p_{k-1} which acts as a low-pass filter to remove high-frequency patterns. The pathline of the current timestep is uniformly sampled, and the averaged noise value is injected on these sample points. This process can be written as

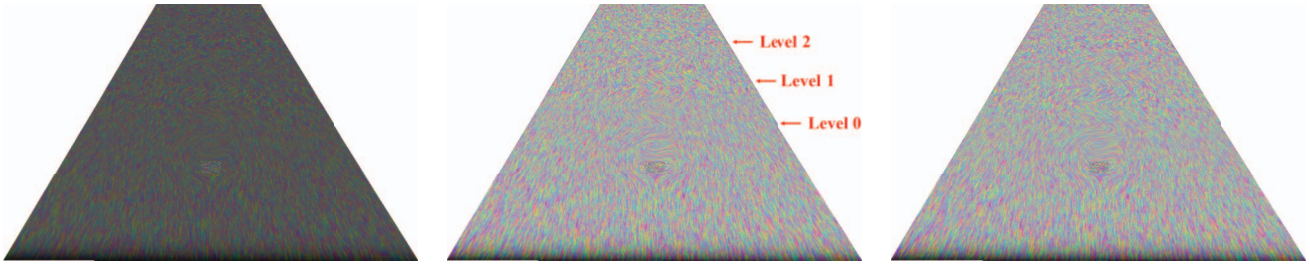
$$F(p_k;k) = (1-\alpha)F(p_{k-1};k-1) + \alpha G_{avg}, \tag{13}$$

$$G_{avg} = \frac{1}{N}\sum_{i}^{N} G\left(\left(1-\frac{i}{N}\right)p_{k-1} + \frac{i}{N}p_k;k\right). \tag{14}$$

In our experiments, N is set to be 5. The improvement over conventional ways is displayed in Fig. 11.

5 Results and Discussions

The primary advantage of our approach is that it is simple and efficient, and achieves comparable quality as the flow charts approach [15] and interactive performance. The results shown in the figures and video demonstration verify the efficiency and robustness. The vector fields on the Moai, Bunny, Buddha, Fish, and CAD models are generated with the methods of [17], [25]. The other data set are from real-world simulations. The Cooling-jacket data set [26] is produced by Robert S. Laramee; the unsteady flow on the plane is produced by Tino Weinkauf [27] using the Free Software *Gerris Flow Solver* [28]. The unsteady flows on the Bunny and Kitty models are produced with a fluid simulation method [29].



(a) Linear (b) Nearest (c) Ours

Fig. 10. The mipmap noise is used to achieve coherent screen resolution. (a) The linear filter distorts the distribution of the phase noise, and thus leads to incoherent noise intensity and time-variant illuminance. (b) The nearest filter avoids the issue, but yields popping and discontinuous artifacts. (c) Our method interpolates the output noise and produces pleasing results.

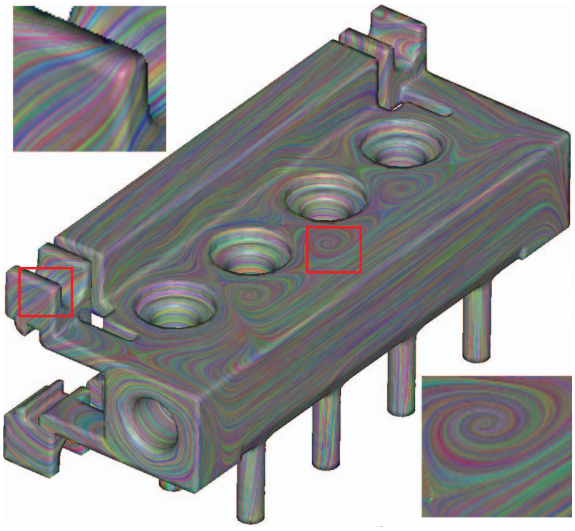


Fig. 12. Our result for a complex CAD model. Note that a consistent LIC visualization is very helpful to locate and check the singularities on the model.

One important advantage of our approach is that it is parameterization-free, and works for arbitrary mesh models. Fig. 12 shows a quite challenging case. The model is composed of many triangular patches. A low-distortion global parameterization is intractable even if the patches are merged into a single manifold mesh. Conventional parameter space LIC approaches can hardly handle this case, while our approach achieves consistent visualization that allows the user to easily locate and view interesting singularities. Without the requirement of parameterization, our method can also handle large-scale models which have a complex surface topology. The data set shown in Fig. 13 contains 227K faces and many holes, posing great challenges for the parameter-space methods like [15].

To demonstrate the coherence of the streamlines during an interactive visualization, we show two consecutive LIC frames in Fig. 14. The visualization effects about the resolution, the contrast, and the length of streamlines can be easily configured with our method. By modulating the parameter of the LOD bias, the noise density used for LIC is modified, and consequently the streamline resolution is changed (see Fig. 15).

Several methods have been proposed to avoid contrast loss caused by image blending [18], [30]. Following the idea of [18], we increase the image variance and fix the mean for contrast enhancing, and modulate the saturation (see Fig. 16).

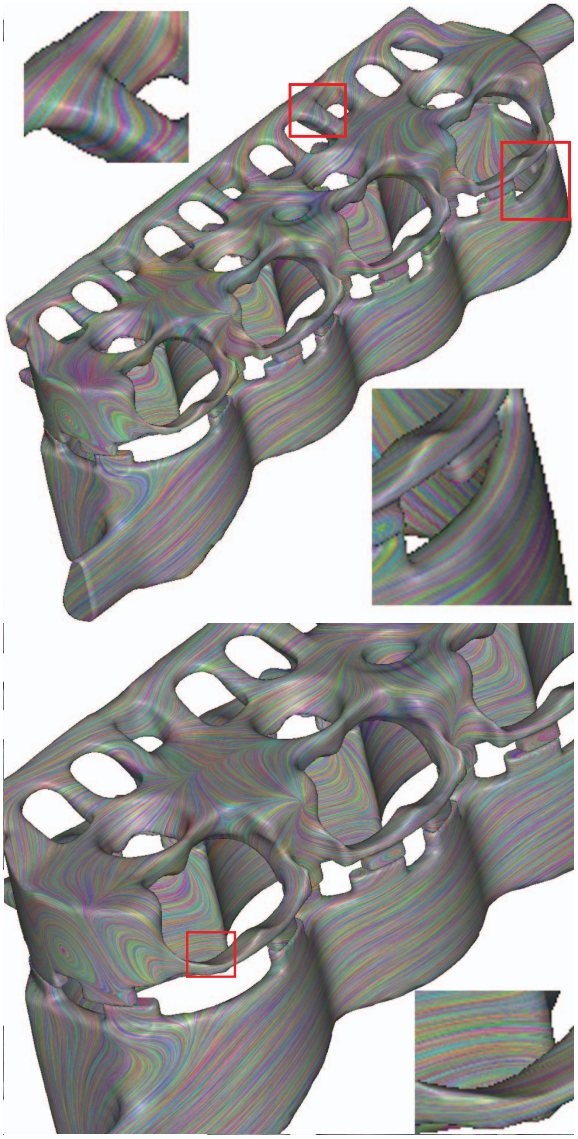


Fig. 13. LIC visualization for the cooling jacket data set.

In LIC, the streamline length is adjustable and influences the results, as shown in Fig. 17. There are two ways to measure the streamline length. If the streamline length is measured in the image space, the length scales well with the model zooming. Although a degree of almost invisible inconsistency may appear, a scale-dependent LIC is achieved. In contrast, measuring the length in object space

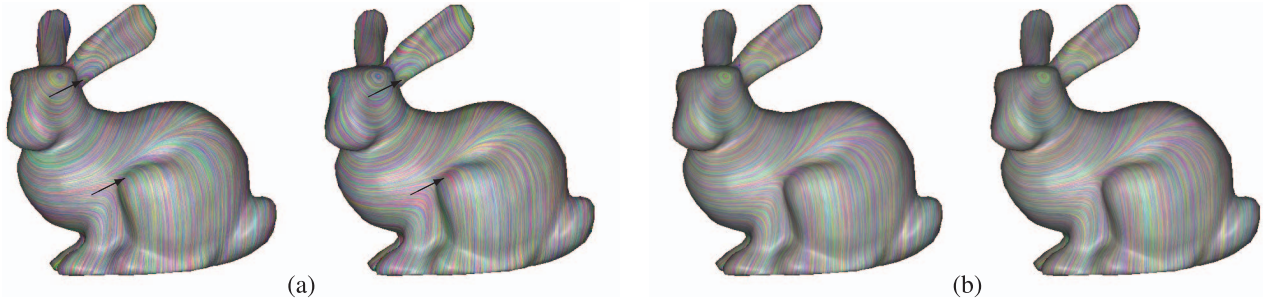


Fig. 14. Two consecutive LIC frames for the Bunny model. Our result (b) is more consistent than that of [3] (a). Please see the accompanying video for a better illustration.

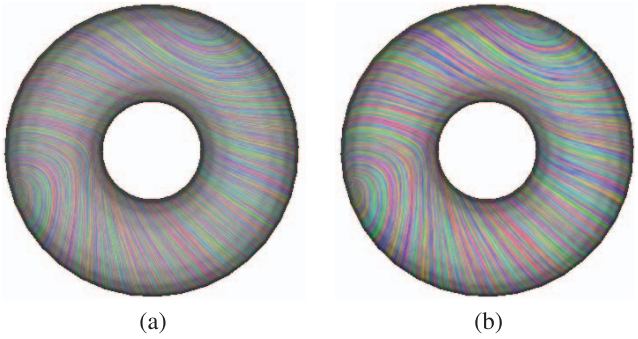


Fig. 15. Results with different resolutions. The triangles are mapped onto the noise texture with a higher resolution noise texture in the pyramid for (a), yielding thinner streamlines than those in (b).

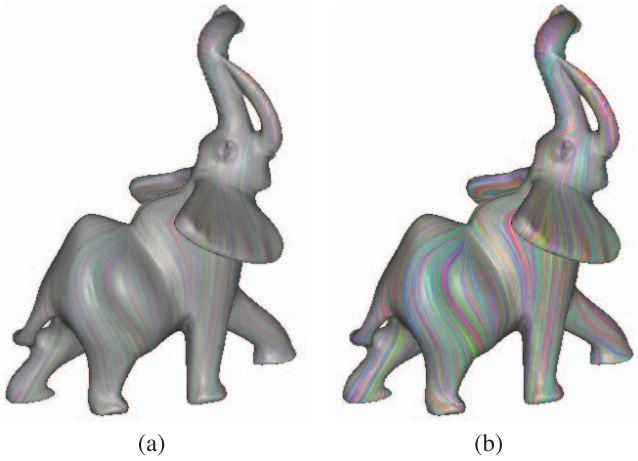


Fig. 16. Result comparison on the elephant model without (a) and with streamline contrast enhancement (b). This modulation leads to a balance between the details and contrast.

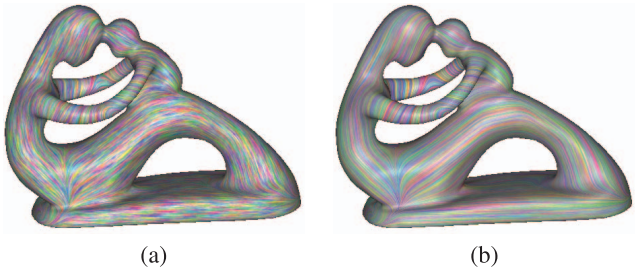


Fig. 17. The LIC streamline length on the fertility model used in (b) is longer than that of (a).

increases the consistency, but loses the property of adaptive granularity. In practice, the user can choose an appropriate way for different purposes, for example, varying the streamline length according to the vector magnitude.

In fact, all aforementioned enhancements can be incorporated into our approach without sacrificing the interactivity of the vector field visualization (see the accompanying video). This verifies the advantages of the image-space surface LIC over the object-space surface LIC schemes.

Our method can be easily integrated into other LIC algorithms. As a demonstration, we apply the noise map scheme into the LIC method [18] to visualize rotational symmetry fields [31], [32], [33] (see Fig. 18).

Fig. 19 shows the extension of our method on unsteady flow visualization. During the user interaction, the pathlines

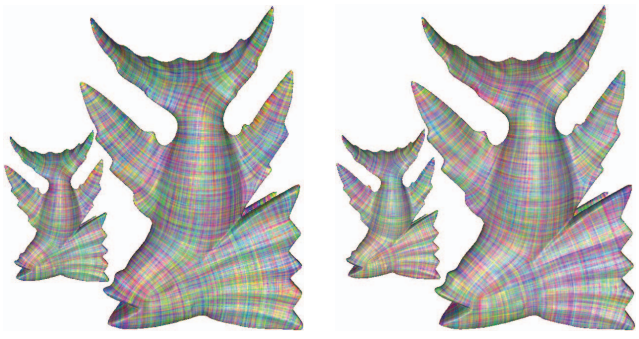


Fig. 18. Our method is compatible with other LIC algorithms. (a) The result of applying our method to visualize rotational symmetry fields [18]. Compared with (b) that uses a set of uncorrelated images in the texture pyramid, our method achieves better color consistency in different scales.

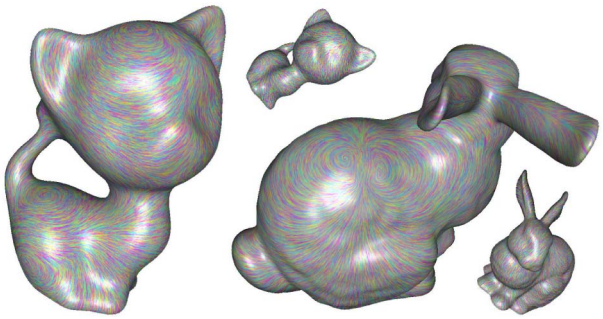


Fig. 19. Our IBFV extension provides coherent experience when the viewpoint is changed. The subfigures show several frames during a user interaction. Please see the accompanying video for a better demonstration.

appear coherent in terms of both the distribution and the screen resolution.

Like most surface flow visualization techniques, the main computation cost of each frame consists of three parts:

1. The basic model rendering process including sending triangles into the GPU, occlusion culling, and lighting.
2. Specific computation on the mesh, such as projecting the flow field onto the image space, evaluating texture coordinates for each triangular face.
3. Image-space computation, such as the LIC integration, tracing and blending for IBFV.

The cost of the first two parts is dependent on the size of the underlying model. View frustum culling and visibility culling are typical methods for speedup, and can be integrated into our approach.

We focus on the cost beyond a basic mesh rendering, especially the additional cost over conventional image-space LIC and IBFV approaches. The noise texture pyramid can be pre-computed and reused for all models. Another difference lies in the way of computing the texture coordinates of each triangle, which is generated by geometrical shaders with little cost. In the visualization stage, our approach projects the underlying vector field and the noise texture to the image space twice to handle the silhouette. This is still fast because every operation can be implemented on the GPU.

TABLE 1
LIC Performance Measurement in Milliseconds on the Cooling Jacket Model (227,868 Triangles) in Two Different Resolutions

algorithm	500 × 500	1000 × 1000
Basic surface rendering	7.8	8.5
2D vector field and noise	8.2	9.1
LIC without back face culling	1.1	1.9
LIC with back face culling	1.8	3.2
Contrast adjustment	0.7	1.2

The term of "2D vector field and noise" includes sending triangles into GPU, generating texture coordinates and triangle rasterization.

TABLE 2
Performance Measurement in Milliseconds with Two Different Resolutions

model	# tri	algorithm	500 × 500	1000 × 1000
Kitty	31126	IBFV	2.9	3.3
Complex CAD	66158	LIC	4.1	5.2
Bunny	65542	IBFV	4.8	6.0
Cooling Jacket	227868	LIC	11.9	14.2

We measure the performance on a desktop PC with Intel i7-2600K CPU and Nvidia GTX 560Ti GPU. The performance is dependent on the screen resolution, the integral length in LIC visualization and the subdivision of the backward tracing in the IBFV extension. Tables 1 and 2 show that our method achieves high frame rates for various configurations, and requires little cost for the basic surface rendering.

6 Conclusions and Future Work

This paper proposes a novel image-space surface flow visualization technique that achieves consistent and smooth results for both steady and unsteady flows. The key idea is to leverage a consistent 2D noise texture pyramid that works for arbitrary models and can be precomputed with the computational and memory complexity of $O(N^2)$. Our approach not only provides smooth transition and consistent screen resolution, but also leads to similar color distribution in different scales. The GPU-based implementation compares favorably with existing solutions in terms of performance and quality, and allows the user to interactively study a surface flow at an arbitrary resolution.

Concerning the future work, we would like to first apply the presented technique to the visualization of integral surfaces (e.g., the stream-surfaces [34]) or streamline visualization. Although our approach works fine for most models, popping artifacts can be still visible for some complicated models (e.g., the model shown in Fig. 20). We believe that a depth peeling process that is capable of extracting more than two surface layers (the front and back faces) can be used to address this problem.

In our implementation, 10 mipmap levels are employed. This configuration can still yield popping artifacts in some extreme situations, for example, when the viewpoint is very far away from or very close to the surface. It is interesting to explore new techniques to construct the texture images on-the-fly with respect to the current scale.

Acknowledgments

The authors would like to thank the reviewers for their valuable comments. This research was partially supported by NSFC (Nos. 61170139 and 61232012), China 863 Program (No. 2012AA120903), and the Fundamental Research Funds for the Central Universities (No. 2012FZA5019). Guoning Chen was partially supported by DOE VACET. Hujun Bao and Wei Chen are the corresponding authors.

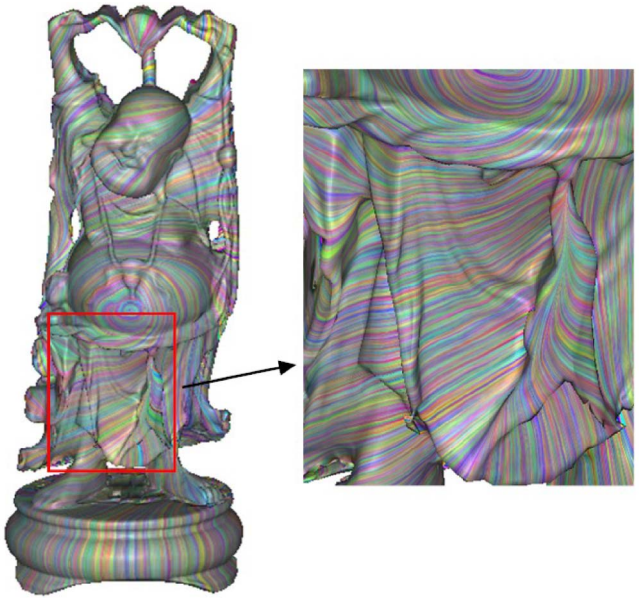


Fig. 20. Results for the Buddha model. The complex wrinkles still yield popping artifacts.

References

[1] B. Cabral and L. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Proc. ACM SIGGRAPH*, pp. 263-272, 1993.
[2] J.J. van Wijk, "Image Based Flow Visualization," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 745-754, 2002.
[3] W. Li, B. Vallet, N. Ray, and B. Lévy, "Representing Higher-Order Singularities in Vector Fields on Piecewise Linear Surfaces," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1315-1322, Sept./Oct. 2006.
[4] D.H. Laidlaw, R.M. Kirby, C.D. Jackson, J.S. Davidson, T.S. Miller, M. da Silva, W.H. Warren, and M.J. Tarr, "Comparing 2D Vector Field Visualization Methods: A User Study," *IEEE Trans. Visualization and Computer Graphics*, vol. 11, no. 1, pp. 59-70, Jan./Feb. 2005.
[5] R. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. Post, and D. Weiskopf, "The State of the Art in Flow Visualization: Dense and Texture-Based Techniques," *Computer Graphics Forum*, vol. 23, no. 2, pp. 203-221, 2004.
[6] J.J. van Wijk, "Spot Noise-Texture Synthesis for data Visualization," *Proc. ACM SIGGRAPH '91*, vol. 25, pp. 309-318, 1991.
[7] A. Okada and D. Lane, "Enhanced Line Integral Convolution with Flow Feature Detection," *Proc. SPIE*, vol. 3017, pp. 206-217, Jan. 1997.
[8] L.K. Forssell and S.D. Cohen, "Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation, and Unsteady Flows," *IEEE Trans. Visualization and Computer Graphics*, vol. 1, no. 2, pp. 133-141, June 1995.
[9] H.W. Shen, C.R. Johnson, and K.L. Ma, "Visualizing Vector Fields Using Line Integral Convolution and Dye Advection," *Proc. IEEE Symp. Vol. Visualization*, pp. 63-70, Oct. 1996.
[10] L. Forssell, "Visualizing Flow over Curvilinear Grid Surfaces Using Line Integral Convolution," *Proc. IEEE Visualization*, pp. 240-247, 1994.
[11] H.W. Shen and D.L. Kao, "A New Line Integral Convolution Algorithm for Visualizing Time-Varying Flow Fields," *IEEE Trans. Visualization and Computer Graphics*, vol. 4, no. 2, pp. 98-108, Apr. 1998.
[12] M. Falk and D. Weiskopf, "Output-Sensitive 3D Line Integral Convolution," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 4, pp. 820-834, July 2008.

[13] D. Stalling and H. Hege, "Fast and Resolution Independent Line Integral Convolution," *Proc. ACM SIGGRAPH,* pp. 249-256, 1995.
[14] H. Battke, D. Stalling, and H. Hege, "Fast Line Integral Convolution for Arbitrary Surfaces," *Visualization and Math.,* pp. 181-195, 1997.
[15] G.S. Li, X. Tricoche, D. Weiskopf, and C.D. Hansen, "Flow Charts: Visualization of Vector Fields on Arbitrary Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 5, pp. 1067-1080, Sept./Oct. 2008.
[16] W. Heidrich, R. Westermann, H.P. Seidel, and T. Ertl, "Applications of Pixel Textures in Visualization and Realistic Image Synthesis," *Proc. Symp. Interactive 3D Graphics,* pp. 127-134, 1999.
[17] E. Zhang, J. Hays, and G. Turk, "Interactive Tensor Field Design and Visualization on Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 1, pp. 94-107, Jan. 2007.
[18] J. Palacios and E. Zhang, "Interactive Visualization of Rotational Symmetry Fields on Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 17, no. 7, pp. 947-955, July 2011.
[19] J.J. van Wijk, "Image Based Flow Visualization for Curved Surfaces," *Proc. IEEE 14th Visualization (VIS '03),* pp. 123-130, 2003.
[20] R. Laramee, B. Jobard, and H. Hauser, "Image Space Based Visualization of Unsteady Flow on Surfaces," *Proc. IEEE Visualization (VIS '03),* pp. 131-138, Oct. 2003.
[21] R.S. Laramee, J.J. van Wijk, B. Jobard, and H. Hauser, "ISA and IBFVS: Image Space-Based Visualization of Flow on Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 6, pp. 637-648, Nov./Dec. 2004.
[22] J. Huang, W. Pei, C. Wen, G. Chen, W. Chen, and H. Bao, "Output-Coherent Image-Space LIC for Surface Flow Visualization," *Proc. IEEE Symp. Pacific Visualization,* pp. 137-144, 2012.
[23] D. Weiskopf and T. Ertl, "A Hybrid Physical/Device-Space Approach for Spatio-Temporally Coherent Interactive Texture Advection on Curved Surfaces," *Proc. Graphics Interface,* pp. 263-270, 2004.
[24] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-Time Hatching," *Proc. ACM SIGGRAPH,* pp. 581-586, 2001.
[25] G. Chen, K. Mischaikow, R.S. Laramee, P. Pilarczyk, and E. Zhang, "Vector Field Editing and Periodic Orbit Extraction Using Morse Decomposition," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 4, pp. 769-785, July 2007.
[26] R.S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser, and H. Hagen, "Visual Analysis and Exploration of Fluid Flow in a Cooling Jacket," *Proc. IEEE Visualization,* pp. 623-630, 2005.
[27] T. Weinkauf and H. Theisel, "Streak Lines as Tangent Curves of a Derived Vector Field," *IEEE Trans. Visualization and Computer Graphics,* vol. 16, no. 6, pp. 1225-1234, Nov. 2010.
[28] S. Popinet, "Free Computational Fluid Dynamics," *ClusterWorld,* vol. 2, no. 6, 2004.
[29] L. Shi and Y. Yu, "Inviscid and Incompressible Fluid Simulation on Triangle Meshes: Research Articles," *Computer Animation and Virtual Worlds,* vol. 15, nos. 3/4, pp. 173-181, July 2004.
[30] B. Jobard, G. Erlebacher, and M.Y. Hussaini, "Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 8, no. 3, pp. 211-222, July 2002.
[31] A. Hertzmann and D. Zorin, "Illustrating Smooth Surfaces," *Proc. ACM SIGGRAPH,* pp. 517-526, 2000.
[32] N. Ray, W.C. Li, B. Lévy, A. Sheffer, and P. Alliez, "Periodic Global Parameterization," *ACM Trans. Graphics,* vol. 25, no. 4, pp. 1460-1485, Oct. 2006.
[33] N. Ray, B. Vallet, W.C. Li, and B. Lévy, "N-Symmetry Direction Field Design," *ACM Trans. Graphics,* vol. 27, no. 2, pp. 10:1-10:13, May 2008.
[34] R.S. Laramee, C. Garth, J. Schneider, and H. Hauser, "Texture Advection on Stream Surfaces: A Novel Hybrid Visualization Applied to CFD Simulation Results in Data Visualization," *Proc. Eighth Joint Eurographics/IEEE VGTC Conf. Visualization (EuroVis),* pp. 155-162, May 2006.

Jin Huang received the PhD degree from the Computer Science Department, Zhejiang University, China, in 2007, with Excellent Doctoral Dissertation Award of the China Computer Federation. He is an associate professor in the State Key Laboratory of CAD & CG at Zhejiang University, P.R. China. His research interests include geometry processing and physically based simulation. He has been serving as a reviewer for the ACM SIGGRAPH, the EuroGraphics, the Pacific Graphics, and the TVCG. He is a member of the IEEE.

Zherong Pan received the BS degree majoring in software engineering from Shanghai Jiaotong University, in July 2011. He is currently working toward the PhD degree in the State Key Lab of CAD & CG at Zhejiang University, P.R. China. His research interest includes physics-based animation and visualization.

Guoning Chen received the bachelor's degree from Xi'an Jiaotong University, China, in 1999, the master's degree from Guangxi University, China, in 2002, and the PhD degree in computer science from Oregon State University in 2009. He is an assistant professor at the Department of Computer Science, University of Houston. Before joining the University of Houston, he was a postdoctoral research associate at Scientific Computing and Imaging Institute, University of Utah. His research interests include scientific visualization, computational topology, geometric modeling, and computer graphics. He is a member of the IEEE and ACM.

Wei Chen received the PhD degree in Fraunhofer Institute for Graphics, Darmstadt, Germany, in July 2002. He is a professor in State Key Lab of CAD & CG at Zhejiang University, P.R. China. From July 2006 to September 2008, he was a visiting scholar at Purdue University. In December 2009, he was promoted as a full professor of Zhejiang University. He has performed research in computer graphics and visualization and published more than 60 peer-reviewed journal and conference papers in the last five years. His current research interests include visualization and visual analytics.

Hujun Bao received the BS and PhD degrees in applied mathematics from Zhejiang University, in 1987 and 1993, respectively. Currently, he is a professor and the director of State Key Laboratory of CAD & CG at Zhejiang University. His main research interests include computer graphics and computer vision, including real-time rendering technique, geometry computing, virtual reality, and 3D reconstruction. He is a member of the IEEE.